БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

ИНСТИТУТ ПО ИНФОРМАЦИОННИ
И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Д И С Е Р Т А Ц И Я

ЗА ПРИСЪЖДАНЕ НА ОБРАЗОВАТЕЛНА
И НАУЧНА СТЕПЕН „ДОКТОР“

РАЗПРЕДЕЛЕНА СИСТЕМА ЗА ПРОГНОЗИРАНЕ НА ВРЕМЕВИ РЕДОВЕ С ЕВОЛЮЦИОННИ АЛГОРИТМИ И ИЗКУСТВЕНИ НЕВРОННИ МРЕЖИ

Научна област:

4 „Природни науки, математика и информатика“

Професионално направление:

4.6 „Информатика и компютърни науки“

Научна специалност:

01.01.12 „Информатика“

Докторант:

инж. Тодор Димитров Балабанов

Научен ръководител:

проф. д-р инж. Васил Стефанов Василев

доц. д-р инж. Красимира Борисова Генова

Научен консултант:

проф. д-р инж. Милена Кирилова Лазарова

София, 2017

Съдържание

Списък на фигурите

Списък на таблиците

Списък на съкращенията

Съкращение на английски език	**Термин на английски език**	**Съкращение на български език**	**Термин на български език**
ANN	Artificial Neural Network	ИНМ	Изкуствена невронна мрежа
GA	Genetic Algorithm	ГА	Генетичен алгоритъм
BP	Back Propagation	ОРГ	Обратно разпространение на грешката
EA	Evolutionary Algorithms	ЕА	Еволюционни алгоритми
DE	Differential Evolution	ДЕ	Диференциална еволюция (еволюция на разликите)
PBMOA	Population-based metaheuristic optimization algorithm	ПА	Популационни алгоритми
RANN	Recurrent Artificial Neural Networks	РИНМ	Рекурентни изкуствени невронни мрежи
GMDH	Group Method of Data Handling	МГОА	Метод за групово отчитане на аргументи
BRFNN	Radial Basis Function Neural Network	ИНМРБФ	Изкуствени невронни мрежи с радиална базова функция
ROLS	Regular Orthogonal Least Square	РОСКА	Регулиран ортогонален средно квадратичен алгоритъм
SOM	Self-organized Map	СОК	Само-организираща се карта
SA	Simulated Annealing	СЗ	Симулирано закаляване
SVD	Singular Value Decomposition	РЕС	Разлагане на единични стойности
ALOPEX	Algorithms of Pattern Extraction	АИШ	Алгоритъм за извличане на шаблони
MA	Memetic algorithms	МА	Меметичен алгоритъм
LGP	Linear Genetic Programming	ЛГП	Линейно генетично програмиране
MAPE	Mean Absolute Percentage Error	САПГ	Средна абсолютна процентна грешка
ARIMA	Autoregressive Integrated Moving Average	АРИПС	Авто регресивна интегрирана плъзгаща средна
GPU	Graphic Processing Unit	ГИЕ	Графичен изчислителен елемент

Речник на използваните термини

Термин	Значение
bias	В българската терминология се използва „отместване“. Невроните, задаващи отместването, винаги имат на своя изход сигнал за максимална активация. По този начин всички тегла, свързани с тези неврони, подават активационен сигнал към невроните с които са свързани.
overfitting	В българската (а и в руската) терминология се използва „преобучение“. Това е ситуация при машинното обучение, когато системата много добре научава известните примери, но показва слаби резултати при непознати примери.
дълга поръчка	Извършване на търговска операция при която се залага на факта, че цената на търгуваната стока ще поскъпне.
къса поръчка	Извършване на търговска операция при която се залага на факта, че цената на търгуваната стока ще поевтинее.
меметичен алгоритъм	Алгоритъм основан на поведенчески модели. Най-често намира приложение в популационните и еволюционните алгоритми.
wavelet	Математическа функция, която се използва за разлагането на функции или непрекъснати във времето сигнали по честотни елементи.
donated computing power	Изчисления в разпределена среда, при които потребителите дарят (без заплащане) изчислителни ресурси на личните си компютри.
индикатор	При търговията на Форекс често се използват инструменти за технически анализ. Една от големите групи инструменти се наричат „индикатори“. Индикаторът представлява математическо преобразуване на информацията от финансовия времеви ред. Съществуват индикатори базирани на статистически формули, формули от редицата на Фибоначи и други.
осцилатор	Осцилаторите са вид инструменти в техническия анализ, които приличат на индикаторите, но най-често се визуализират в отделна графика спрямо финансовия времеви ред. Най-често се използват за обозначаване на „трептения“ в промяната на цената.

Увод

Множество задачи за планиране и анализ в метеорологията, икономиката, образованието, транспорта, екологията и други области могат да бъдат сведени до обработката и прогнозирането на времеви редове [GEJE1976]. Прогнозирането на времеви редове се подрежда на 3 място (от 4 възможни – категориини данни (categorical data), много вариантна статистика (multivariate statistics), времеви редове (time-series) и анализ на преживяването (survival analysis)) по приоритет в изследователските направления на научната дисциплина „Статистика“. Анализът на времеви редове е свързан с корелационния анализ, регресионния анализ, статистическия извод, събирането на данни и дескриптивната статистика. Задачите за прогнозиране на времеви редове могат да бъдат разделени на два отделни класа, в зависимост от тяхната формална постановка: 1. Методи в честотната област; 2. Методи в времевата област.

В дисертационната работа се разглеждат проблеми от втория клас, а именно със задачи за прогнозиране на времеви редове от валутните пазари във времевата област. При прогнозирането на времеви редове се определя стойност за наблюдаваната величина в бъдещи моменти от времето. В общия случай не съществува единствено решение което да гарантира абсолютна точност на прогнозата. Тъй като през N на брой точки в двумерното пространство могат да се прекарат безкраен брой криви, то от съществено значение са кривите които минимизират грешката от прогнозата и реалната величина. Намирането на подходяща крива предлагаща прогноза е едно-критериална оптимизационна задача в многомерното пространство. Броят параметри (при използването на изкуствени невронни мрежи това е броят на теглата в мрежата), характеризиращи кривата съответстват на размерността, на пространството. Тегловните коефициенти в изкуствена невронна мрежа е основен подход за представяне на параметри за крива, преминаваща максимално близо до N на брой точки в двумерното пространство.

Стойностите на теглата в изкуствената невронна мрежа се определят с помощта на точни оптимизационни числени методи или вероятностни (евристични) числени методи. Към групата на точните числени методи спада методът за обратно разпространение на грешката.

Обратно разпространение на грешката е градиентен оптимизационен метод целящ намаляването на грешката която мрежата допуска в изходния слой. Към групата на евристичните методи спадат популационните алгоритми каквито са генетичните алгоритми и диференциалната еволюция.

Основните резултати намерили място в дисертационната работа са свързани с прилагането на евристика (диференциална еволюция или известна също като еволюция на разликите) за обучение на изкуствени невронни мрежи, така че да пресмятат прогноза за предварително определен времеви ред.

Дисертационната работа се състои от четири глави и заключение. В първата глава е извършен анализ на съществуващите методи и подходи за решаване на задачи от прогнозирането на времеви редове. Във втора глава са описани разработените нови подходи за машинно обучение които разширяват възможностите за ускоряване на обучението и повишаване точността на прогнозите предлагани от изкуствени невронни мрежи. В трета глава са представени два метода и техните алгоритмични схеми които подобряват сходимостта в обучението на изкуствени невронни мрежи. В четвърта глава са описани предназначението, структурата, функциите и начинът на работа с разработената експериментална програмна система за прогнозиране на времеви редове. В заключението са изложени основните приноси от дисертационната работа.

При описанието на текста в отделните глави на дисертационната работа са използвани стандартни математически означения. Номерацията на формулите, таблиците и фигурите се отнасят за отделните точки в съответната глава.

Глава 1 - Обзор на системи за прогнозиране на времеви редове

Прогнозирането на времеви редове е сложна задача, която намира приложение в множество области в които по настояще усилено се работи. За създаването на прогнозиращи софтуерни системи се изискват множество знания и умения. Важността на задачите за прогнозиране е толкова голяма, че те намират приложение в почти всяка сфера от човешкия живот (температура, валежи, енергийни запаси и други). Подробен обзор за използването на изкуствени невронни мрежи, генетични алгоритми и размита логика за капиталовите пазари е представен в [SMSK2001]. Обзорът е съпроводен с кратко представяне на трите метода. Класификация на различни методи за прогнозиране на електрическото потребление е представено в [HAMN2002][JVGM2002] като също се разглеждат генетични алгоритми и изкуствени невронни мрежи. Обзор за използването на изкуствени невронни мрежи, генетични алгоритми и размита логика в областта на капиталовите пазари, с кратко представяне на методите, е достъпен в [ARSH2003]. Обзор за използваните алгоритми в икономиката също е представен и в [EUVO2005] като е направено представяне на възможностите за прогнозиране с изкуствени невронни мрежи и генетични алгоритми. При обзора на инструментите за прогнозиране на времеви редове с изкуствени невронни мрежи и еволюционни алгоритми е съществен аспектът за използването на съществуващи софтуерни решения описано в [RSQZ2006] и наличието на софтуерен пакети както описанията в [CDTY2006] (пакет създаден на програмния език C, за операционна система Mac OS X и след това е преработен за операционна система Microsoft Windows). Задълбочен обзор на методите за съставяне на инвестиционен портфейл и начините по които изкуствени невронни мрежи или генетични алгоритми могат да се ползват при тази задача е представен в [MVDL2008]. Обзор на видовете изкуствени невронни мрежи се предлага в [YAHU2009]. Също така се предлагат начини за тяхното приложение в земеделието. В [MSMJ2009] се предлага обзор на видовете изкуствени невронни мрежи и приложението им в геотехническото инженерство. Споменават се практически насоки за избор на размер в скрития слой (примерно 75% от размера на входния слой). Другата препоръка е размерът на скрития слой да бъде между средното и сумата от размера на входния и изходния слой. Споменава се и стратегия в която

прогресивно се започва от малък брой елементи в скрития слой и се увеличава броят докато няма подобрение в параметрите за обучение на изкуствени невронни мрежи. Препоръките също са броят на тренировъчните примери драстично да надхвърля броя на елементите в скрития слой. В [MAWE2010] е изложен изключително подробен обзор на евристичните оптимизационни алгоритми. Описва се паралелна реализация на диференциална еволюция с централен възел който синхронизира миграцията между останалите възли. Топологията е кръгова и мигрантите преминават само в посока към следващия пряк съсед. Всичко това се синхронизира от централния възел който на практика прави топологията звездовидна. В [JMAM2012] се излага обзор на пул базираните еволюционни алгоритми за реализацията на разширяеми и асинхронни изчисления в разпределена среда.

1.1 Изкуствени невронни мрежи обучавани с евристични алгоритми

Един от най-популярните подходи за оптимизация на теглата в многослойна изкуствена невронна мрежа е с помощта на генетични алгоритми. За потвърждаване на ефективността се прави сравнение с обратно разпространение на грешката и се показва ефективността на генетичните алгоритми пред алгоритъма за обучение с обратно разпространение на грешката. Има изследване на различни операции в генетичните алгоритми показано в [DMLD1989]. При някои модели, обучението на изкуствени невронни мрежи с фиксирана структура които са или еднопосочни или рекурентни се извършва с генетични алгоритми като се определя как да се променят теглата и отместванията (bias), но не самите стойности теглата и отместванията [JRVC1998]. Правено е сравнение между симулирано закаляване и генетични алгоритми. Оптимизират се теглата на изкуствени невронни мрежи като за репродуциране се избират тези хромозоми които дадат най-ниска средно-квадратична грешка (хромозомите се зареждат в изкуствена невронна мрежа). Направени са шест теста като данните са нормализирани в интервал -1 до +1. Основен проблем при симулираното закаляване е, че се нуждае от твърде много ръчни настройки на използваните параметри. Резултатите от сравнението са представени в [RSRD1999]. Резултати от обучението на пълно свързани изкуствени невронни мрежи, на които теглата се кодират в генетичен алгоритъм са представени в [HARS2001]. При някои реализации на

процедура за оптимизация на теглата в изкуствени невронни мрежи с използването на генетични алгоритми се търсят точки в които може да се направят „къси“ или „дълги“ поръчки като на входа на изкуствени невронни мрежи се подават стойности от пълзящи средни както е описано в [ASIC2002]. Генетични алгоритми може да се използват както за оптимизиране на теглата на връзките между слоевете в изкуствени невронни мрежи но и за избор кои стойности от времевия ред да се използват, с цел редуциране на данните и използване само на силно значими стойности както е описано в [LHBB2002]. В някои случаи се избират отделни свойства при прогнозирането на времевите редове с помощта на компактен генетичен алгоритъм. Самото прогнозиране се извършва с изкуствени невронни мрежи, генетични алгоритми или статистически методи [WLRP2002]. Възможни са варианти в които генетични алгоритми се използва за оптимизиране на центровете и радиусите в радиално базова изкуствени невронни мрежи. При тази реализация за оптимизация на теглата се ползва разлагане на единична стойност (SVD) [SOCL2002]. При точните числени методи е възможно теглата на изкуствени невронни мрежи да се представят като хромозоми в генетични алгоритми. След това върху избрани седем функции се прави тест за обучение на изкуствени невронни мрежи с генетични алгоритми или с обратно разпространение на грешката. При този вариант се разчита на добре подбрани първоначални тегла преди действителното обучение. Тестовете се провеждат при идентични архитектури без да се набляга върху оптимизирането на самата архитектурата [SVEK2003]. В [HFJL2003] се прави сравнение между симулирано закаляване и генетични алгоритми. Оптимизират се теглата на изкуствени невронни мрежи като за репродуциране се избират тези хромозоми които дадат най-ниска средно-квадратична грешка. Направени са шест теста като данните са нормализирани в интервал -1 до +1. Симулираното закаляване се нуждае от твърде много ръчни настройки на използваните параметри. Модел на разпределени генетични алгоритми за обучение на фронтални изкуствени невронни мрежи е представен в [VLOL2004]. В [JLJL2005] е представен алгоритъм за диференциална еволюция с който се обучават радиално базисни изкуствени невронни мрежи. В [MDMP2006] са представени модели на пълно-свързани изкуствени невронни мрежи на които теглата се кодират в генетични алгоритми с цел обучение на мрежата. В [KYKI2006] генетичен алгоритъм оптимизира теглата на връзките между слоевете на изкуствени невронни мрежи. Също така, генетичен алгоритъм избира кои стойности от времевия ред да се използват. Това се прави с цел да се намалят данните, така че да се използват само силно значими стойности. В [AOAC2007] се предлага

изкуствена невронна мрежа обучавана с генетични алгоритми за прогнозиране на стоковата борса в Истанбул. В [GNYZ2007] диференциална еволюция се ползва за оптимизация на теглата в изкуствени невронни мрежи. Вероятността за мутация се настройва автоматично за да може в началото да схожда по-бързо и да не попада в локални екстремуми. Експериментите са направени с математически подбрани „гладки“ функции. В [JDDH2007] генерализиран вариант на диференциална еволюция се ползва за обучение на изкуствени невронни мрежи. Целта на мрежата е да класифицира листа на конкретни растения. В [ATPG2007] е направена модификация на диференциална еволюция за обучение на изкуствена невронна мрежа при която теглата са само цели числа. Тестовете са направени над три общо известни проблема от обучението на изкуствени невронни мрежи. В [SKHK2008] се представя обобщена регресионна изкуствена невронна мрежа. При нея генетични алгоритми се използват за оптимизиране на теглата в мрежата. Тази системата се използва за прогнозиране на водната циркулация между земята и атмосферата. В [MPAC2008] се предлага фина настройка на параметрите в диференциалната еволюция, така че максимално добре да оптимизира теглата на изкуствени невронни мрежи. В [ABWD2008] е представено изследване за трите алгоритъма - диференциална еволюция, генетични алгоритми и алгоритъм за рояка от частици. Целта е да се оптимизират теглата в изкуствени невронни мрежи (тоест обучение на мрежата). При това изследване диференциалната еволюция показва по-добри резултати от другите алгоритми. В [ASMB2008] диференциална еволюция за оптимизация на теглата в изкуствени невронни мрежи се сравнява с други два подхода. Интересното при това изследване е, че единственият параметър който се задава предварително на диференциалната еволюция е броят на индивидите в популацията. В [HMAB2009] се предлага изкуствени невронни мрежи обучавани с диференциална еволюция за метеорологични прогнози. В [TTSS2009] се предлагат изкуствени невронни мрежи оптимизирани с диференциална еволюция която е модифицирана с дегенерация. Постига се с увреждане на гените и това води до дегенерацията им. В [IEMH2011] се представя рекурентна изкуствена невронна мрежа на която оптимизацията се прави с генетични алгоритми. Целта е да се прогнозира скоростта на вятъра за вятърни турбини които генерират ток. Този подход се сравнява с регресия от поддържащи вектори. В [NAFK2011] се предлага диференциална еволюция за обучение на изкуствени невронни мрежи които са базирани на куполовидни функции под формата на активационни функции. Прогнозира се силата на вятъра във ветрени полета. В [ABWD2011] се предлага смесен подход за диференциална

еволюция като оптимизация на параметрите в изкуствени невронни мрежи и на теглата. В [HSRG2011] се модифицира идеята за рояк от частици и се предлага изкуствена колония от пчели за обучение на изкуствени невронни мрежи. Моделът се използва при предсказанието на земетръсни времеви редове.

1.2 Използване на евристики за оптимизация на топологията, която изкуствените невронни мрежи използват

При някои реализации не се оптимизират теглата в мрежата, а се оптимизира топологията (връзките между възлите) на изкуствени невронни мрежи. Освен топологията, често обект на оптимизация стават и параметри за обучение. При този подхода за обучение се наблюдава ефекта overfitting. По-голям брой неврони в скрития слой намалява обобщаващите свойства на мрежата като нараства приучаването към конкретните обучаващи примери [CHSL1990][PCJM1996a][ANWE1997]. Индиректно може да се повлияе на топологията в изкуствените невронни мрежи като с помощта на генетични алгоритми се редуцира множеството от входни данни. На практика, това влияе на броя входове в изкуствените невронни мрежи и съответно на топологията от страната на входния слой. Когато популацията на генетичния алгоритъм се разпределя на различни машини това води до изчисление в разпределена среда. Чрез контрол на връзките между възлите в изкуствените невронни мрежи допълнително се намали размерът на цялата изкуствена невронна мрежа [JORI1991]. При някои реализации на генетични алгоритми (както е в [MAMA1995]) се кодирани параметрите за топологията на изкуствените невронни мрежи много по-подробно. Това включва информация за слоевете и връзките между тях (включително и обратни връзки) спрямо други предложени модели. Не само топологията на мрежата може да се оптимизира с генетични алгоритми, но също така и типът мрежа да бъде обект на оптимизацията както е представено в [BLLE1997]. В някои модели за оптимизация на топологията и параметрите на изкуствени невронни мрежи генетичните алгоритми се комбинират с допълнителни средства каквито са контекстно свободна граматика (граматична еволюция) описани в [XYYL1997]. В [ITDG2008a] и [ITDG2008b] също се предлага оптимизация на топологията и параметрите на изкуствени невронни мрежи с генетични алгоритми и контекстно свободна граматика (граматична еволюция). Използва се формална граматика (контекстно свободна) и се

предлага начин за разпределени изчисления. Табу търсене е друга възможност за оптимизация на параметрите на изкуствени невронни мрежи описана в [RSBA1998]. При някои реализации на генетични алгоритми се оптимизира структурата на изкуствени невронни мрежи (брой скрити слоеве и брой неврони в тях), но и параметрите за обучение като активационна функция за всеки слой, наличие/отсъствие на отместване (bias) и най-подходящ начин за корекция на теглата както е описано в [IDAD1998]. При някои реализации на генетични алгоритми за оптимизация на изкуствени невронни мрежи (типа на входовете, броя неврони в скрития слой и вида на връзките между слоевете) е съществено как влияе елитарността. Проверката при това изследване е направена над данни от валутния пазар и е представена в [MATO1999]. При добра дефиниция на времевите редове е възможно прилагането генетично базирани самоорганизиращи се изкуствени невронни мрежи описано в [KMKC2000]. При някои модели с генетични алгоритми се оптимизира топологията на изкуствени невронни мрежи, но и началните стойности на връзките. При този вариант изкуствени невронни мрежи се обучава с обратно разпространение на грешката за да се получи жизнената стойност на индивида в популацията [RSRD2000]. При някои модели топологията не се манипулира директно с генетични алгоритми, а генетичните алгоритми се използват за редукция на множеството от входни данни което индиректно влияе на броя входове в изкуствените невронни мрежи. При тези модели генетични алгоритми не се ползват за директна оптимизация на топологията или за оптимизация на теглата. При конкретната реализация описана в [VPMV2000] популацията на генетичните алгоритми се разпределя на различни машини което е вид изчисление в разпределена среда. Също така се контролират и връзките между възлите в изкуствените невронни мрежи, така че да се намали размерът на цялата мрежа. Използване на генетични алгоритми за постигане на оптимална топология на изкуствени невронни мрежи при прогнозиране на стойностите на стоковите пазари е представено в [EFKA2001]. Изследване за влиянието на елитарността в генетичните алгоритми при което се определя типа на входовете, броя неврони в скрития слой и вида на връзките между слоевете в изкуствени невронни мрежи е представено в [JARG2001]. Сравнителният анализ при тази разработка е над данни от валутните пазари. Възможно е в генетичните алгоритми да се кодират много по-подробно параметрите за топологията на изкуствени невронни мрежи (слоеве, връзки между слоевете, обратни връзки), както е представено в [RPTL2002]. Изкуствени невронни мрежи със забавяне във времето с цел попълване на липсващи данни при автомобилен трафик се проектират с генетични алгоритми

(определя се топологията), а обучението се извършва с обратно разпространение на грешката представено в [EAMT2002][MZPL2004]. В [JIJK2003] също се представят генетични алгоритми който оптимизира структурата на изкуствени невронни мрежи (брой скрити слоеве и брой неврони в тях) и параметрите за обучение (активационна функция за всеки слой, наличие/отсъствие на „отместване“ и най-подходящ начин за корекция на теглата). При прогнозирането на финансови времеви редове се ползват данни за отваряне, затваряне, най-ниска, най-висока и обем. При [DZDP2003] се прогнозира следващото ниво на отваряне като изкуствени невронни мрежи се обучава с обратно разпространение на грешката, но топологията се формира с еволюционен алгоритъм. При сравнителен анализ на многослойна изкуствена невронна мрежа (обучавана с обратно разпространение на грешката) с рекурентна изкуствена невронна мрежа (също се обучава с обратно разпространение на грешката), топологиите на мрежите могат да се формират с еволюционен алгоритъм както е представено в [DAZA2003]. На база превключващи елементи топологията на изкуствени невронни мрежи може да се оптимизира с генетични алгоритми което е представено в [JDKM2003]. Както е представено в [MVRB2004] с генетични алгоритми може се избират не само параметри на мрежата като тип мрежа, топология, но и параметри на обучаващо множество. Паралелен генетичен алгоритъм описан в [HNTH2004] се използва за прогнозиране на замърсяването на въздуха на спирка от градския транспорт. При тази реализация генетични алгоритми се използва за избор на входа и проектиране на архитектурата (брой скрити слоеве и техните размери) в пълно-свързана (между слоевете) изкуствени невронни мрежи. Една от най-интересните разработки,представена в [EHMW2004] представлява генериране на изкуствени невронни мрежи с обратни връзки, чрез използване на еволюционни алгоритми. Точно преди да се добавят обратните връзки е проведено обучение с обратно разпространение на грешката както във [ROBA2009]. При тази реализация изкуствена невронна мрежа измерва и достоверността на прогнозата. Всяка стойност от исторически представените се подава на отделен входен елемент. Изходът се получава само в един изходен елемент. Достоверността на прогнозата се взема от допълнителен изходен елемент в изходния слой. При [MASE2005] се прогнозира с ден напред цената на затваряне. За целта се използва изкуствена невронна мрежа и генетични алгоритми като входните данни са индикатори от техническия анализ. Информацията от индикаторите се подава на входа на изкуствени невронни мрежи. Топологията на изкуствените невронни мрежи също подлежи на оптимизация с генетични алгоритми. В [RPSL2005] е представена изкуствена невронна мрежа за прогнозиране на

локалната безработица в Германия. При това изследване структурата на изкуствените невронни мрежи се оптимизира с генетични алгоритми. Точна и ясна дефиниция на времевите редове и задачата за прогнозирането им е предложена в [LESP2006]. Моделът стъпва на генетично базирани самоорганизиращи се изкуствени невронни мрежи. При [SSST2006] многослойна изкуствена невронна мрежа се обучава с обратно разпространение на грешката и се сравнява с рекурентна изкуствена невронна мрежа която също се обучава с обратно разпространение на грешката. В работата е направен сравнителен анализ между двата вида мрежи като топологиите на мрежите са постигнати с еволюционни алгоритми. В [MHRK2009] се представят изкуствени невронни мрежи с превключващи елементи. Топологията на изкуствените невронни мрежи се оптимизира с генетични алгоритми. В [AYGU2009] се представя линейно генетично програмиране за прогнозиране на потока в речна станция. С генетични алгоритми се оптимизира топологията на изкуствени невронни мрежи. При извършения сравнителен анализ линейното генетично програмиране се представя по-добре от изкуствените невронни мрежи. В [JPGG2010b] се предлага генетичен алгоритъм за оптимизация на топологията на изкуствени невронни мрежи и началните стойности на връзките. Обучението на изкуствените невронни мрежи се извършва с обратно разпространение на грешката за да се получи жизнената стойност на индивида в популацията. В [CLCW2011] се представя модифициран алгоритъм за диференциална еволюция. Целта на изследването е чрез диференциална еволюция да се проектира функционално размита рекурентна изкуствена невронна мрежа. В [YLYL2011] се предлага диференциална еволюция за оптимизация на изкуствени невронни мрежи. Архитектурата на изкуствените невронни мрежи може да се приема като целочислено оптимизиране, тъй като параметрите са цели числа. Получените изкуствени невронни мрежи се проверяват за ефективност с прогнозиране на времеви редове.

1.3 Едновременна оптимизация на теглата в изкуствените невронни мрежи и на топологията

Възможно е оптимизацията на изкуствени невронни мрежи да протича и по двете направления (тегла и топология). Когато обучението се извършва в хетерогенна мрежа от компютри се постига разпределено обучение на изкуствени невронни мрежи с генетични

алгоритми. За целите на обучението са доказани съответните теореми в [SOMF1992]. При комбинираната оптимизация в последващи реализации паралелно на топологията се оптимизират и теглата, но с най-близкия съсед при изкачване по хълм . Целта е да се постигне минимално голяма изкуствена невронна мрежа която обаче запазва възможностите си за генерализиране. Изкуствените невронни мрежи се представя като дървовидна структура. Получава се така, че всеки изход на изкуствената неврона мрежа е дървовидна структура която представлява формална граматика [BZHM1993]. В [JERI1997] чисто формално под формата на теореми е доказана ефективността от разпределеното обучение на изкуствени невронни мрежи с генетични алгоритми като се оптимизира както топологията така и теглата. При модела описан в [RSNS2001] се използва изкуствена невронна мрежа в която с генетични алгоритми се оптимизират теглата. Ползват се двадесет индивида в популацията (десет стари и десет нови) и единична точка на срязване при кръстосването. Прилага се двустепенна мутация в която тегла се блокират на стойност 0 което на практика отрязва връзката между невроните и индиректно влияе на топологията. При този модел изкуствената невронна мрежа е с три слоя и скритият започва с 1 неврон като на всеки 100 цикъла обучение се добавя нов неврон в скрития слой. При три последователни добавяния на неврони, ако няма подобрение в обучението, то скритият слой се фиксира и се правят още 2500 цикъла на обучение. Възможни са реализации при които паралелно на топологията се оптимизират и теглата, но с най-близкия съсед при изкачване по хълм. В този случай целта е да се постигне минимално голяма изкуствена невронна мрежа със запазва възможностите за генерализиране. Всеки изход на изкуствената невронна мрежа се явява дървовидна структура от вид формална граматика [BCJB2003]. В [BYXH2006] се предлага диференциална еволюция за обучение на радиално базисни изкуствени невронни мрежи като се оптимизира архитектурата. В [ARGH2012] се прогнозират финансови времеви редове като от данните за отваряне, затваряне, най-ниска, най-висока и обем се прогнозира следващото ниво на отваряне. Изкуствена невронна мрежа се обучава с обратно разпространение на грешката, но топологията се формира с еволюционен алгоритъм. В [MDMB2012] се прогнозират валутни курсове с изкуствени невронни мрежи. Генетичен алгоритъм се ползва за оптимизиране на параметрите на изкуствени невронни мрежи, но самото обучение се извършва с обратно разпространение на грешката.

1.4 Хибридни модели с точни числени методи и/или евристични

В литературата са познати модели за обучение базирани на точни числени методи и евристични алгоритми за оптимизация. В [PCJM1996b] е представена пълно-свързана изкуствена невронна мрежа, обучавана с генетични алгоритми и обратно разпространение на грешката. При това изследване експериментите са направени с програми написани на програмните езиците C и Prolog. При един от хибридните подходи теглата на изкуствените невронни мрежи се представят като хромозоми в генетични алгоритми. Върху избрани функции се прави тест за изкуствени невронни мрежи обучавани с генетични алгоритми или с обратно разпространение на грешката. Разчита се на добре подбрани първоначални тегла за изкуствените невронни мрежи. Тестовете са проведени при идентични архитектури без да се набляга върху оптимизирането на архитектурата [RSRD1998]. Възможно е да се използва диференциална еволюция за обучение на изкуствени невронни мрежи като теглата са само цели числа описано е в [VPDS1998a]. Тестовете са направени над три общо известни проблема от обучението на изкуствени невронни мрежи. Хибридна оптимизация е възможна с локално търсене и клетъчен генетичен алгоритъм за обучение на рекурентни изкуствени невронни мрежи. При този вариант теглата на изкуствените невронни мрежи се кодират в генетични алгоритми както е описано в [VPDS1998b] и [KKMM1999]. Хибриден метод за оптимизация може да се базира на GMDH и генетични алгоритми като се търси първо оптимална стойност за броя на невроните в скрития слой на полиномиална мрежа, а след това оптимална стойност за теглата описано в [AMFO1999]. При хибридните реализации е възможно да се използва йерархия на две нива за конструиране на BRF мрежи на база комбиниран генетичен алгоритъм с регулиран ортогонален средно квадратичен алгоритъм (ROLS) представено в [SCYW1999]. При по-сложните хибридни реализации се среща използването на два типа изкуствени невронни мрежи - класическа (многослоен перцептрон) и само-организираща се карта. Само-организиращата се карта служи за предварително класифициране на входните нива в пет групи за които се ползват пет различни вида изкуствени невронни мрежи от тип многослоен перцептрон. При тази реализация се ползва размита логика и генетични алгоритми за да се комбинират прогнозите от изкуствените невронни мрежи и да се получи единна прогноза. Този модел успешно е използван за прогнозиране на наводнения и е представен в [LSSO1999]. Когато рекурентни изкуствени

невронни мрежи са тренирани с еволюционни алгоритми може да се оптимизира и топологията, а не само теглата на мрежата. В началото на обучението има предварително тренирана чрез обратно разпространение на грешката като изкуствената невронна мрежа е само с връзки напред [JYJH1999]. Възможно е използването на изкуствени невронни мрежи с обратно разпространение на грешката за прогнозиране на позициите в пространството на мобилен терминал като се използва хибриден алгоритъм с обратно разпространение на грешката и генетични алгоритми за модифициране на теглата [CDAH1999]. Възможни са модели базира на комбинация от изкуствени невронни мрежи, стационарни wavelet трансформации и статистически анализ на времеви редове както е представено в [EAJT1999]. При хибридните реализации на два типа изкуствени невронни мрежи - класическа (многослоен перцептрон) и само-организираща се карта, само-организиращата се карта предварително класифицира входните нива в групи. За всяка от групите се ползват различни вида изкуствени невронни мрежи от тип многослоен перцептрон. С помощта на размита логика и генетични алгоритми се комбинират прогнозите от няколко изкуствени невронни мрежи и се получи единна консолидирана прогноза. Този хибриден подход е приложен за прогнозиране на наводнения и е представен в [DIPI2002]. При трениране на рекурентни изкуствени невронни мрежи с еволюционни алгоритми може да се оптимизира и топологията заедно с теглата на мрежата. При разработката представена в [ERHU2004] началото на обучението има предварително тренирана изкуствена невронна мрежа (чрез обратно разпространение на грешката) като се ползват само връзките в права посока. При по-сложните хибридни модели се съчетават няколко метода за обучение на изкуствени невронни мрежи. В [HIAL2004] е представен модел, който се базира на диференциална еволюция, генетични алгоритми, обратно разпространение на грешката и симулирано закаляване. В [DWFG2005] са представени рекурентни изкуствени невронни мрежи с оптимално линейно съпоставяне. В [MUAB2005] се ползва комбинация от изкуствени невронни мрежи и генетични алгоритми за апроксимация на липсващи стойности в база данни. Генетичен алгоритъм има за задача да предложи заместители на липсващите стойности. Целта на тази апроксимация е да се намали грешката допускана от изкуствените невронни мрежи. При изкуствените невронни мрежи които прогнозират времеви редове е нужно данните на входа да се нормализират. В [LOBO2004] се предлага най-често използваната формула за нормализация между 0 и 1. В [YCBY2005] са представени гъвкави невронни дървета за прогнозиране на времеви редове. Йерархичната структура е развита с помощта на

вероятностен програмен еволюционен алгоритъм. Финото настройване на параметрите се прави със симулирано закаляване. В [ZPLL2006] е представен хибриден алгоритъм с обратно разпространение на грешката и генетични алгоритми за модифициране на теглата. Системата се използва за прогнозиране на позициите в пространството на мобилен терминал. В [PDAA2006] се прави прогнозиране на продажбите с радиално базисни изкуствени невронни мрежи и генетични алгоритми. В [PGSC2007] се предлага трислойна изкуствена невронна мрежа която се обучава с обратно разпространение на грешката за прогнозиране на натоварването на електрическата система в щата Илинойс. В [NMMS2008] се използват изкуствени невронни мрежи с обратно разпространение на грешката за прогнозиране на парите налични в банкоматите. Използва се плъзгащ прозорец върху множеството от входни данни. В [XJHA2008] се предвиждат структурните измествания в конструкцията на сградите. Генетичен алгоритъм за намиране на оптимално управление на силите при високи сгради със стоманена конструкция. Генетичен алгоритъм се комбинира с размит динамичен wavelet невро-емулатор. В [BSDJ2008] е представен меметичен алгоритъм за диференциална еволюция (прилага се локално търсене в околност на индивидите). Локалното търсене се извършва с обратно разпространение на грешката и е в околностите на конкретния индивид. В [LMRB2009] се представя обобщен авторегресионен условен подбор на случайни променливи за захранване на изкуствени невронни мрежи с данни. В [JSKU1998] също се предлага обобщен авто регресивен условен подбор на случайни променливи за захранване на изкуствени невронни мрежи с данни. В [CHTA2009] се представя модел базиран на изкуствени невронни мрежи, стационарни wavelet трансформации и статистически анализ на времеви редове. В [TGTT2009] се представя изкуствена невронна мрежа написана на програмния език Python с библиотеката FANN за прогнозиране на водните ресурси в Кипър. В средния слой се добавят неврони и така мрежата нараства докато се обучава. В [MAHL2009] се предлагат изкуствени невронни мрежи с превключване на елементите. Използва се за дневните обороти на валута според особеностите на сезона (работни дни, почивки и други). В [BSDJ2009] се предлагат изкуствени невронни мрежи обучавани с диференциална еволюция за нелинейна идентификационна система като ползва хибриден модел с градиентен метод. В [SLKG2009] се предлага хибриден модел на изкуствени невронни мрежи с диференциална еволюция, така че се оптимизират описателните параметри на мрежата. Като параметри това са брой на елементите в скрития слой, активационна функция във входния слой, активационна функция в изходния слой и

коефициент за обучение (learning rate). В [KBWK2009] се предлага алгоритъм за обучение на изкуствени невронни мрежи с хибриден подход между диференциална еволюция и градиентен метод. След всяко кръстосване и мутация се пуска градиентен оптимизационен алгоритъм. Значително по-добре се избягва попадането в локален оптимум, но обучението е по-бавно от чисто градиентен метод. В [SDAA2009] се предлага хибридна схема за мутация. Изборът на вектор за мутацията се извършва в малка околна област от съседи в общата популация. В [JPGG2010a] се предлага подход за автоматизирано настройване на всички параметри които изкуствените невронни мрежи може да имат. Параметрите на изкуствените невронни мрежи се кодират в хромозомите на еволюционни алгоритми и се проверява коя изкуствена невронна мрежа ще постигне най-добри резултати след обучение с обратно разпространение на грешката. В [JDXL2011] също се предлага подход за автоматизирано настройване на всички параметри които изкуствени невронни мрежи може да имат. Параметрите на изкуствените невронни мрежи се кодират в хромозомите на еволюционни алгоритми и се проверява коя изкуствена невронна мрежа ще постигне най-добри резултати след обучение с обратно разпространение на грешката. В [MKMB2010] се предлага интегриран авто регресионен подход на пълзящи средни като на втора фаза се ползва изкуствена невронна мрежа. В [ANBG2011] също се предлага хибриден модел на изкуствени невронни мрежи с използването на авто регресивна интегрирана плъзгаща стойност (ARIMA). В [MKMB2011] се предлага комбинация между линеен модел на авто регресивна плъзгаща средна и нелинеен модел на изкуствени невронни мрежи за прогнозиране на финансови времеви редове. В [ABEB2012] се извършва прогнозиране на финансови времеви редове с хибридна комбинация от авторегресивни пълзящи средни и генетично програмиране при което индивидите в популацията са математически функции представени като символни низове. В [SKYS2010] се избират отделни свойства при прогнозирането на времеви редове с помощта на компактен генетичен алгоритъм. Самото прогнозирането се извършва с изкуствени невронни мрежи, генетични алгоритми или статистически методи. В [MOAW2010] се предлага генетични алгоритми за оптимизиране на центровете и радиусите в радиално базова изкуствена невронна мрежа. За оптимизация на теглата се ползва декомпозиция по сингулярни числа (SVD). В [MMMH2010] се предлага прогнозиране на индийския стоков пазар, стойност на затваряне с изкуствени невронни мрежи. Входната информация се мащабира в интервал от 0 до 1, като се ползва най-голямата и най-малката постигната стойност. Скритият слой е с по-малко на брой неврони спрямо входния слой.

Използва се само един неврон за изход. Точността на прогнозата се определя с нормализирана средно-квадратична грешка. В [RGAC2010] се предлага иновативен подход за обучаване на изкуствени невронни мрежи с модифициран оптимизационен алгоритъм, наречен инвазивна плевелна оптимизация. Индивидите в популацията се представят като плевели които разпръскват семена в околните точки в пространството на търсенето. Плевелите с най-голяма жизненост генерират най-много семена, а тези с най-ниска жизненост най-малко семена като разпределението на бройките семена е в линейна зависимост от най-ниска жизненост към най-висока жизненост. Интересна модификация за правило на разпръскването може да бъде интензивност с r^2 както е влиянието на гравитационното поле при планетите. Жизнеността на отделните индивиди може да служи като константа за гравитационно поле. В [JMAP2011] се представя сравнително изследване на четири различни модела изкуствени невронни мрежи за грешката която допускат при прогнозирането на времеви редове (рамките на 10%). Всичките изкуствени невронни мрежи са с дванадесет входа и един изход. За измерване на допуснатата грешка се ползва MAPE метрика. В [IVTL2011] се предлага многослойна изкуствена невронна мрежа към която се подават ограничен брой параметри. По-незначителните входни променливи се отсяват с хибридни методи. В [PPAM2011] се предлага еднослойна изкуствена невронна мрежа с един изходен неврон за прогнозиране на финансова информация. Скритият слой е премахнат заради функционално разпространение на входния шаблон. Използва се динамично настройване на коефициента за обучение. В [HTBM2011] също се предлага еднослойна нелинейна архитектура на изкуствени невронни мрежи с използване на нелинейно съпоставяне на входните сигнали. За обучението се използва диференциална еволюция като оптимизационен алгоритъм. В [YWWL2011] се представя хибриден модел състоящ се в комбинация между диференциална еволюция и оптимизация на рояка от частици. В [HKMS2012] се прави сравнение между диференциална еволюция и рояка от частици за обучение на изкуствени невронни мрежи. За нуждите на сравнителния анализ се прогнозират стокови и финансови инструменти. Диференциалната еволюция дава по-добри резултати като и при двата алгоритъма на входа на изкуствените невронни мрежи се подават стойностите на няколко индикатора. В [JNJS2011] се предлагат топологично активни мрежи за разпознаване на сегменти в изображения. Методът се комбинира с локални лакоми алгоритми. В [IBAT2012] се предлага оптимизиране на параметрите в wavelet и прогнозиране на времеви редове с изкуствени невронни мрежи. Теглата се кодират в хромозомите. Wavelet-ът се ползва

за предварителна обработка на входната информация и последваща обработка на изхода от мрежата. В [RAVA2012] се предлага интеграция между изкуствени невронни мрежи, wavelet функция и база знания с размита логика. В [RSKP1995] са представени мрежи с обратно разпространение на грешката за прогнозиране на парите налични в банкоматите. Използва се плъзгащ прозорец върху множеството от входни данни.

1.5 Еволюционни и/или популационни алгоритми за прогнозиране, без използване на изкуствени неверонни мрежи

Съществуват и системи в които не се използват изкуствени невронни мрежи, но въпреки това еволюционните алгоритми намират приложение. Такава система е описана в [SMGM1996] и [FLHL2003] където генетичен алгоритъм оптимизира група от правила и реално се развива експертна система без да се използва експерт човек. При други реализации генетичен алгоритъм получава запис от времевия ред и предоставя вектор от подходящи характеристики (редуциране на данните). Прилага се генетично програмиране, а програмата създадена по този начин прави редуциране на характеристиките от входния времеви ред. Използват се двадесет операции и се строи дърво на изразите [DEDH2002] и [VPMV2002]. Когато става въпрос за пресмятания в разпределена среда то моделите за еволюционни алгоритми могат да бъдат приложени. Като пример в [CGMP2003a] се използва модел на изолирания остров. Еволюционният алгоритъм в [TBWK2003] е разделен на главен, който извършва операциите по еволюционния алгоритъм и подчинени в които се пресмята жизнената функция. Общата популация е разделена на под-популации и между тях се прави миграция на хромозоми. Динамично разпределение на размера от хромозоми под формата на главен-подчинен за множество разпределени еволюционни популации е представен в [CGMP2003b]. При този вариант изчисляването на жизнената функция се извършва на отдалечения възел. В [JIHU2004] е представен генетичен алгоритъм който с ползва за намиране на коефициенти в ред на Тейлър. Този ред се използва като прогнозираща функция. Този подход е сравнен с алгоритъм който е комбинация между генетични алгоритми и изкуствени невронни мрежи. При тази реализация изкуствени невронни мрежи се ползват за определяне на това как двата родителя да се кръстосат в генетични алгоритми. Генетичен алгоритъм има за задача да определя и размерът на скрития слой. Паралелен еволюционен

алгоритъм за откриване на правила за решения при анализ на големи по обем данни е представен в [WOKW2004]. Разпределянето на задачите за пресмятане е на принципа „главен-подчинен“. В хромозомите се кодират множество от правила. Тъй като най-бавната част е изчислението на жизнената функция точно тя се възлага на подчинените възли. В [NPVP2006] се използва генетично програмиране за развитие на по-ефективни операции в диференциалната еволюция. Обстойно сравнение на осем вариации на диференциална еволюция е представено в [EMJV2006]. Сравнението е насочено в посока до колко ефективно различните вариации изпълняват глобална оптимизация. В [NWZM2007] се предлага динамично генетично програмиране за прогнозиране на брутния продукт и индекса на инфлация в САЩ. Алгоритъмът се адаптира по отношение на прозореца който се формира от входните исторически данни. Също така се ползва и предишно натрупано знание с цел по-бързо обучение. В [DXSL2007] се предлага подобрение на диференциалната еволюция базирано на ALOPEX операция и селекция на памет. В [ASML2007] диференциална еволюция се модифицира, така че изборът на индивиди за диференциалния вектор да не става на равно-вероятностен принцип, а да се прилага стратегия за селекция. В [ZAKA2007] се развиват четири модификации на диференциална еволюция. Първият алгоритъм е с наказателна функция за управление на ограниченията. Вторият алгоритъм се базира на първия, но включва множество от филтри като механизъм за подобряване на разнообразието. Третият алгоритъм е базиран на диференциална еволюция, но включва локален алгоритъм за подобряване на решенията който се базира на техника за търсене на шаблони. Четвъртият алгоритъм комбинира множеството от филтри и техниката за търсене на шаблони. В [RTSY2007] е предложен само настройващ се алгоритъм за мутация в еволюционните алгоритми. Идеята е базирана на q-Гаусово разпределение. Параметърът q се кодира в хромозомата и също еволюира като по този начин променя формата на вероятностното разпределение. В [SDAA2008] се предлага настройка на скоростта в рояк от частици с помощта на диференциална еволюция. Целта е да се създаде по-ефективна евристика за глобална оптимизация. В [MEVP2008] се представя само-балансираща се хибридна операция за мутация в диференциална еволюция. В [SRHT2008] се представя подобряване на диференциална еволюция чрез опозиционно базирано обучение за подобряване на скоростта за оптимизация. Използват се противоположни числа вместо напълно случайни. За всяко открито решение се проверява и противоположното му като от двете се избира по-доброто. В [TCPL2009] се прави сравнение между еволюционните алгоритми и алгоритми за

прогнозирано разпределение. Вторият тип алгоритми дава по-добри резултати при търсене на под низове. В [WGAF2010] се предлага адаптивна стратегия за селекция в диференциална еволюция базирана на вероятностно съпоставяне и откриване на близост между индивидите. Прави се пул от възможни стратегии за селекция и се напасва коя от тях би била най-удачна за конкретния проблем който се решава. В [NNHI2010] се предлага клетъчна реализация на диференциална еволюция. При този вид организация генетичните операции се прилагат само върху индивиди които се намират близко едни до други в популацията. Идеята идва от разпределените реализации където се преследва по-добра ефективност в рамките на отделните изчислителни възли. При тази организация има малка площ на припокриване между отделните локални популации и това води до бавна дифузия на информацията. В [RMPS2010] се предлага пул от възможни стратегии за мутация, кръстосване и параметрите на диференциална еволюция (вероятност за кръстосване, множител за диференциалния вектор, размер на популацията). Използването на такъв пул е продиктувано от факта, че различните оптимизационни проблеми се решават различно ефективно при различни стратегии за кръстосване, мутация и контролни параметри. В [MEVO2011] се представя генетични алгоритми за подбор на множество от финансови индикатори които с добра ефективност показват сигнали за купуване или продаване. В [YGSG2011] се предлага модифицирана операция за мутация на диференциална еволюция която се комбинира със стратегия за разделянето на популацията по определен принцип и след това събиране на разделените части. Част от експериментите се извършват върху задача за прогнозиране на времеви редове. В [AIXL2011] се предлага модификация на диференциална еволюция, така че да има насоченост при подбора на следващи точки в пространството на търсене. В [WGZC2011] се предлага механизъм за адаптивна стратегия приложен върху диференциална еволюция. Различните стратегии се организират в пул от който се избират. В [YWZC2011] се предлагат три различни стратегии за генериране на пробен вектор и контролни параметри (размер на популацията, вероятност за кръстосване и фактор за преоразмеряване) в диференциална еволюция. Комбинацията между трите стратегии за създаване на пробен вектор и трите параметъра се избира на случаен принцип. В [PCPV2011] се предлага многокритериална оптимизация на мрежови графици с генетични алгоритми. В [AMJT2012] се предлага геометрична диференциална еволюция за дискретни и непрекъснати пространства.

1.6 Алгоритми за пресмятане в разпределена среда

При използването на евристични методи за оптимизация е приложимо използването на изчисления в разпределена среда. Както е описано в [DTNP2004] под популациите на диференциалната еволюция са организирани в ринг и най-добрите индивиди от всяка популация могат да мигрират към следващия съсед. Диференциална еволюция която се ползва за откриването не само на един оптимум, но на множество оптимуми в разпределена среда е представен в [DAZA2004a]. При тази реализация основната цел е да се избягват пресмятанията в централната популация. Сравнителен анализ за възможностите на различни разширения на диференциална еволюция да откриват и поддържат множество оптимуми е представено в [DAZA2004b]. Хибридните подход за диференциална еволюция се предлагат под формата на изчисления в разпределена среда. Когато оптимизацията се осъществява на повече от един компютър са актуални въпросите за миграцията на индивиди между отделните локални популации. Изследване на степента за миграция и честотата на миграция в разпределена диференциална еволюция е представено в [EAGL2004]. Реализация на симулирано закаляване в разпределена среда е представено в [MATF2004]. При тази разработка, като изчисления в разпределена среда е използвана паралелната версия на алгоритъма за симулирано закаляване. Целта на изследването е да се избегне изпадането в локален минимум и по-възможност да се ускори търсенето на оптимално решение. В [BHKT2005] е представен еволюционен Монте-Карло метод. Реализацията е под формата на изчисления в разпределена среда. Като база на алгоритъма се използва вариация на генетични алгоритми. Акцентът при тази разработка е върху алгоритъм за миграция на индивидите между отделните под популации. Паралелен алгоритъм за диференциална еволюция е приложен за обучение с учител на импулсни изкуствени невронни мрежи в [NPDT2005]. При импулсните изкуствени невронни мрежи възлите се активират в определен момент от времето (като импулс). Паралелната обработка се получава с отделни подмножества на популацията които се изпълняват независимо на отделни процесори. Миграцията на най-добрите индивиди се осъществява в кръгова топология. В [GWDG2005] се прави обобщено изследване за паралелните еволюционни алгоритми. Най-простият вариант е когато има една популация в главния възел, а само жизнената функция се пресмята в подчинените изчислителни възли. Вторият, малко по-сложен вариант, е разделяне на общата

популация в множество по-малки популации които се изпращат на изчислителните възли. При този модел се извършва миграция между възлите. Третия модел отново има множество изчислителни възли и те държат малко на брой индивиди от популацията, но рекомбинацията между индивидите става глобално. Четвъртият вариант е хибриден и представлява комбинации от първите три. Разпределена система за класификация с генетични алгоритми е представена в [VBGM2006]. Системата използва изкуствена невронна мрежа която оценява параметри от класификацията като оценката е жизнена функция за генетичния алгоритъм. В [KKAS2006] се предлага модификация на стратегията за миграция при паралелна реализация на диференциална еволюция. Паралелна имплементация на диференциална еволюция е с цел подобряване на скоростта за търсене на оптимално решение. Използва се конкретна схема за миграция между възлите. Организацията на пресмятанията е в кръгова топология на миграция. При миграцията най-добрият индивид замества най-стария индивид в локалната популация на съответния следващ възел. В [DZGC2006] на база мембранните системи се предлага подход по който се прилагат операциите в еволюционните алгоритми. Всяка операция се прилага с избиране от вероятностно разпределение. Популацията е с променлив размер като нараства при кръстосване и мутация и намалява при селекция. Също се предлага и обогатяване на популацията със случайно генерирани индивиди. Идеите от мембранните изчисления се прилагат при стратегиите за миграция в разпределени еволюционни алгоритми. В [WRWI2006] е представена децентрализиран еволюционни алгоритми в peer-to-peer архитектура. Използва се задачата за осемте царици, така че да се провери ефективността на разпределената реализация на алгоритъма. Същата идея е доразвита в [WWMS2007] като разпределен еволюционен алгоритъм с адаптивна автономна селекция. Упражнява се стриктен контрол върху размера на популацията, така че да не се предизвика експлозия или имплозия. Задачата за осемте царици се използва за потвърждение на надеждността в peer-to-peer реализацията без наличието на какъвто и да било централизиран възел. Хронологичен обзор на паралелните генетични алгоритми е предложен в [MICE2006]. Авторите предлагат паралелна имплементация на самоорганизиращ се миграционен алгоритъм който е сравнен с диференциална еволюция. При избора на конкретен програмен език за реализиране на разпределени генетични алгоритми една от възможностите е представена в [SIHA2006] и се базира на Mono-C#. Модел за прогнозиране на натоварването в клъстер предвиден за еволюционни пресмятания е предложен в [MDCG2006]. Разработката представлява клиент-сървър архитектура като има една централна популация, а на

отдалечените възли само се пресмята жизнената функция. Прогнозирането на натоварването е нужно за да се подобри разширяемостта на разпределената система. В [JMAM2007] е представена уеб базирана система за пресмятане на еволюционни алгоритми в разпределена среда. Разработката е базирана на Ruby on Rails и адресира възможностите за по-голяма разширяемост на системата. В [TJGC2007] се представя стратегия за управление на миграцията в разпределени генетични алгоритми която се базира на променлив брой острови. Броят острови варира с времето. В [GDAM2008] е представено обучение на изкуствени невронни мрежи в разпределена среда на клъстерна система. Прави се пълно копие на изкуствената невронна мрежа във всеки възел на клъстера. Всеки възел тренира мрежата на подмножество от тренировъчните примери за всяка епоха. При всеки рунд на обучение възлите си разменят теглата и обновяват локалните им стойности. Описва се и подход за обучение на изкуствени невронни мрежи с генетични алгоритми. В [KSSL2008] се представя опортюнистична еволюция в разпределена среда. Централният възел изпраща индивиди на подчинените възли. Те пресмятат жизнеността, но не връщат веднага резултата на централния възел, а правят допълнителна локална оптимизация. Всеки подчинен възел пресмята група от индивиди. Глобалните оптимизационни евристики, базирани на популация и/или еволюция са едни от подходящите за реализация в разпределена среда от тип „тока-в-точка“. В [JLAE2008] се представя разпределен еволюционен алгоритъм. За целите на комуникацията се използват newcast и gossiping протоколите в peer-to-peer архитектурата. В [JLPC2008] идеята се разширява и се анализира поведението на системата при включвания и изключвания на възли. Също се ползват newcast и gossiping протоколите за комуникация в разпределената среда. В [JULA2010] също се предлага паралелен еволюционен алгоритъм, който се реализира в peer-to-peer архитектура. В [LAJM2009] се предлага сравнение между политиките за миграция в разпределените еволюционни алгоритми. Предлага се алгоритъм за подбор на мигрантите според това те да са максимално различни спрямо популацията към която биват изпращани. В [SREN2010] се предлага паралелна реализация на диференциална еволюция с използване на островния модел и миграция на елементи между островите. Предложеният модел се прилага за проблема на завършените корелационни матрици. В [KTTI2010a] се предлага паралелна реализация на последователен вариант на диференциална еволюция. Предложението е за реализация на компютърна система с многоядрени процесори. Използва се модел за паралелна обработка наречен MapReduce който в общи линии се базира на принципа "разделяй и владей". С развитието на графичните процесори актуални стават и

възможностите да се реализират глобални оптимизационни алгоритми с използването на графичните ускорители. На тази основа в [RCAD2010] се предлага реализация на диференциална еволюция работеща на GPU. В [SGNB2011] също се предлага GPU реализация на изцяло паралелна диференциална еволюция. В [PKJP2011a] се извършва сравнение между генетични алгоритми и диференциална еволюция реализирани за GPU. Сравнението се извършва със задача за планиране на ресурси. Като резултати диференциалната еволюция предоставя значително по-добри решения за фиксиран интервал от време. В [PKJP2011b] се предлага GPU реализация на диференциална еволюция която се ползва за решаване на задачата за разпределение на ресурси. В [LRCC2011] се разглеждат възможностите за използване GPU реализация на диференциална еволюция за оптимизация на проблеми от био-информатиката. В [DDJG2012] се предлага GPU реализация на диференциална еволюция за оптимизиране на график. В [FFRK2012] е представена GPU реализация на сдвоена диференциална еволюция за решаване на min-max задачи. В [KTTI2010b] се предлага паралелна имплементация на последователен алгоритъм за диференциална еволюция. При последователната реализация на диференциална еволюция новосъздаденият вектор се сравнява с родителя и ако е по-добър веднага го замества в популацията. В [TDDA2010] се прави детайлно сравнение между различни евристични алгоритми приложени в различни системи за паралелни пресмятания. Опитите са направени в многоядрени системи, хомогенни клъстъри и хетерогенна разпределена система в Интернет. Акцентът е върху резултатите при хетерогенната разпределена система в Интернет където някои еволюционни алгоритми не успяват да открият оптимумите на тестовите функции, а друг еволюционни алгоритми откриват оптимуми които в другите реализации не успяват. Една относително малка част от изчисленията в разпределена среда се извършват с използването на дарена изчислителна мощност (donated computing power). В [NCTD2010] се представя пресмятане на еволюционни алгоритми като дарена изчислителна мощност през инфраструктурата на BOINC [SINI2011][HREL2011]. В [TDMM2010] се представя валидиране не резултатите от еволюционни алгоритми в разпределени системи с участници на доброволни начала. На практика еволюционните алгоритми продуцират резултати които дори не валидирани лесно отпадат при слаби жизнени функции на следващи етапи от оптимизацията. Това е възможно тъй като при еволюционните алгоритми голяма част от междините резултати не се съхраняват до достигането на крайно решение. При изчисленията в разпределени среди (особено когато става въпрос за дарена изчислителна мощност) няма

гаранция за надеждността на резултатите получени на изчислителни машини под чужд контрол [TEGE1994]. За алгоритми различни от еволюционните алгоритми този факт представлява сериозен проблем и се разработват различни подходи за борбата с такъв вид компрометиране на изчисленията. Най-често прилаганият способ е едно и също пресмятане да се извършва на няколко независими, отдалечени машини. Това е ефективен подход, но идва с цената на загубена изчислителна мощност. В случая на еволюционните алгоритми, компрометиране на пресмятанията (дори с изцяло злонамерена цел) не само, че не води до проблем в дългосрочен план, но дори може да бъде полезно тъй като би въвело разнообразие в еволюционния процес. В [MWFN2011] се предлага паралелна имплементация на диференциална еволюция. Използва се случайно разбъркване на индивидите в под популациите. Втората предложена модификация е промяна на фактора за преоразмеряване в под популациите. При разбъркването индивидите от отделните популации се смесват и формират нови под популации. Коефициентът за преоразмеряване се променя по случаен принцип в различните популации. В [PEBU2011] се предлага адаптивна паралелна реализация на диференциална еволюция базирана на процеса за миграция. В това изследване топологията звезда дава по-добри резултати от миграцията спрямо кръговата топология. В [JOTT2011] се предлага хибриден паралелен асинхронен метод за глобална оптимизация базиран на симулирано закаляване и елементи от диференциалната еволюция. При асинхронната паралелна реализация всеки работен възел извършва изчисления и докладва резултата без да се интересува на какъв етап от изчисленията са останалите работни възли. В [KTTI2011] се предлага паралелна реализация на диференциална еволюция за решаване на проблеми с висока степен на несигурност. При някои проблеми с висока степен на несигурност се използват Монте-Карло симулации. В [MDRT2011] се предлага паралелна диференциална еволюция за многоцелева оптимизация еднакво добре приложим в хомогенна или хетерогенна паралелна архитектура от изчислителни възли. Базира се на клиент-сървър организация и прилага асинхронна комуникация между възлите в компютърната мрежа [ATHA2004][GAVI2002]. В [EZMT2011] се предлага комбинация на множество еволюционни алгоритми (генетични алгоритми, диференциална еволюция, рояк от частици, еволюционни стратегии) в множество изчислителни възли. Различните възли извършват различен еволюционен алгоритъм което дава богато разнообразие за изследване на пространството от решенията. Различните оптимизационни алгоритми се комбинират с три различни стратегии за периодична миграция. В [MGJM2011] се представят еволюционни изчисления в

разпределена среда базирани на облачна инфраструктура (в случая Dropbox). Отделните възли в изчислението са хетерогенни което означава различни операционни системи. Облачната среда се ползва предимно като хранилище на файлове върху различните възли участващи в изчислението. В [JMMG2011] също се предлага облачно базирана реализация на еволюционни алгоритми в разпределена среда с помощта на Dropbox. В [KITA2012] се предлага сравнителен анализ за паралелна реализация на диференциална еволюция. Изследването е направено на многоядрени машини, като се сравняват две модификации. Моделът с динамично заделяне на задачите се представя по-добре от модела със статично заделяне на задачите. В [DXLD2012] се предлага само адаптираща се паралелна версия на диференциална еволюция. Параметрите на диференциалната еволюция (фактор за преоразмеряване и вероятност за кръстосване) се кодират в самите индивиди от популацията. Популацията се разделя на множество сегменти и се имитира кръгова топология с миграция. В [DLHL2012] се предлага модификация на диференциална еволюция в областта за избор на оператор за мутация. Подобрението е в следствие посоката на отместване както е представено в [PRST2005][FEOK2006][STOR1996][JLAM2002]. Реализират се няколко паралелни популации на диференциална еволюция в които се пазят няколко различни атрактора (най-добри намерени решения). В [ERFO1993] се предлагат разпределени генетични алгоритми за обучение на фронтални изкуствени невронни мрежи.

1.7 Анализ, проблеми, цели и задачи

В резултат на направения обзор могат да се формулират група проблеми които да бъдат заложени като цели на дисертационния труд, а самите цели да се разбият на задачи, решението на които да доведат до постигане на поставените цели.

1.7.1 Анализ

Основен недостатък на разгледаните решения за обучение на изкуствени невронни мрежи с еволюционни алгоритми е фактът, че почти не се срещат реализации на пълно-свързани изкуствени невронни мрежи. Вторият основен недостатък на разгледаните

съществуващи решения е, че масово изкуствени невронни мрежи се използват на две фази - обучение и експлоатация. При проблемите описвани с времеви редове е характерно с напредване на времето времевия ред да нараства и е съвсем естествено изкуствени невронни мрежи да бъде преобучавани спрямо новопостъпилата информация. Използването на точни числени методи (като най-популярният обратно разпространение на грешката [BILL1998]) е ефективно, но относително неприложимо при изкуствени невронни мрежи с пълно-свързана топология. В литературата са представени множество хибридни модели за обучение на изкуствени невронни мрежи с точни числени методи в комбинация с методи за евристична оптимизация. Трети, макар и второстепенен проблем, е правилното избиране на подходящи точни и евристични методи за извършването на хибридно обучение. В проучените информационни източници се забелязва относително слабо застъпване на възможностите изкуствени невронни мрежи да се обучават в условията на разпределена среда, така че максимално да се ползват възможностите за паралелна реализация на оптимизационните алгоритми от групата на еволюционните алгоритми и популационните алгоритми.

Приложението на изкуствени невронни мрежи при прогнозирането на времеви редове в икономиката се разглежда от различни автори, някои от които са Dunis [DUWI2002], Giles [GILA2001] и Moody [MOOD1995]. Като основен модел се използват така наречените Feed Forward Neural Networks (виж Haykin [HAYK1999]). Мрежите от тип FFNN са много ефективни, но страдат от основен недостатък свързан с липсата на кратковременна памет. Избягването на този проблем се осъществява с използването на така наречените Recurrent Neural Networks. Основната трудност при RNN идва от невъзможността за тяхното обучение да се използват точни градиентни методи (виж Werbos [WERB1990]). Комбиниран подход за обучение на изкуствени невронни мрежи с помощта на еволюционни алгоритми е предложен от Yao [YAOX1999] и други автори. Еволюционните алгоритми дават значително по-добри резултати за търсене на оптимуми в сложни многомерни пространства с наличие на множество локални оптимуми, в които градиентните методи не дават добри резултати (виж Holland [HOLL1975]).

1.7.2 Проблеми

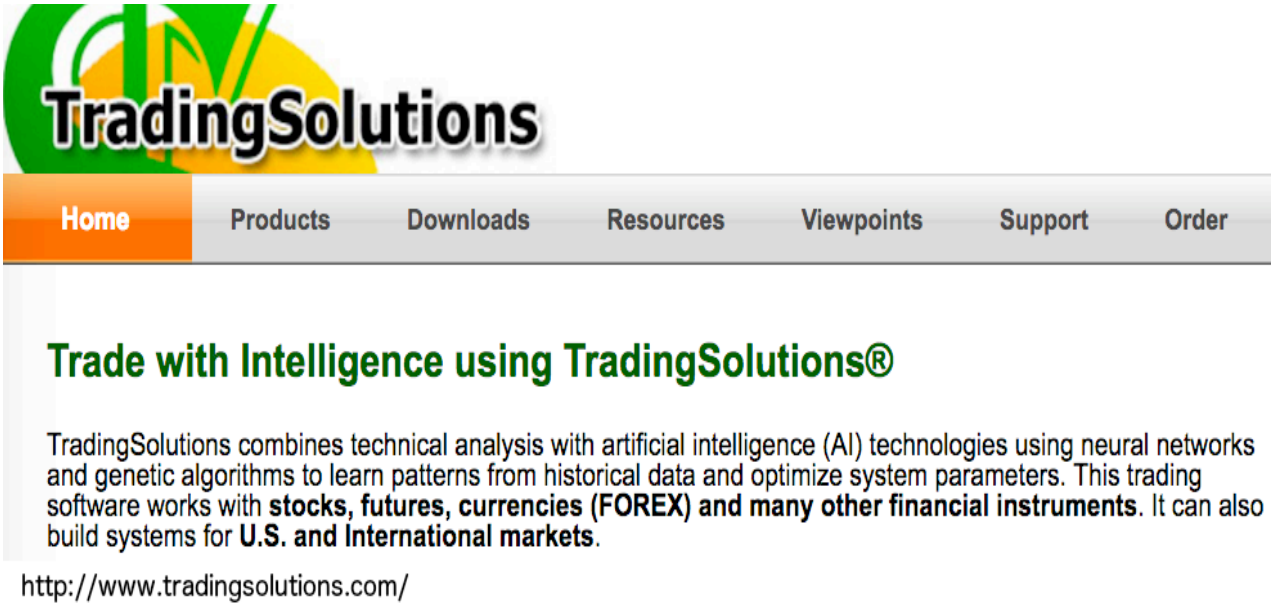
При търговията с финансови инструменти са важни три основни параметъра:

1. В коя посока ще се промени цената (повишение или намаление);
2. Колко време ще продължи това нарастване или намаляване;
3. Какъв да бъде обемът на дългата или късата поръчка;

Тъй като търговията носи своите рискове посредниците/брокерите се налага да правят ежедневни анализи на световната икономическа обстановка. В общия случай на отговорно управление те инвестират средства на свои клиенти (физически или юридически лица). При подбирането на инструменти в които да се реализират инвестициите посредниците най-вече се интересуват от посоката в която ще се промени цената. Търгуващите на глобалния пазар условно се разделят на две групи „мечки“ и „бикове“. Метафората за названието на тези две групи идва от факта, че мечката напада противника си опитвайки се да го захлупи с лапите си, а бикът напада противника си опитвайки се да го промуши с рогата си отдолу нагоре. Двете групи търгуващи са в постоянно противопоставяне и по този начин влияят на пазара, така че той да повишава цените си или да намалява цените си, в зависимост от това коя от двете групи надделява в конкретния момент [GENE2000]. Вторият съществен параметър е колко дълго ще продължи едно повишение или намаляване на цената. След като посредникът е отворил своята позиция и успешно е предвидил посоката на промяна става актуален въпросът колко дълго тази позиция да бъде отворена. Отговорът на този въпрос е пряко свързан с размера на печалбата която конкретната позиция ще реализира. Ако позицията бъде затворена твърде рано то тя ще има нереализирана печалба. Ако позицията бъде затворена твърде късно е възможно акумулирана печалба да се изгуби. Най-оптималният момент за затваряне на една отворена позиция е когато движението на цената от възходящо стане низходящо или обратното. Последният, трети параметър, е свързан с размера на поемания риск. Опита на посредниците бързо ги учи, че наличните активи не се инвестират само в един финансов инструмент, а в цяла група инструменти наречена „инвестиционен портфейл“. Желанието на всеки търгуващ е да инвестира колкото се може по-голяма част от активите в печеливши позиции. Това което ограничава посредниците да инвестират всичко в един конкретен инструмент и една конкретна позиция е рискът да не разпознаят правилно посоката в която ще се промени цената и продължителността на съответното увеличение или намаление.

Ако бъде направено погрешно решение за инвестиране (посока, продължителност, обем) могат да последват значителни загуби за съответната позиция. Проблемите, които търгуващите на свободните пазари срещат са свързани точно с прогнозиране на посоката и продължителността в промяната на цените. Решението за търгувания обем е задача от областта управление на финансов портфейл.

В подкрепа на идеите за разработката на дисертационния труд са две частни търговски инициативи. На първо място е пакетът от решения TradingSolutions на фирмата NeuroDimension Inc.



Фиг. 1.1 *TradingSolutions на фирмата NeuroDimension Inc.*

Серията продукти TradingSolutions са насочени към прогнозиране на валути и акции (Фиг. 1.1). В основата на тези продукти са заложени алгоритми от областта на изкуствения интелект. Основно се залага на изкуствени невронни мрежи и генетични алгоритми.

https://en.wikipedia.org/wiki/MoneyBee

Фиг. 1.2 *MoneyBee на фирмата i42 GmbH.*

На второ място е системата MoneyBee. Системата е разработена от фирмата i42 GmbH. Системата MoneyBee основно се състои от screen saver (Фиг. 1.2). Когато screen saver-ът бъде активиран върху компютърната система на потребителя той извършва обучение на изкуствени невронни мрежи, така че да прогнозират финансови времеви редове. Самият screen saver разчита на дарена от потребителите изчислителна мощност. Резултатите от извършените изчисления се предават на централизиран сървър.

1.7.3 Цели

1. Да се изследват и предложат подобрени модели за обучение на пълно-свързани изкуствени невронни мрежи.

2. Да се изследват и предложат подобрени алгоритми за непрекъснато обучение, дори в процеса на експлоатация на изкуствените невронни мрежи.

3. Да се изследват и предложат подобрени хибридни комбинаци от представените в литературата точни и евристични алгоритми за обучение на изкуствени невронни мрежи.

1.7.4 Задачи

1. Реализацията на софтуер за обучение на изкуствени невронни мрежи в разпределена изчислителна среда. Обучението на мрежите да се осъществи във вид на C++ и JavaScript програма.

а) Реализация на клиентско приложение.

б) Реализация на сървър приложение.

2. Реализация на непрекъснато обучение на предложения пълно-свързан модел на изкуствени невронни мрежи. Обучението трябва да протича в непрекъснат 24/7 цикъл

а) Реализация на алгоритми за непрекъснато разпределяне на задачи от сървъра към клиентските приложения.

б) Реализация на непрекъснато обучение за локалните копия на изкуствените невронни мрежи които клиентските машини поддържат.

3. Реализиране на комбинация от алгоритми като генетични алгоритми и диференциална еволюция с алгоритми като обратно разпространение на грешката, така че времетраенето на обучението на изкуствени невронни мрежи да бъде намалено.

4. Техническата реализация на системата за изчисления в разпределена среда и провеждането на необходимите експерименти за потвърждаване на ефективността с която тя работи.

Глава 2 - Модел за прогнозиране на времеви редове с изкуствени невронни мрежи, обучавани с еволюционни алгоритми

Настоящото изложение представя модел за прогнозиране на времеви редове, базиран на изкуствени невронни мрежи, обучавани с алгоритъм за диференциална еволюция, в разпределена среда. Сложността на задачата за прогнозиране на времеви редове изисква разработването на по-бързо действащи алгоритми, които водят до прогнози с по-голяма точност. В основата на изкуствените невронни мрежи за прогнозиране е заложена идеята за обучение, на база последователност от данни за изминал период. Изборът на обучаващ алгоритъм може да бъде направен от две основни направления - точни числени методи и евристични оптимизационни метод. Прилагането на подходящи евристични алгоритми може да доведе до използването им в разпределена среда и значително ускоряване на процеса за обучение.

2.1 Определяне на потребителските потребности

Прогнозирането на времеви редове е задача, в която по зададени хронологични данни, трябва да се направи прогноза за данни в предстоящ период от времето. Когато става въпрос за финансови инструменти дял от науката, наречена „Иконометрия”, се занимава с разработка на алгоритми, финансови индикатори и осцилатори, с помощта на които да се анализират данните. Създаването на ефективни и надеждни средства за финансово прогнозиране е трудоемка и сложна задача. Поради тази причина много актуална е областта за разработка на самоорганизиращи се или самообучаващи се системи за прогнозиране.

В настоящото изложение се представя модел на самообучаваща се система за прогнозиране движението на валутните пазари. Последователно се разглеждат концепциите за времеви редове, изкуствени невронни мрежи, диференциална еволюция, предложение за

модел и проблеми, които да бъдат изследвани.

2.2 Времеви редове

Времевите редове представляват последователност от данни, събрани на равни или неравни интервали от време. В общия случай на всяка стойност от времето може да се съпостави стойност на определена величина:

(1) $$y = f(t)$$

Основна характеристика на времевите редове е, че всяка следваща стойност е в зависимост от предходните стойности. Тази зависимост може да описва лесно предсказуем процес, но и чисто хаотичен процес. Пример за времеви ред е средната дневна температура в рамките на една година. В дисертационния труд се разглеждат времеви редове на валутните пазари. Цената на отделните валути варира във всеки един момент от времето, под влиянието на купувачите и продавачите на пазара.

2.3 Изкуствени невронни мрежи

В класическият вариант изкуствени невронни мрежи представляват насочен тегловен граф. Всеки възел в графа реализира изчислителен процес, по аналогия на естествените невронни системи. Теглата на връзките, между отделните невронни възли, определят нивото на влияние по между им. Спрямо мрежата първа група невронни възли се използват за получаване на входна информация, втора група невронни възли се използват за представяне на изход, а трета група невронни възли обработват информацията в скритите слоеве. Целта на всяка една изкуствена невронна мрежа е да изгради функционална зависимост между входните и изходните данни.

Настоящото изложение разглежда класически изкуствени невронни мрежи, които се

използват с алгоритъм за обратно разпространение на грешката. Структурата по която са свързват отделните невронни възли определя топологията на изкуствените невронни мрежи. При мрежите с обратно разпространение на грешката най-разпространената топология е с три слоя. В този вариант връзки има само между два съседни слоя. Алгоритъмът за обучение с обратно разпространение на грешката по своята същност е точен числен, градиентен метод който се стреми да минимизира грешката допусната в изходния слой на мрежата. До момента алгоритъмът с обратно разпространение на грешката дава най-добра скорост за обучение, в процеса на обучение, но притежава няколко много сериозни недостатъка: 1) Подходът за корекция на теглата може да доведе до често попадане в локален екстремум; 2) Корекцията на теглата може да доведе до преобучение на мрежата (състояние на катастрофално забравяне); 3) Трудно се обучават изкуствени невронни мрежи с обратни връзки, тъй като не може да се пресметне грешката в изходния слой и след това грешката да се разпространи обратно по слоевете. При прогнозирането на времеви редове с изкуствени невронни мрежи е съществено да има елемент на кратковременна памет, която се постига с обратни връзки в мрежата.

Идеята за прогнозиране на времеви редове с изкуствени невронни мрежи се състои в подаване на последователност от данни за отминал период от време на входа на изкуствената невронна мрежа и получаване на прогноза която мрежата предлага в изхода.

2.4 Диференциална еволюция

Диференциалната еволюция представлява популационен евристичен оптимизационен алгоритъм с общо предназначение. Разработен е на основата на генетичните алгоритми, като основните разлики се изразяват в операцията по мутация. В алгоритъма първоначално се прилага класическо кръстосване. Кръстосването е познато в генетичните алгоритми като то може да бъде в една точка или многоточково. За мутацията се използва претеглен диференциален вектор вместо единична промяна на някой от гените във вече кръстосаната хромозома. Претегленият диференциален вектор се изчислява от две случайно избрани хромозоми чрез изваждане на единия вектор от другия и умножаването му с тегловен

коефициент. Същественото подобрение при диференциалната еволюция, спрямо генетичните алгоритми се състои в това, че мутацията води до малка стъпка в околността на точката представяща решението като стъпката се прави във всички посоки на N-мерното пространство. При генетичните алгоритми стъпка в околността на точката представяща решението се прави само по една от координатите на N-мерното пространство.

В настоящото изложение диференциалната еволюция е избрана с цел подобряване сходимостта на процеса за обучение и заради идеалните възможности на алгоритъма да бъде реализиран в разпределена среда. Всяка хромозома в диференциалната еволюция кодира един комплект тегла за изкуствената невронна мрежа. Оценката за жизнеността на хромозомите се определя според успеваемостта за прогнозиране на подаваните обучаващи примери. Възможността да се изпълняват отделни копия на диференциалната еволюция на различни изчислителни машини позволява пространството на решенията да бъде по-подробно изследвано. Получава се така, че всяко локално копие работи в определен регион на това пространство.

2.5 Изчисления в разпределена среда

При изчисленията в разпределена среда се изпълняват паралелни алгоритми, като за разлика от паралелното програмиране, пресмятанията се извършват на различни изчислителни машини. Разликите между разпределените системи и многопроцесорните системи е изключително малка и не е толкова концептуална, а чисто техническа. Многопроцесорните системи са под контрола на институцията, която ги притежава, докато разпределените системи се изпълняват на хардуер който е под контрола на различни хора или институции. Този факт води до първия проблем свързан с надеждността на извършените изчисления и с възможностите за злонамерено манипулиране на получените резултати. Втората основна разлика е латентността на компютърната мрежа която води до забавяне при предаването на задачи за изчисление и връщане на резултатите. Като част от този проблем е и ситуацията в която въобще не се връща резултат за възложено изчисление. При многопроцесорните системи този проблем не съществува, тъй като комуникацията е в

рамките на системната шина или в рамките на високо скоростна и надеждна вътрешна компютърна мрежа. Поради посочените проблеми проектирането на системи за изчисления в разпределена среда е по-сложно отколкото създаването на паралелни програми за многопроцесорни системи. Тази допълнителна сложност най-често се отразява в крайната производствена цена на съответния софтуер.

Разпределените изчисления намират съществено приложение при повишаване на ефективността от пресмятанията. Основно предимство на този вид изчисления пред използването на супер компютър е драстичното понижаване на разходите за хардуер, тоест на цената на масово достъпна компютърна система (служеща за сървър) може да се получи изчислителната мощ на супер компютър от по-нисък клас. Второто предимство е възможността чрез локалните копия да се изследват различни части от пространството на решенията. Същият ефект може да се постигне и с многопроцесорна система, но на по-висока цена. Използването на евристичен оптимизационен алгоритъм води до извършването на множество изчисления повечето от които нямат директно приложение, а служат само за получаване на междинни резултати. Наличието на междинни пресмятания води до идеята за тяхната реализация да се използват съществуващите изчислителни ресурси при клиентите. Обичайно клиентските компютърни системи работят неефективно и в голяма част от времето не се натоварват с полезни изчисления. Ако пресмятанията се извършват на супер компютър, то изчислителните ресурси също така не се използват ефективно, тъй като голяма част от междинните пресмятания не се използват пряко в крайните резултати.

2.6 Предложение за прогнозираща разпределена изчислителна система

В дисертационния труд е предложено за целите на прогнозирането да се използва математически модел базиран на изкуствени невронни мрежи, чието обучение се извършва с диференциална еволюция, под формата на изчисления в разпределена среда. Топологията на изкуствените невронни мрежи представлява изследователски интерес и за това в софтуерната система топологията се определя от оператор на отдалечения сървър. Допустимите видове

топологии са: 1) Многослойни; 2) Многослойни с обратна връзка; 3) Пълно-свързани. Предложеният модел е базиран на класическа мрежа, каквито се използват в моделите с обратно разпространение на грешката. Като сумираща функция е прието да се използва линейна функция:

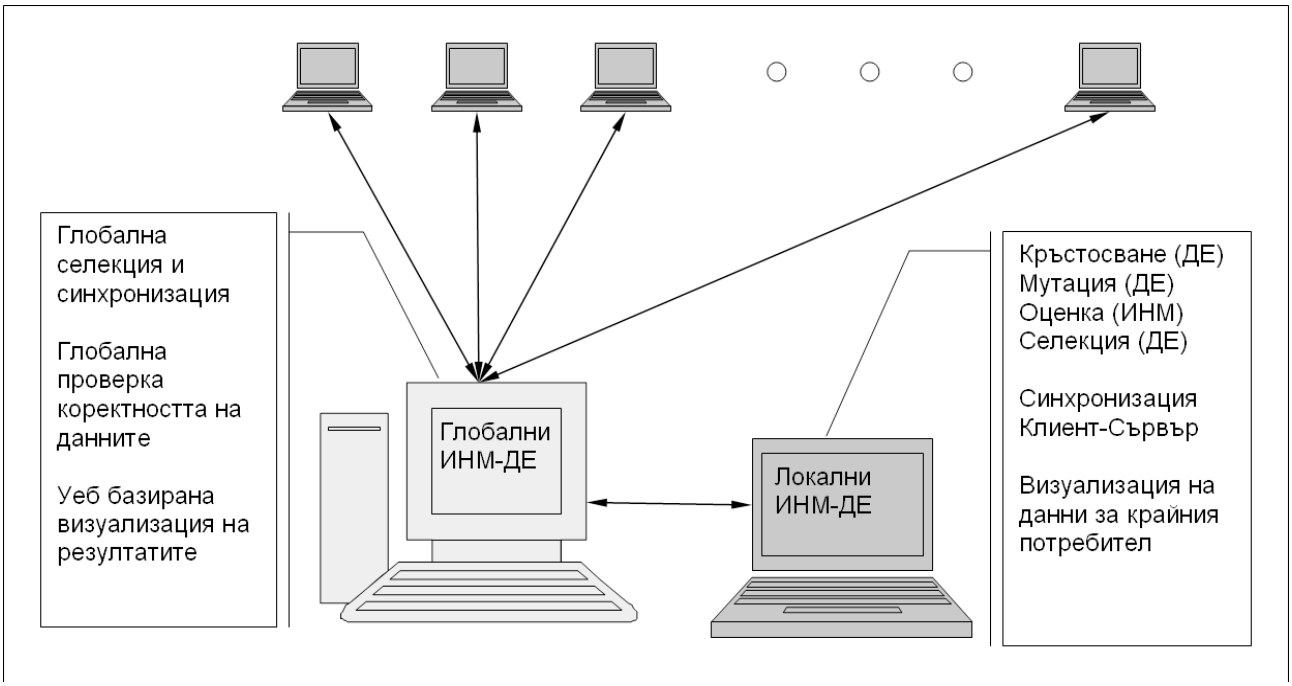
(2) $$u_i = \sum w_{ij} * x_j$$

Където i обозначава индексът на конкретния неврон, а j индексите на невроните с които е свързан. Сумиращата функция определя начина, по който входните сигнали, в комбинация с тегловните коефициенти, ще влияят върху активността на съответния неврон. Известни са и други видове сумиращи функции, но за сега изследванията ще са насочени само към линейна функция тъй като тя е най-често използвана от други автори в областта.

Резултатът от сумиращата функция е необходимо да се нормира с помощта на нормираща функция (прагова функция). В избрания модел е предпочетена сигмоидна функция със стойности в диапазона от 0.0 до 1.0, тъй като различният брой връзки при различните неврони биха повлияли различно, ако не се нормира сумата.

(3) $$x_i = \frac{1}{1 + e^{-u_i}}$$

Сигмоидната функция е избрана за нормираща функция поради нейните свойства по отношение на диференцируемост и асимптотичност. Възможно е използването на двоична функция или линейна функция (с цел подобряване на бързодействието), но техните свойства влошават резултатите (примерно линейна функция). Ако мрежата работи със стойности от -1 до +1, то като нормираща функция може да се използва хиперболичен тангенс.



Фиг. 2.1 *Система за обучение на изкуствени невронни мрежи с диференциална еволюция, в разпределена среда.*

Обучението на изкуствени невронни мрежи се осъществява с помощта на диференциална еволюция. В рамките на алгоритъма диференциална еволюция всяка хромозома представя един комплект тегла за конкретна топология изкуствени невронни мрежи към която се отнасят определен брой обучаващи примери. Последователно отделните хромозоми (комплекти тегла) се зареждат в структурата на изкуствената невронна мрежа след което се подават обучаващите примери. Следва изчисляване на грешката която изкуствената невронна мрежа допуска за всеки пример. След това се сумират грешките от отделните примери и се определя коефициент за жизнеспособност на хромозомата (комплект тегла). Начинът по който се определя коефициентът на жизнеспособност е един от най-ключовите стъпки от които зависи успехът на цялостното прогнозиране. При времевите редове е необходимо обучаващите примери да се подават в хронологична последователност което създава затруднения при изкуствени невронни мрежи обучавани с алгоритъм за обратно разпространение на грешката. Също така, отдалечените във времето обучаващи примери е необходимо да оказват по-малко влияние при формирането на коефициента за жизнеспособност. След като бъдат оценени отделните хромозоми те влизат в изчислителната

схема на диференциалната еволюция като над тях се извършва селекция, кръстосване и мутация.

При паралелния вариант на алгоритъма за диференциална еволюция се създават локални копия на изкуствената невронна мрежа и на диференциалната еволюция [BAZA2010]. Обучението протича локално като всяка хромозома в популацията обозначава отделна точка (индивид) в пространството на търсене. Макар и различни индивидите се групират относително близко един до друг в това пространство. При разделяне на изчисленията върху множество изчислителни машини подобно групиране на индивиди може да доведе до локално изследване на различни области от пространството на решенията (пространството на търсене) [BATO2011]. В този аспект най-съществена е политиката за синхронизиране като процеса на синхронизиране включва излъчването на най-добрите индивиди и събирането им на едно централизирано място (клиент-сървър архитектура). Това множество на глобално най-устойчивите индивиди може да се използва при създаването на следващи локални ареали. Освен клиент-сървър архитектура е възможно да се реализира peer-to-peer решение, при което отсъства централизиран сървър, а всяко локално работещо приложение се грижи за комуникацията с други локално работещи приложения. Основно предимство на предложената клиент-сървър система за изчисления в разпределена среда е нейната изключително висока степен на скалируемост (изключително улеснено включване на допълнителни компютърни системи в системата за разпределени изчисления) [BAZA2011]. Честотата на комуникация между клиентите и сървъра може да се регулира прецизно, като за интервали може да се ползват часове, седмица и дори месеци. При наличие на голям брой клиенти, натоварването на сървъра може да се компенсира с намаляване честотата на комуникация с клиентите.

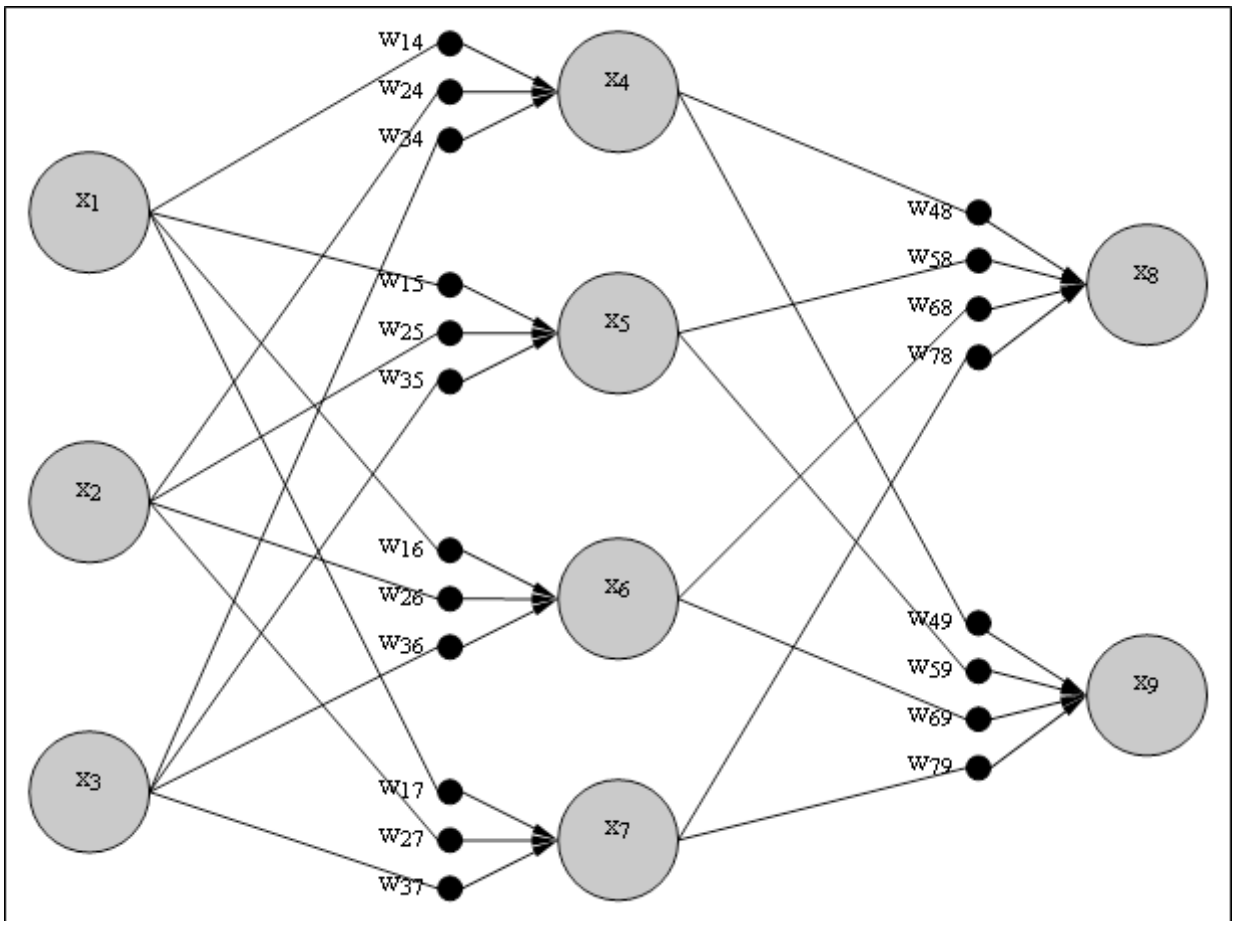
Първото предимство на предложения модел се състои в използването на диференциална еволюция за обучението на изкуствени невронни мрежи като по този начин се избяга опасността от преобучение [BATO2015]. Второто предимство е възможността чрез диференциална еволюция да се обучават изкуствени невронни мрежи с обратни връзки. Третото предимство е възможността чрез диференциална еволюция да се обучават изкуствени невронни мрежи без да е от значение редът по който се подават обучаващите

примери. Четвъртото предимство е възможността да се обучават различни копия на изкуствени невронни мрежи и това да става паралелно водещо до подобряване на бързодействието и по-добро покритие на пространството за търсене. При евентуални злонамерени клиенти които изпращат некоректна информация към сървъра (комплекти тегла), то тази информация може да се провери лесно. От друга страна некоректната информация получена от злонамерени клиенти може да се използва в неподправен вид за обогатяване генетичния фонд на сървъра. Минимална бройка злонамерените клиенти не представляват проблем за предложения модел дори ако не се вземат специални мерки за борба с тях. При постъпване на злонамерено увредени данни в глобалната популация те биват валидирани още при първото оценяване в локалните популации и биха изпаднали от списъка на индивидите в популацията на диференциялната еволюция.

Освен предимства предложеният модел има и следните недостатъци: Използването на изкуствени невронни мрежи е свързано с бавно обучение. Макар и много ефективни изкуствените невронни мрежи изискват големи количества изчислителни ресурси основно във фазата на обучение. Използването на диференциална еволюция значително забавя процеса на обучение в сравнение с алгоритъма за обратно разпространение на грешката, но поради множеството изброени предимства диференциална еволюция е предпочитан пред обратно разпространение на грешката. Интересен би бил модел в който диференциална еволюция се комбинира с обратно разпространение на грешката макар и това да не може да се постигне без някакъв компромис по отношение топологията на изкуствените невронни мрежи. Разработването на програми за изчисления в разпределена среда е по-сложно от писането на строго последователни или паралелни програми.

2.6.1 Структура на изкуствена невронна мрежа

Изкуствената невронна мрежа се състоят от следните основни компоненти - неврони за входна информация, неврони в скритите слоеве, неврони за изходна информация и тегла на връзките между отделните неврони. За избор на структура на изкуствени невронни мрежи се разглежда мрежа с 3 входа, 4 възела в скрития слой и 2 изхода.



Фиг. 2.2 *Изкуствена невронна мрежа с топология 3-4-2.*

Всеки неврон е обозначен със своята изходна стойност от x1 до x9, а теглата на връзките между невроните със стойност wij, където i е индекс на източник, а j е индекс на получател. В конкретния пример, входната информация се записва в x1, x2 и x3, а изходната информация се получава в x8 и x9. Стойностите на невроните могат да се представят като вектор:

(4) $$\vec{x} \equiv \left[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 \right]$$

Където векторът *x* е с размер 9, тъй като на Фиг. 2.2 е представена мрежа с 9 възела. В матрична форма теглата на изкуствената невронна мрежа могат да се представят с квадратна матрица която има размер n*n (където n е броят неврони в мрежата):

(5)
$$\overline{w} = \begin{bmatrix} 0 & 0 & 0 & w_{14} & w_{15} & w_{16} & w_{17} & 0 & 0 \\ 0 & 0 & 0 & w_{24} & w_{25} & w_{26} & w_{27} & 0 & 0 \\ 0 & 0 & 0 & w_{34} & w_{35} & w_{36} & w_{37} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{48} & w_{49} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{58} & w_{59} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{68} & w_{69} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{78} & w_{79} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Матрицата *w* е с размери 9x9, тъй като отразява силата на връзки между възлите, в мрежата представени на Фиг. 2.2 (матрица на съседство от теория на графите). Силата на връзките между невронните се регулира с матрица на свързаност като тази матрица определя и топологията на изкуствената невронна мрежа (тоест кой възел с кои други възли е свързан):

(6)
$$\overline{a} = \begin{bmatrix} 0 & 0 & 0 & 1.0 & 1.0 & 1.0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 1.0 & 1.0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 1.0 & 1.0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Матрицата *a* също има размери 9x9. С помощта на матрицата за свързаност може да се регулира кои връзки между невроните ще вземат участие във формирането на изхода на изкуствената невронна мрежа. Чрез подходящо подбиране на активности на теглата могат да се зададат топологии само с връзки напред, с връзки назад или с връзки на всеки възел към всеки друг.

Матриците (5) и (6) пряко отразяват връзките в мрежата представена на Фиг. 2.2.

Наличието на редове и стълбове с нули в тези две матрици се дължи на факта, че според фигурата не съществува връзка между съответните неврони. Ако онагледим една изкуствена невронна мрежа с водоснабдителната система на един град то матрицата (6) онагледява максималният дебит вода който може да премине през тръбите, а матрицата (5) онагледява за какъв моментен дебит са настроени тръбите.

2.6.2 Изчисления от входа към изхода

Пресмятането на стойностите за невроните, при разпространението на входните сигнали в права посока, може да се представи по следния начин:

(7) $$\vec{y} = \vec{x}.\overline{w}$$

Умножението на вектора x с матрицата w представлява сумирането на претеглените входни сигнали. Преди да бъде извършено умножението в матрицата w към нулса се отвеждат елементите според зададените нули в матрицата a.

(8) $$z_i = \frac{1}{1+e^{-y_i}}$$

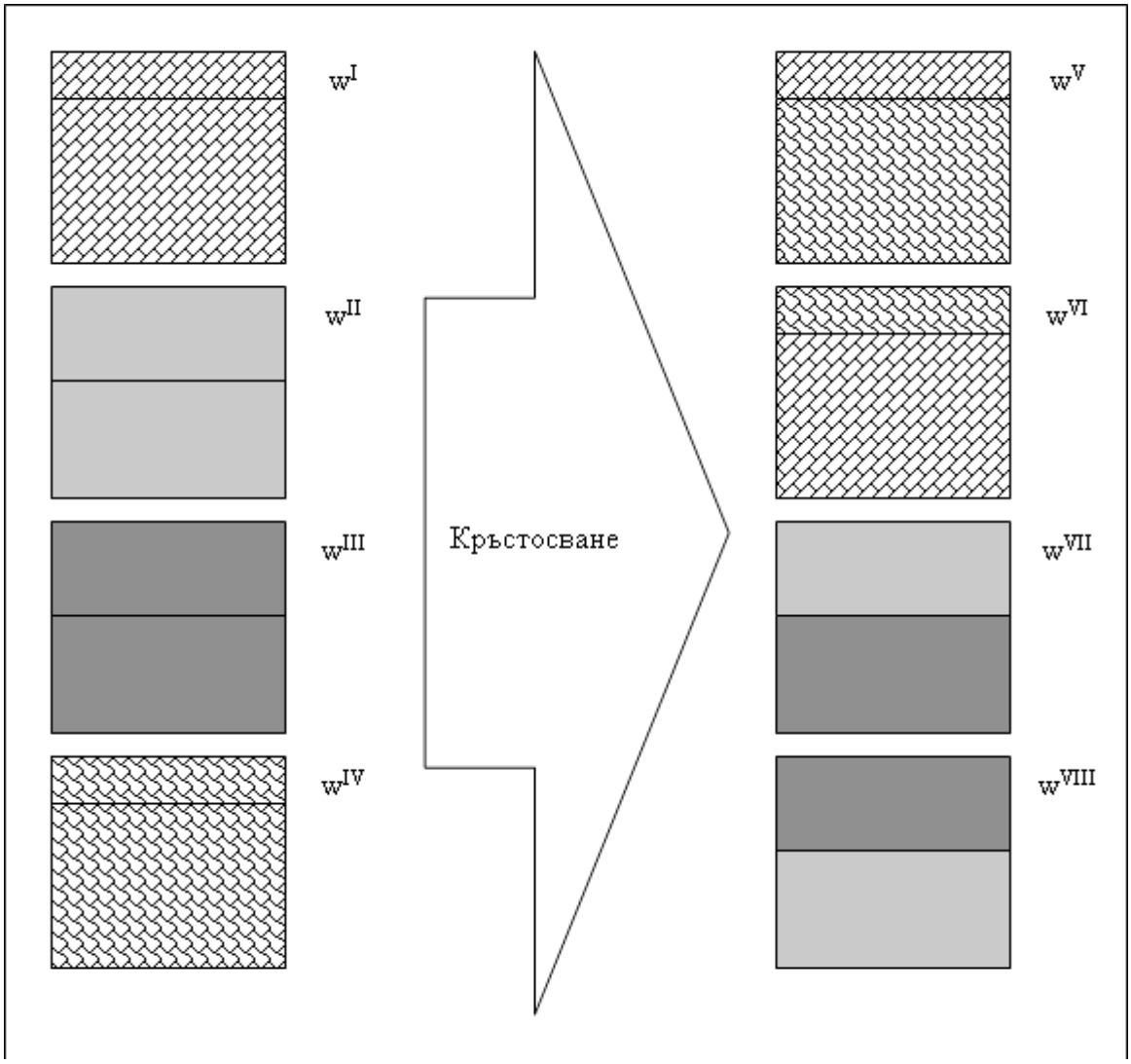
След сумирането на претеглените входни сигнали следва те да се нормират, така че да бъдат в желания диапазон от (0.0-1.0). Интервалът е отворен, тъй като сигмоидната функция има асимптотична сходимост към нула и единициа в двата си края.

(9) $$\vec{x} = \vec{z}$$

Преди да се извърши пресмятането в елементите x1, x2 и x3 се записват стойностите на входните сигнали, които трябва да са в диапазона (0.0-1.0). Резултатът от работата на мрежата се получава в елементите x8 и x9 като също са в диапазона (0.0-1.0).

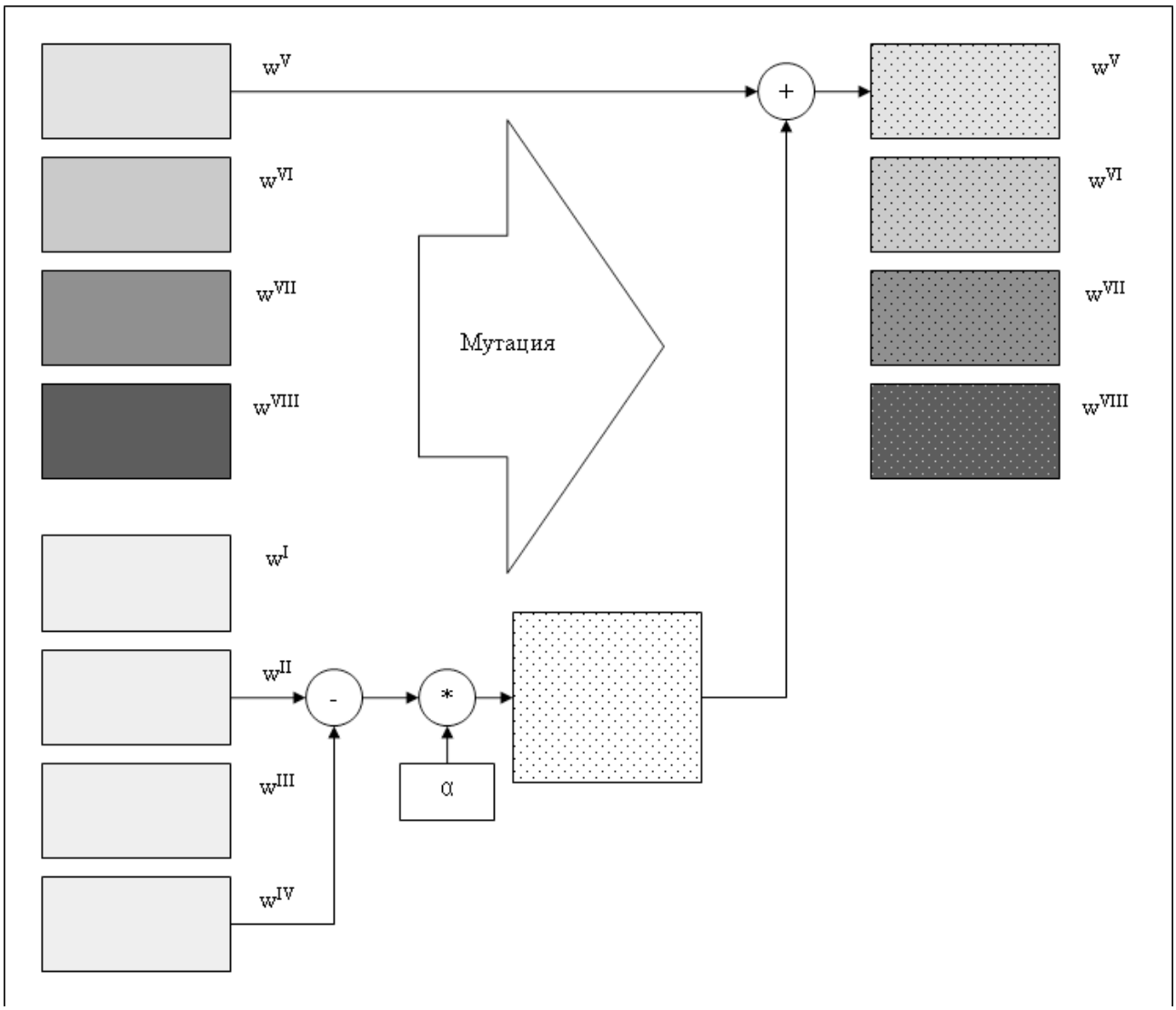
2.6.3 Манипулация на теглата в изкуствените невронни мрежи с диференциална еволюция

Целта при обучението на изкуствени невронни мрежи е да се намери подходящ комплект тегла, така че мрежата да наподобява функцията, която изразява движението на цената, спрямо времето. Когато се използва диференциална еволюция, за разлика от обратно разпространение на грешката, в процеса на обучение няма обратен пас за разпространение на информация в мрежата. По своята същност обучението с диференциална еволюция представлява изпробване на относително случайно комбинирани комплекти тегла (индивиди в термините на генетичните алгоритми). На база на резултатите които постигат отделните комплекти се прилага стратегия за отсяване на по-добре представилите се тегла от по-зле представилите се. След отсяването по-добре представилите се комплекти се използват за рекомбинация и получаване на нови комплекти. Броят на комплектите тегла които участват в изчислението се нарича размер на популацията. Всяка популация се състои от индивиди в настоящото поколение (генерация) и индивиди в новото поколение.



Фиг. 2.3 *Кръстосване на комплекти тегла.*

При кръстосването популацията условно се разделя на стара генерация (I, II, III и IV) и нова генерация (V, VI, VII и VIII). За да се формира новата генерация се избират два индивида (комплекти тегла) по случаен принцип (в примера I и IV, като първа група, а II и III, като втора група). На случайно избраните родителски индивиди се избира случайна точка на срязване (точка на кръстосване). От двата родителски индивида се получават два нови индивида (V и VI, от първа група, а VII и VIII, от втора група). Освен една точка на кръстосване е възможно да се използват две или повече точки, както и равномерно кръстосване. Операцията по кръстосване може да има и по-сложен вид, като дори може да се използват случайно избрани елементи от двамата родители, без те да са последователни.



Фиг. 2.4 *Мутация на комплекти тегла.*

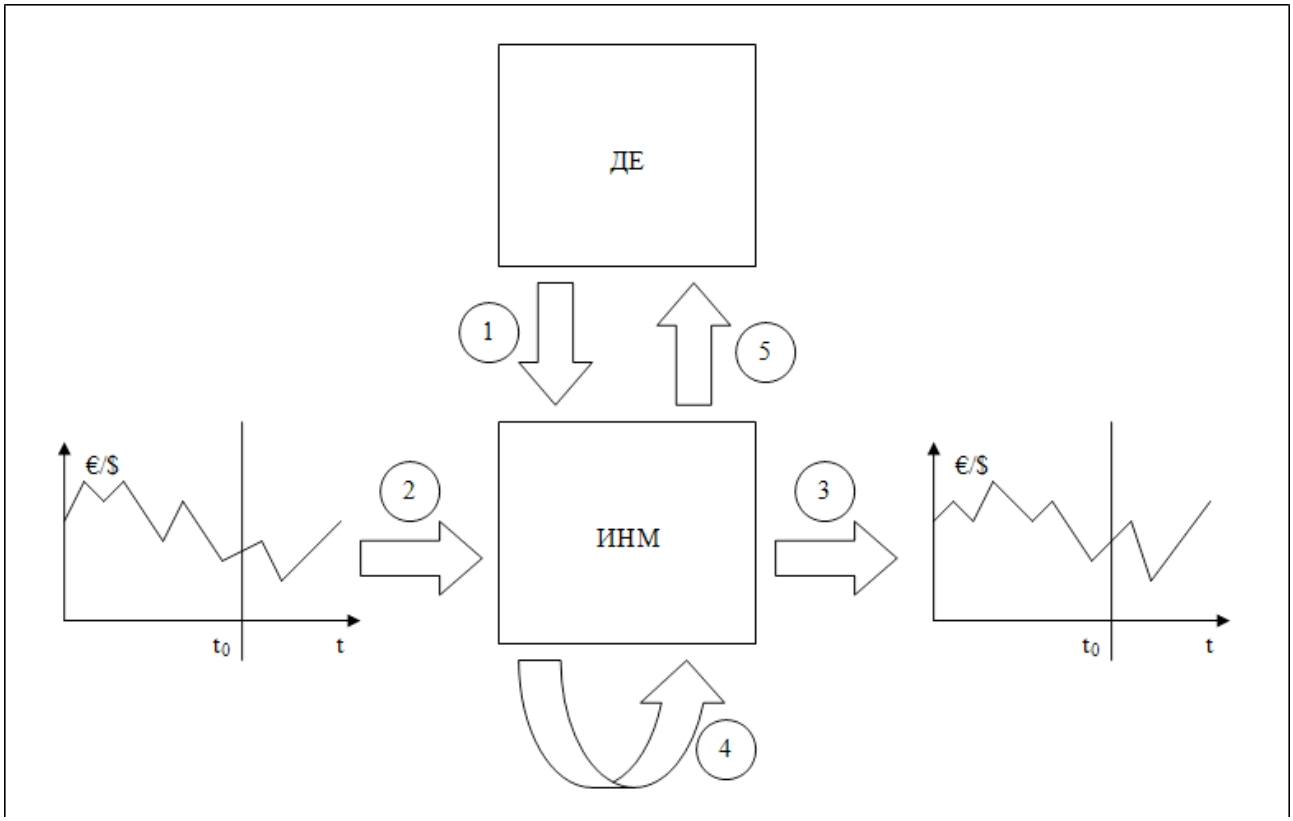
За целите на мутацията се формира претеглен диференциален вектор, който след това се събира с предварително кръстосания индивид. Диференциалният вектор се получава като разлика от два индивида, а след това се претегля, като се умножи с число α в диапазона [0.0-1.0]. Изборът на α се определя експериментално, като използването на динамична промяна може да подобри сходимостта на алгоритъма.

2.6.4 Представяне на входните и изходните данни

Възможни са различни варианти за представяне на входната и изходната информация. В случая изборът е входните данни да се мащабират спрямо най-голямата и най-малката стойност във времевия ред. Множеството от стойности във времевия ред се разделя условно на две под множества – прозорец от минали стойности и прозорец от стойности в бъдещето (lag frame и lead frame).

От начина, по който се представят входните и изходните данни зависи какъв брой неврони ще участват във входния и изходния слой. Намирането на подходящ баланс за броя входни и изходни възли е от съществено значение за броя на връзките в изкуствените невронни мрежи, респективно на теглата които се подлагат на оптимизация.

2.6.5 Оценка на индивидите в популацията



Фиг. 2.5 *Процедура за оценка на комплектите тегла.*

За да се оценят комплектите тегла, то всеки комплект трябва да се зареди в мрежата (стъпка 1). При заредена мрежа многократно се подават входните данни (стъпка 2). Многократно сигналът от входа се предава към изхода (стъпка 3). След което, за всеки пример се оценява до колко добре мрежата прогнозира изходните данни (стъпка 4). Като оценка за грешката на конкретна двойка входно-изходни данни k се използва разстояние между два вектора (прогноза и реална стойност) в евклидово пространство:

(10) $$e_k = \sqrt{\sum (y_i - o_i)^2}$$

Данните условно се разделят на данни от минал период и данни от бъдещ период. Разделението е условно защото по своята същност данните са от вече отминал период.

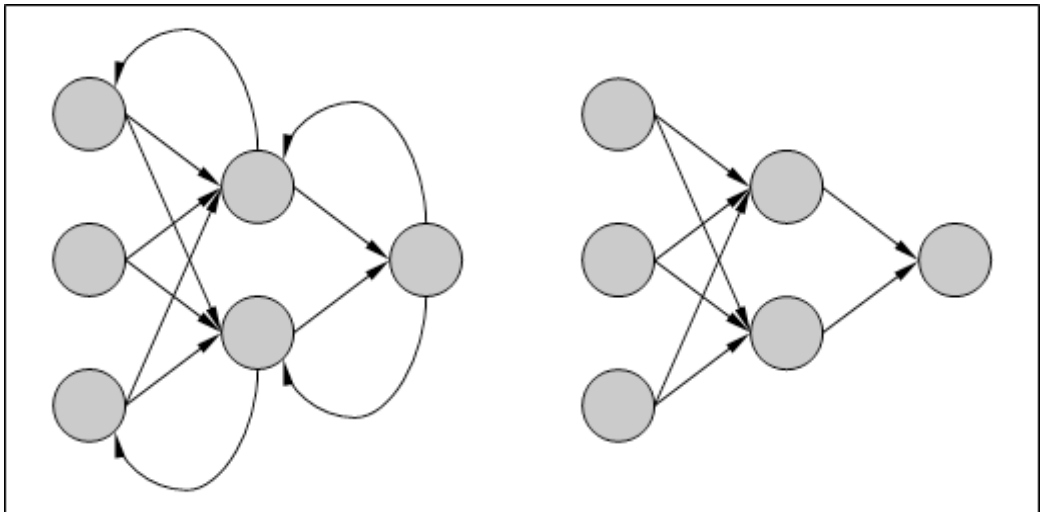
Цялостната грешка, която изкуствената невронна мрежа допуска със заредения комплект тегла се получава като средно аритметично от грешките за всяка двойка входно изходни данни (n на брой):

(11) $$e = \frac{\sum e_k}{n}$$

Сумарната грешка на комплекта тегла се разделя на броя входно-изходни примери за да може да се сравняват комплекти тегла оценени при различен брой входно-изходни примери. Общата грешка на изкуствената невронна мрежа се предава еднократно като жизнена стойност за конкретния комплект тегла в диференциалната еволюция (стъпка 5).

Тъй като обучението в представения модел е непрекъснато, то броят входно-изходни данни постоянно нараства. Възможно е да се въведат тегловни коефициенти на входно-изходните примери, така че данните от по-старите периоди време да оказват по-малко влияние за формирането на общата грешка която изкуствената невронна мрежа натрупва. С подобен механизъм може да се отчете фактът, че по-старите данни по-малко влияят върху това което предстои да се случи.

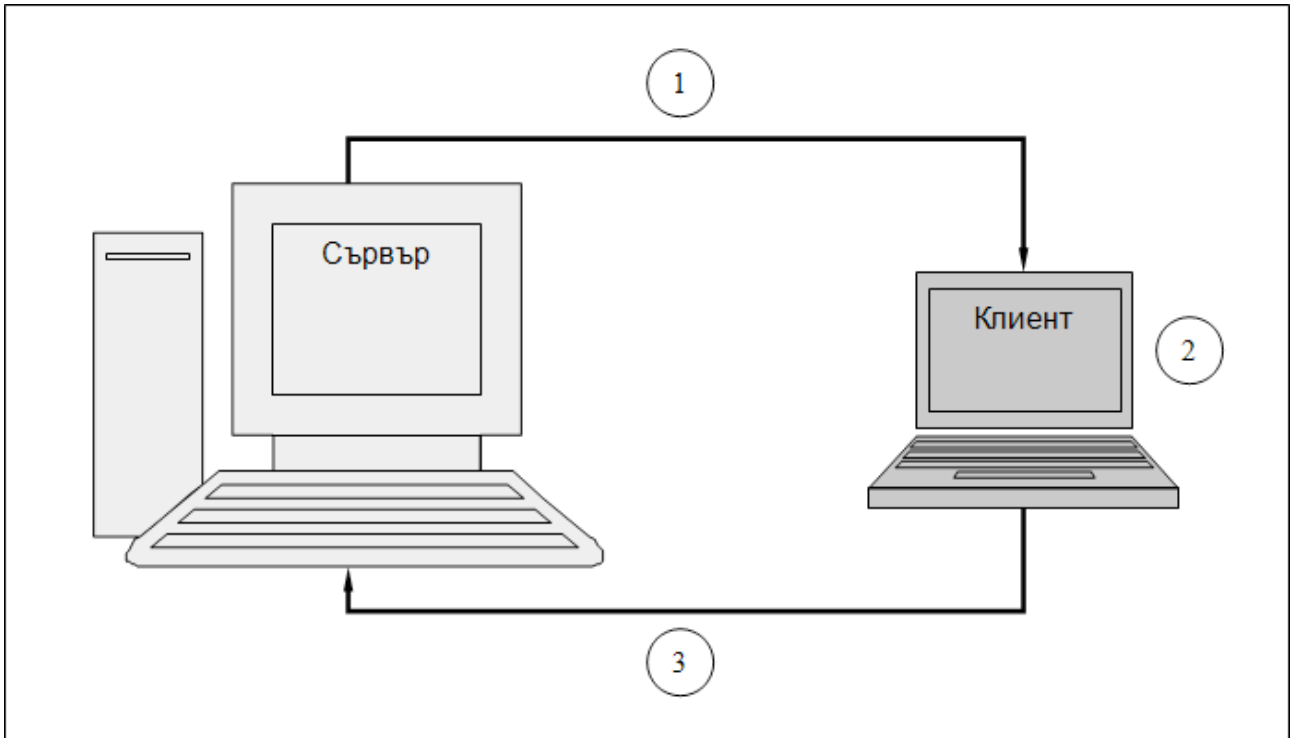
2.6.6 Хибридна реализация с диференциална еволюция и обратно разпространение на грешката



Фиг. 2.6 *Промяна на топологията в изкуствената невронна мрежа за нуждите на алгоритъма за обратно разпространение на грешката.*

Самостоятелното използване на диференциална еволюция, като обучаващ алгоритъм изисква твърде много машинно време извършване на обучението. С цел ускоряване на тази обучението диференциалната еволюция може да се комбинира с обратно разпространение на грешката. Когато се преминава от обучение с диференциална еволюция към обучение с обратно разпространение на грешката трябва да се изключат всички обратни връзки в мрежата. Изключването на връзки в изкуствената невронна мрежа означава промяна на топологията. Обучението с обратно разпространение на грешката ще засяга единствено теглата участващи във връзки от входа, към изхода. Подобен вид модификация на топологията и корекция на част от теглата може да се разглежда като допълнителен вид мутация в диференциалната еволюция.

2.6.7 Обучение на изкуствени невронни мрежи с диференциална еволюция в разпределена среда



Фиг. 2.7 *Обмен на задачи в система за изчисления в разпределена среда.*

При системите за изчисления в разпределена среда централизиран сървър разпределя задания за изчисление на отдалечени компютри (стъпка 1). В някакъв период от време отдалечените компютри изпълняват заданието (стъпка 2). Заданията могат да са оформени като пакети за пресмятане в рамките на няколко минути, но може да са оформени и в пакети за пресмятане в рамките на седмица или месец. Периодът за продължителност на пресмятането силно зависи от естеството на изчисленията които трябва да се извършат. Чрез подходящо регулиране на периода за пресмятане може да се постигне оптимално натоварване на сървъра и максимална актуалност на информацията която сървърът съхранява с цел синхронизиране на разпределения изчислителен процес. Когато клиентът приключи изчислението на възложеното задание връща резултата на сървъра (стъпка 3). Трите основни стъпки в този процес се повтарят докато има работещ сървър и клиентски компютри закачени за него.

2.7 Обобщение

Като кратко обобщение може да се отбележи, че предложения модел за обучение на изкуствени невронни мрежи има редица предимства когато обучението се извършва с диеференциална еволюция която се изпълнява на множество изчислителни машини, а получените резултати се синхронизират на централизиран сървър.

В резултат на научноизследователската работа изложена в тази глава са постигнати следните приноси:

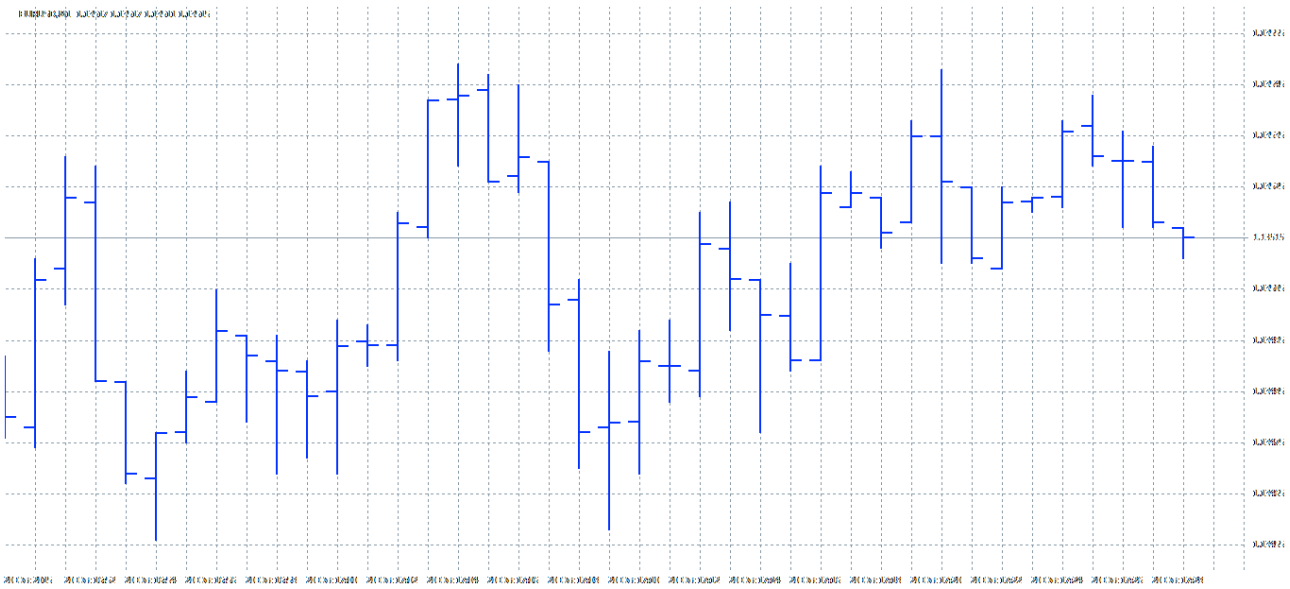
1. Предложен е ефективен начин за разделяне на търсенето в пространството на решенията върху различни изчислителни машини;
2. Предложен е ефективен начин за оптимизацията на теглата в изкуствена невронна мрежа, чрез диференциална еволюция;
3. Предложена е модификация на топологията на изкуствена невронна мрежа с обратни връзки, така че да се изпълнява комбинирано обучение с обратно разпространение на грешката и диференциална еволюция;

Моделът за обучение в разпределена среда е представен в „Работни статии на ИИТ“, на „Юбилейна научна конференция - 40 години катедра Автоматизация на производството” и на International Conference on Large-Scale Scientific Computations. Тези публикации са цитирани в дисертационния труд като референции [BAZA2010], [BATO2011] и [BAZA2011].

Глава 3 - Софтуерна система за прогнозиране на времеви редове с изкуствени невронни мрежи и еволюционни алгоритми

Основен фактор при практическата реализация е наличието на подходящи експериментални данни. Развитието на финансовите пазари, през последните две десетилетия дава изключително добър избор от времево зависима информация. Цената на националните валути се оповестява на така наречения foreign exchange market (FOREX). Търгуването на валутните пазари изключително се улеснява с услугите които предлагат инвестиционните посредници. Самата търговия се извършва с помощта на търговски софтуер наречен търговски терминал. За Източна Европа най-разпространената търговска платформа е MetaTrader 4, произведена от фирмата MetaQuotes Software Corp. [MTMQ2015].

В платформата MetaTrader 4 валутната информация е представена под формата на времеви ред онагледен графично с нива на отваряне, нива на затваряне, нива за най-висока достигната цена и нива на най-ниска достигната цена (Фиг. 3.1). Цялата тази информация е програмно достъпна със средствата на вградения в MetaTrader 4 програмен език, наречен MQL4. Търговският терминал на MetaTrader 4 дава възможност за визуализиране на информацията от времевия ред на няколко различни, равни по дължина, времеви интервала, като следва: M1 - всеки бар отразява една минута, M5 - всеки бар отразява пет минути, M15 - всеки бар отразява петнадесет минути, M30 - всеки бар отразява тридесет минути, H1 - всеки бар отразява един час, H4 - всеки бар отразява четири часа, D1 - всеки бар отразява един ден, W1 - всеки бар отразява една седмица, MN - всеки бар отразява един месец. От ляво на всеки бар (Фиг. 3.1) присъства хоризонтална черта която символизира нивото при което барът е бил отворен. От дясно на всеки бар има втора хоризонтална черта която символизира нивото на което е бил затворен. Долният край на всеки бар показва най-ниското ниво постигнато в рамките на времевия интервал за който се отнася барът. Горният край на всеки бар показва най-високото ниво постигнато в рамките на времевия интервал за който се отнася барът.



Фиг. 3.1 *Котировки EUR/USD в MetaTrader 4.*

3.1 Входна информация в разработената система

Програмните средства на MetaTrader 4, чрез вградения език MQL4, позволяват котировките за всяка валутна двойка да бъдат предоставени извън рамките на търговския терминал. Извеждането на информацията може да стане по няколко различни начина като най-удачен за дисертационния труд е записът на информацията в текстов файл. Преди информацията за всеки бар от графиката с котировките да бъде изведена от търговския терминал информацията се нормализира. На графиката се определя най-високото постигнато ниво, след това най-ниското постигнато ниво и стойностите на всеки бар се нормализират в диапазона [0.0-1.0]. Тази стъпка за нормализация се налага тъй като използваната изкуствена невронна мрежа работи със сигнали нормализирани в диапазона (0.0-1.0). Втората причина за извършването на нормализация е наличието на множество валутни двойки които имат различни номинални стойности. Примерно, отношението EUR/USD е от порядъка на 1.13510 (информация към дата 20.10.2015), докато отношението USD/JPY е от порядъка на 119.830 (информация към дата 20.10.2015). След като изкуствената невронна мрежа предложи своята прогноза се извършва обратната операция по мащабиране на изхода спрямо минималната и

максималната стойност на графиката.

3.2 Изчисления в разпределена среда

По своята същност, MetaTrader 4 е търговски терминал със софтуерна архитектура от тип „клиент-сървър“. Търгуващите клиенти (хора, наречени на професионален жаргон трейдър) получават котировките на валутните двойки от централизиран сървър, управляван от инвестиционния посредник. Тази организация позволява на компютъра, на всеки трейдър да се стартира допълнителен модул за пресмятания (модул предложен в дисертационния труд) който да извършва изчисления по обучението на изкуствени невронни мрежи. Предложеният изчислителен модул представлява стандартен изпълним файл който прочита котировките записани от MQL4 в текстовите файлове. Обучението на конкретна топология изкуствени невронни мрежи за конкретни котировки на конкретен времеви интервал може да се разпредели между множество компютри.

3.3 Развойни средства

Тъй като системата е разделена на няколко логически обособени части, за всяка част са подходящи различни развойни инструменти.

3.3.1 Клиентско приложение

За разработката на програмния код се използват група инструменти за разработка. От страна на MetaTrader 4 естественият програмен език е вградения в търговската платформа MQL4. Малък софтуерен под модул написан на MQL4 има за задача да извежда информацията за времевите редове в текстови файлове и да чете прогнозите генерирани от изкуствените невронни мрежи. Също така MQL4 под модулът има задача да визуализира прогнозата в подходяща форма за разчитане от крайните потребители.

По отношение на разработения в този дисертационен труд софтуерен модул съществуват набор от алтернативи за избор на програмен език. Най-популярните програми езици са: C/C++, Java, C#. За разработката е избран C/C++. Аргументите в негова полза са, че това е програмен език който не е доминиран от конкретна корпоративна фирма, има изключително добра поддръжка на различни операционни системи (MS Windows, Mac OS X, Linux). Изборът е аргументиран и с това, че езикът има добра поддръжка за супер компютри (Open MPI) както и за графични ускорители от тип GPU. Важно е да се отбележи и относителното бързодействие с което се изпълнява програмния код тъй като се свежда до асемблера на конкретната хардуерна архитектура. Не на последно място C/C++ е основният език в платформата BOINC (Berkeley Open Infrastructure for Network Computing) която е една от най-разпространените платформи за изчисления в разпределена среда. Придържайки се към концепцията за пресмятания в BOINC е възможно да се създадат собствени решения за пресмятане в разпределена среда.

3.3.2 Сървър приложение

Тъй като основната идея за пресмятане в разпределена среда е изчисленията основно да се извършват на клиентските машини, то за сървър е удачно да се подберат най-олекотените технологии и инструменти за разработка. Съобщенията разменяни между клиентските компютри и сървъра са в структуриран формат JSON. След 1992 година един от най-широко разпространените комуникационни протоколи е HTTP и това е причината той да бъде избран в дисертационния труд. За обработката на HTTP заявки от страната на сървъра е необходимо постоянно да работи софтуерно приложение наречено уеб сървър. Най-популярните възможности за уеб сървър са Apache, Tomcat и Microsoft IIS. С цел намаляване на финансовите разходи от страна на сървъра за нуждите на дисертационния труд е избран уеб сървър Apache. Уеб сървърът Apache може да се инсталира на множество операционни системи тъй като представлява софтуер с отворен код.

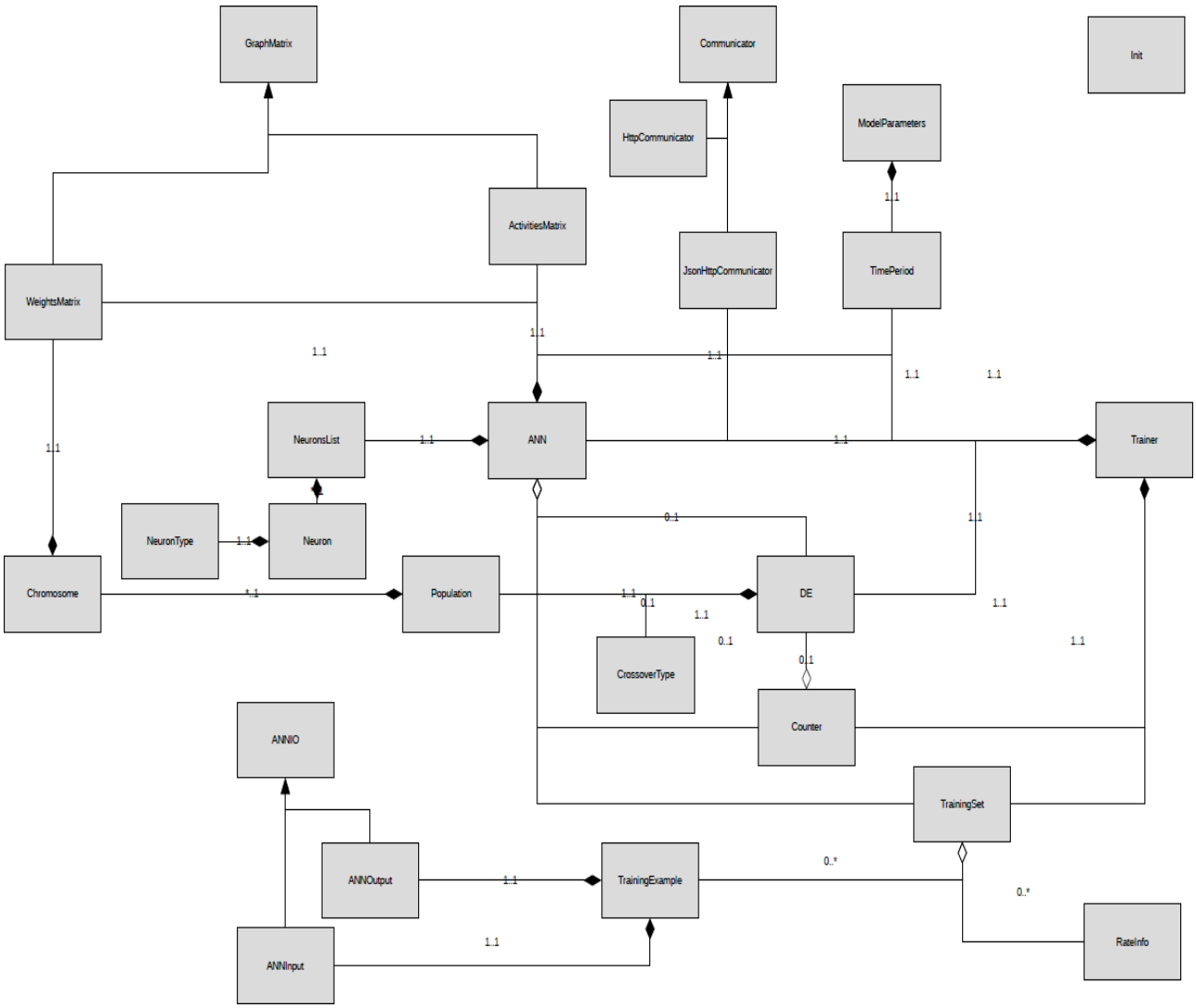
Изборът на Apache уеб сървър е обвързано и с избора на програмен език от страна на

сървъра. Най-разпространените програмни езици за сървърни приложения са PHP, JSP и ASP. В дисертационния труд е избран PHP тъй като е език с основно предназначение за разработка на сървърни скриптове.

Слоят за трайно съхранение на данните от страна на сървъра е представен от релационна система за управление на бази от данни. Най-популярните възможни алтернативи са Oracle, MS SQL Server, PostgreSQL и MySQL. Изборът е MySQL който е значително по-лесна за използване в сравнение с останалите. Компанията Oracle притежава правата над MySQL (към 20.10.2015) което е своеобразна гаранция за стабилното поддържане на тази система за управление на бази от данни.

3.4 Обектно-ориентирана структура на програмния код от страна на клиента

Програмният код работещ от страната на клиентските компютри представлява група класове, компилиращи се до самостоятелен изпълним файл. От класовете в паметта на компютъра се създават обекти. По време на работа вътрешното състояние на обектите се променя. Тази промяна на обектите се съхранява в енергийно независима памет под формата на релационна база данни. За целта се извършва обектно-релационно напасване.



Фиг. 3.2 *Диаграма на класовете.*

За онагледяване на обектно-ориентираната структура, всеки клас е представен със CRC карта (Class Responsibility Collaboration), а връзката между класовете с клас диаграма Фиг. 3.2.

Таб. 3.1 *Отговорници и сътрудници на класа описващ изкуствена невронна мрежа.*

ANN	
Отговорности	Сътрудници
* Представяне на ИНМ. * Приемане на входната за ИНМ информация. * Извеждане на изходната за ИНМ информация.	NuronsList ActivitiesMatrix WeightsMatrix

* Пресмятане на правия пас при разпространение на информацията от входа, към изхода. * Пресмятане на обратния пас, за обучението с ОРГ.	Counter TrainingSet TimePeriod ANNIO

Основният клас в програмата описва изкуствена невронна мрежа което включва вътрешно представяне на мрежата и алгоритми за обработка на постъпващата и излизащата информация.

Таб. 3.2 *Отговорници и сътрудници на класа описващ входа и изхода в изкуствената невронна мрежа.*

ANNIO	
Отговорности	Сътрудници
* Служи като контейнер за вектори от входни/изходни данни към/от ИНМ.	-

Входно-изходната информация представлява вектор от реални числа, а който е предназначен отделен базов клас.

Таб. 3.3 *Отговорници и сътрудници на класа описващ входа в изкуствената невронна мрежа.*

ANNInput	
Отговорности	Сътрудници
* Контейнер за входни данни към ИНМ.	ANNIO

Входната информация е вектор от реални числа и представлява част от входно-изходен вектор.

Таб. 3.4 *Отговорници и сътрудници на класа описващ изхода от изкуствената невронна мрежа.*

ANNOutput	
Отговорности	Сътрудници

* Контейнер за изходни данни от ИНМ.	ANNIO

Изходната информация е вектор от реални числа и представлява част от входно-изходен вектор.

Таб. 3.5 *Отговорници и сътрудници на класа описващ статичните връзки в изкуствената невронна мрежа.*

ActivitiesMatrix	
Отговорности	Сътрудници
* Определя матрица на съседство между възлите на ИНМ. Числено представя топологията, заложена в ИНМ.	GraphMatrix

Матрицата за свързаност на възлите в изкуствените невронни мрежи е двумерен масив от реални числа който определя матрицата на съседство. Използва се за определяне на силата на връзките в изкуствените невронни мрежи като по този начин определя топологията която мрежата има.

Таб. 3.6 *Отговорници и сътрудници на класа описващ хромозомите в диференциалната еволюция.*

Chromosome	
Отговорности	Сътрудници
* Представя хромозома в популацията на ДЕ (матрица с тегла на ИНМ). * Носи информация за жизнеността си. * Генерира хромозома със случайни числа.	WeightsMatrix

Хромозомите са обекти които изграждат популацията на диференциалната еволюция. Всяка хромозома представлява комплект тегла за изкуствена невронна мрежа и има допълнителна характеристика за жизнена стойност на хромозомата.

Таб. 3.7 *Отговорници и сътрудници на базов клас за осъществяване на комуникацията.*

Communicator	
Отговорности	Сътрудници
* Задава базова рамка за комуникационен интерфейс (служи за наследяване).	DE ANN NeuronList WeightMatrix ActivitiesMatrix ModelParameters TimePeriod RateInfo Counter

Комуникацията с отдалечения сървър е поверена на група класове които са организирани в йерархия от наследявания. Този подход е избран за да бъде възможно разширяване на системата с допълнителни комуникационни протоколи. Communicator е базов клас наследен от по-специфични реализации на комуникационната функционалност.

Таб. 3.8 *Отговорници и сътрудници на класа описващ броячите за събиране на статистика.*

Counter	
Отговорности	Сътрудници
* Натрупва статистическа информация за работата на системата. * Поддържа броячи под формата на ключ-стойност двойки.	-

За да се събира статистика по време на работа за системата е представен контейнер с броячи. На база на информацията от броячите се прави статистическа оценка за ефективността на обучението и точността на прогнозите.

Таб. 3.9 *Отговорници и сътрудници на класа описващ типовете кръстосване.*

CrossoverType	
Отговорности	Сътрудници
* Задава различни възможности за операцията по кръстосване в ДЕ.	-

В алгоритъма за диференциална еволюция могат да се приложат различни типове

кръстосване което често зависи от избраната топология на изкуствената невронна мрежа.

Таб. 3.10 *Отговорници и сътрудници на класа описващ диференциалната еволюция.*

DE	
Отговорности	Сътрудници
* Реализира алгоритъма на ДЕ. * Изпълнява операцията по селекция в ДЕ. * Изпълнява операцията за рекомбинация в ДЕ.	Population Counter ANN CrossoverType

Алгоритъмът за диференциална еволюция е реализиран в отделен клас който има отговорност да извършва селекцията, рекомбинацията и оценката на отделните индивиди в популацията.

Таб. 3.11 *Отговорници и сътрудници на класа описващ матрица на съседство.*

GraphMatrix	
Отговорности	Сътрудници
* Задава матрица на съседство в насочен тегловен граф (служи за наследяване).	-

Тъй като теглата и активностите на връзките в изкуствените невронни мрежи се моделират с матрица на съседство е удачно залагането на базов клас който да бъде наследен.

Таб. 3.12 *Отговорници и сътрудници на класа описващ комуникация по HTTP протокола.*

HttpCommunicator	
Отговорности	Сътрудници
* Комуникация с отдалечен уеб сървър, с базов протокол HTTP. * Синтактичен анализ (парсване) на неструктурирана информация, която се обменя към и от уеб сървъра.	DE ANN NeuronList WeightMatrix Communicator ActivitiesMatrix ModelParameters TimePeriod RateInfo Counter

Един от най-разпространените комуникационни протоколи е HTTP което е основна аргументация в системата да бъде предоставена възможност за обмен на съобщения между клиента и сървъра точно на основата на HTTP протокола.

Таб. 3.13 *Отговорници и сътрудници на класа описващ параметри за инициализация на системата.*

Init	
Отговорности	Сътрудници
* Зареждане на параметри за настройката на системата от инициализационен файл.	-

При зареждане на системата без наличие на връзка към сървъра е важно параметрите за обучение на изкуствените невронни мрежи да бъдат инициализирани и поради тази причина е въведена възможността инициализацията да се извърши от инициализационен файл.

Таб. 3.14 *Отговорници и сътрудници на класа описващ JSON базирана комуникация по HTTP комуникационен протокол.*

JsonHttpCommunicator	
Отговорности	Сътрудници
* Комуникация с отдалечен уеб сървър, с базов протокол HTTP. * Синтактичен анализ (парсване) на JSON структурирана информация, която се обменя към и от уеб сървъра.	DE ANN NeuronList WeightMatrix Communicator ActivitiesMatrix ModelParameters TimePeriod RateInfo Counter

При комуникация по протокола HTTP е възможно информацията да бъде предавана в текстов вид без да бъде структурирана но много по-ефективно и защитено от грешки е, ако информацията бъде пакетирана в JSON съобщения. В този аспект е добавена възможност за

комуникация между клиента и сървъра на база HTTP+JSON.

Таб. 3.15 *Отговорници и сътрудници на класа описващ параметрите на модела.*

ModelParameters	
Отговорности	Сътрудници
* Задава стойности на параметрите, които има финансовият ред, който ще бъде прогнозиран, както и параметрите на ИНМ/ДЕ, използвани при прогнозирането.	TimePeriod

Параметрите на всеки модел (ИНМ+ДЕ+ВР) се обработват от помощен клас, така че методите да получават само един обект, а не серия от параметри.

Таб. 3.16 *Отговорници и сътрудници на класа описващ неврон в изкуствената невронна мрежа.*

Neuron	
Отговорности	Сътрудници
* Представя невроните в ИНМ. * Поддържа различни типове неврони. * Поддържа стойност на изходния сигнал. * Поддържа стойност на грешката, за неврона, при обратния пас от обучението на ИНМ с ОРГ.	NeuronType

Отделен клас моделира поведението на невроните. Методите на този клас имат задача да извършват пресмятанията и да запазват вътрешното състояние на неврона.

Таб. 3.17 *Отговорници и сътрудници на класа описващ типове неврони в изкуствената невронна мрежа.*

NeuronType	
Отговорности	Сътрудници
* Задава различните възможности за типове неврони.	-

Основните типове неврони биват - обикновени, входящи, изходящ и bias (отместващи). Според своя тип всеки неврон изпълнява различна роля в мрежата. Възможно

е неврони да съчетават повече от една функция (примерно, неврон който е входен, а в същото време и изходен за мрежата).

Таб. 3.18 *Отговорници и сътрудници на класа описващ списък от неврони.*

NeuronsList	
Отговорности	Сътрудници
* Поддържа списък с всички неврони в ИНМ. * Поддържа информация за броя неврони от различен тип.	Neuron

Всяка мрежа е представена неврони и тегла за връзките между тях. В паметта на компютъра се използва списък от неврони като контейнер за обектите от тип неврон.

Таб. 3.19 *Отговорници и сътрудници на класа описващ популацията в диференциалната еволюция.*

Population	
Отговорности	Сътрудници
* Поддържа популацията за ДЕ. * Инициализира популация със случайни числа. * Поддържа информация за най-добрия индивид в популацията.	Chromosome

Популацията е представена в паметта на компютъра като контейнер от обекти тип хромозома.

Таб. 3.20 *Отговорници и сътрудници на класа описващ информацията за котировките.*

RateInfo	
Отговорности	Сътрудници
* Съхранява информация за един бар от графиката на времевия ред (нива на - отваряне, най-ниска постигната, най-висока постигната, затваряне, търгуван обем, момент от времето).	-

Времевия ред за стойността на валутните двойки се представя под формата на графика

с барове. Всеки бар носи информация за котировката на валутите в конкретен момент от времето.

Таб. 3.21 *Отговорници и сътрудници на класа описващ времевите периоди на котировките.*

TimePeriod	
Отговорности	Сътрудници
* Задава възможните времеви период за които има събрани исторически данни (съответстват на брой минути).	-

В програмата MetaTrader4 времевите редове са организирани на равни интервали от време. Програмата дава възможност финансовата информация да бъде представяна във времеви редове с различен период, между отделните отчитания. Времевите периоди са както следва: M1 (една минута), M5 (пет минути), M15 (петнадесет минути), M30 (тридесет минути), H1 (един час), H4 (четири часа), D1 (един ден), W1 (една седмица), MN1 (един месец).

Таб. 3.22 *Отговорници и сътрудници на класа описващ процеса на обучение.*

Trainer	
Отговорности	Сътрудници
* Извършва процеса по обучение на ИНМ. * Използва се за извличане на прогнози. * Обновява обучаващото множество. * Докладва за най-доброто намерено решение.	TimePeriod Counter ANN DE TrainingSet JsonHttpCommunicator

Множеството от обекти за изкуствена невронна мрежа, диференциалната еволюция, обучаващо множество, комуникационен обект и броячи се помества в обект от класа за трениране на мрежата. Задачата на този клас е да служи като контейнер от високо ниво за останалите обекти.

Таб. 3.23 *Отговорници и сътрудници на класа описващ тренировъчните примери.*

TrainingExample	
Отговорности	Сътрудници
* Представя тренировъчните примери за ИНМ, които включват вход, изход и очакван изход.	ANNIO

Тренировъчният пример е стойност на невроните, както следва: входни сигнали в изкуствени невронни мрежи, изходни сигнали от изкуствени невронни мрежи и очаквани изходни сигнали от изкуствени невронни мрежи.

Таб. 3.24 *Отговорници и сътрудници на класа описващ тренировъчното множество.*

TrainingSet	
Отговорности	Сътрудници
* Съдържа набор от тренировъчни примери. * Трансформира информацията от външната среда във вид подходящ за въвеждане в и извеждане от ИНМ.	RateInfo TrainingExample ANNIO TrainingSet

Обучаващото множество е последователност от три паралелни масива както следва: вход в изкуствени невронни мрежи, изход на изкуствени невронни мрежи и очаквана стойност на изхода на изкуствени невронни мрежи.

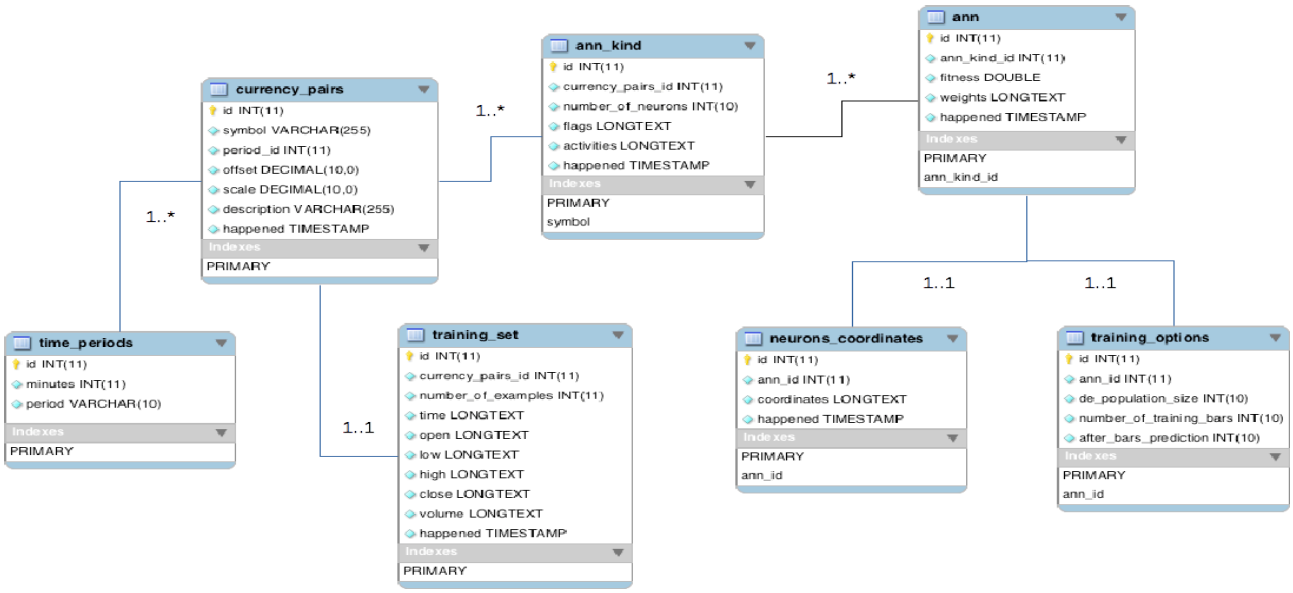
Таб. 3.25 *Отговорници и сътрудници на класа описващ теглата в изкуствената невронна мрежа.*

WeightsMatrix	
Отговорности	Сътрудници
* Определя матрица на съседство между възлите на ИНМ. Числено представя обучените тегла на ИНМ.	GraphMatrix

Матрицата за теглата на връзките в изкуствена невронна мрежа е двумерен масив от реални числа който определя матрицата на съседство. Използва се за определяне на силата на сигналите между невроните в изкуствени невронни мрежи като по този начин определя знанието което мрежата има.

3.5 Релационен модел за съхранение на данните от страна на сървъра

Съхранението на информацията в системата се извършва на централизирания сървър с помощта на система за управление на бази от данни, в която данните са моделирани под формата на същности и релации, изобразени на Фиг. 3.3 като диаграма на базата данни.



Фиг. 3.3 *Диаграма на базата данни..*

В термините на системите за управление на бази от данни същностите и релациите се представят под формата на таблици в базата данни, както следва:

Таб. 3.26 *Таблица в базата данни, описваща конкретна изкуствена неверонна мрежа.*

ann	
Атрибути	Връзки
id	
ann_kind_id	ann_kind.id
fitness	
weights	
happened	

Всяка изкуствена невронна мрежа е описана с матрица на съседство (полето wights) която съдържа теглата на връзките в мрежата. Различните мрежи имат различен комплект тегла и съответно биха дали различна точност при прогнозирането. Общата допусната грешка при прогнозирането се използва за жизнена стойност в алгоритъма за диференциална еволюция (полето fitness). Различните изкуствени невронни мрежи се характеризират с тип на мрежата и това се отразява с помощна таблица връзката към която е полето ann_kind_id.

Таб. 3.27 *Таблица в базата данни, описваща типовете изкуствена неверонна мрежа.*

ann_kind	
Атрибути	Връзки
id	
currency_pairs_id	currency_paris.id
number_of_neurons	
flags	
activities	
happened	

Видовете изкуствени невронни мрежи основно се описват с топология на възлите и силата на връзките между отделните възли (полето activities). Полето activiteis също представлява матрица на съседство и определя дали между два възела (включително и примките) съществува връзка и колко силна е тази връзка. Всеки вид изкуствена невронна мрежа има определен брой неврони (полето number_of_neurons) и информация какъв тип е всеки от невроните (полето flags). Тъй като не се търсят връзки между отделни валутни двойки, всеки вид изкуствена невронна мрежа се отнася само за конкретна валутна двойка и конкретен период на времевия ред (външна връзка през полето currency_pairs_id).

Таб. 3.28 *Таблица в базата данни, описваща валутните двойки.*

currency_pairs	
Атрибути	Връзки
id	

symbol	
period_id	time_periods.id
offset	
scale	
description	
happened	

Всеки времеви ред в системата отразява информацията за една валутна двойка. Валутните двойки се характеризират със символно название (примерно EURUSD, полето symbol). Втората характеристика на времевия ред е периодът (външна връзка през полето period_id).

Таб. 3.29 *Таблица в базата данни, описваща координатите на невроните при визуализация.*

neurons_coordinates	
Атрибути	Връзки
id	
ann_id	ann.id
coordinates	
happened	

За целите на системната администрация, всяка изкуствена невронна мрежа се визуализира в административен панел. На практика се изобразяват невроните като възли в граф и теглата като ребра на графа. За да се осъществи визуализацията всеки неврон се описва с координати в двумерната равнина (полето coordinates).

Таб. 3.30 *Таблица в базата данни, описваща времевите периоди на котировките.*

time_periods	
Атрибути	Връзки
id	
minutes	
period	

В програмата MetaTrader4 са заложени няколко стандартни времеви интервала за

времевите редове. Интервалите се описват със символно название (полето period) и брой минути на които отговаря интервалът (полето minutes).

Таб. 3.31 *Таблица в базата данни, описваща параметрите на обучението.*

training_options	
Атрибути	Връзки
id	
ann_id	ann.id
de_population_size	
number_of_training_bars	
after_bars_prediction	

Системата би могла да определя параметри за обучението на изкуствени невронни мрежи с отделна таблица в релационната база данни. Като параметри на обучението се използват броят индивиди в диференциалната еволюция, броят барове за вход в мрежата, броят барове за изход от мрежата и кой вид мрежа ще се обучава.

Таб. 3.32 *Таблица в базата данни, описваща обучаващото множество.*

training_set	
Атрибути	Връзки
id	
currency_pairs_id	currency_pairs.id
number_of_examples	
time	
open	
low	
high	
close	
volume	
happened	

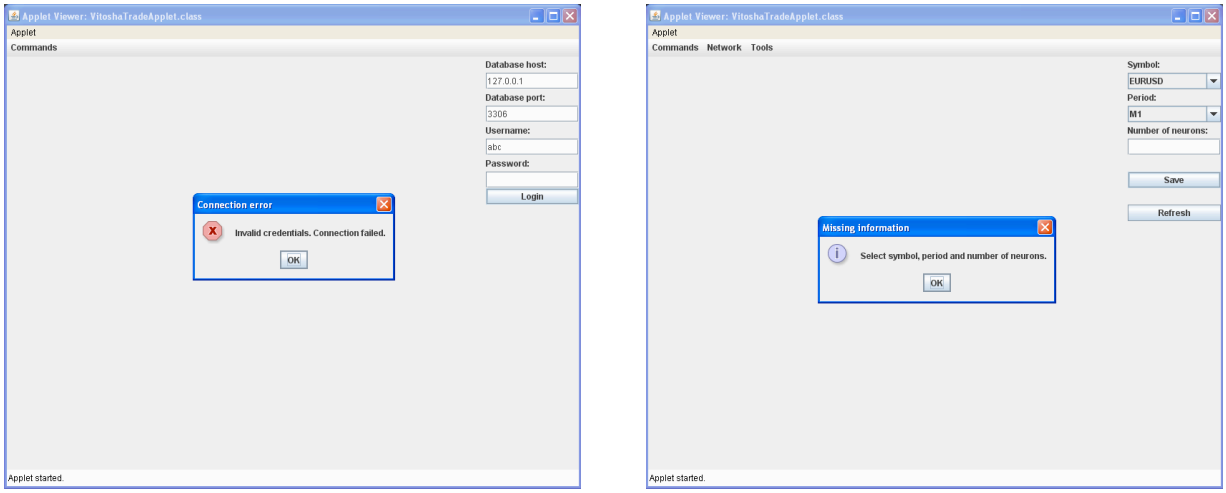
Тренировъчните примери за изкуствената невронна мрежа представляват самия времеви ред който постъпва в системата от изхода на програмата MetaTrader4. Времевият ред

съдържа информация за валутната двойка, периода на времевия ред, колко стойности от времевия ред са представени и група паралелни масиви които имат следното значение: time - момент във времето за който се отнася стойността, open - ниво на отваряне за съответния интервал, close - ниво на затваряне за съответния интервал, low - най-ниско постигнато ниво за съответния интервал, high - най-високо ниво постигнато за съответния интервал, volume - изтъргуван обем в конкретния времеви интервал.

Полетата id са служебни полета и служат за номерация на записите в таблиците. При представения модел всичките id полета са с характеристика за автоматично увеличаване на номерацията и са заложени да бъдат първични ключове. Полето happened е помощно поле което записва time stamp информация за момента от времето в който е направен съответният запис в съответната таблица от базата данни.

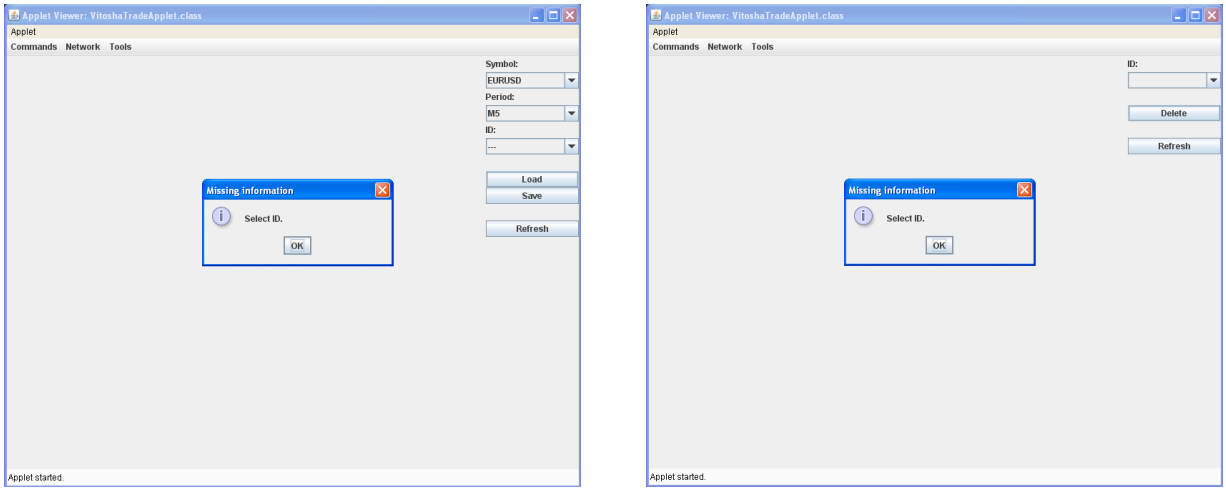
3.6 Административен модул за наблюдение и настройване на системата

Модулът за наблюдение и настройване на системата се ползва за задаване на параметри при които протича обучението на изкуствени невронни мрежи. Модулът е разработен под формата на Java Applet за да бъде достъпен от различни компютърни системи, използващи стандартен уеб браузър. Модулът е реализиран под формата на двуслойна софтуерна архитектура (потребителски интерфейс - база данни). Системата за сигурност (автентификация и авторизация) е изцяло базирана на системата за сигурност, която предоставя системата за управление на бази от данни. Административният модул е предназначен единствено за операторите на системата които са в ролята на доверени лица. Допълнително нивото на сигурност в системата за управление на бази от данни ограничава достъпа до данните само в рамките на локалната мрежова инфраструктура.



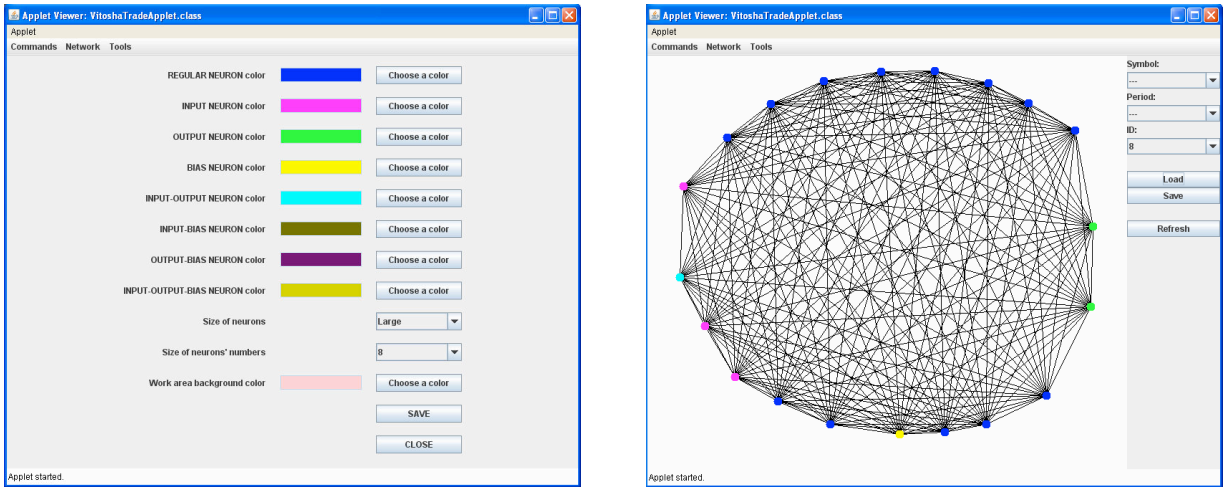
Фиг. 3.4 *Екран за вход в административната система [NIRA2012][ANMO2011] (ляво) и екран за създаване на нова изкуствена невронна мрежа в базата данни [NIRA2012] [ANMO2011] (дясно).*

Основна задача на модула за администриране е да осъществява връзка с базата данни (Фиг. 3.4 - ляво) и да предоставя в разбираем за оператора графичен вид, информацията за различните изкуствени невронни мрежи, съхранени в енергийно независимата памет. Административният модул дава възможности за създаване на нови записи в базата данни, които да отразяват информацията за съответен вид изкуствена невронна мрежа (Фиг. 3.4 - дясно).



Фиг. 3.5 *Екран за избор на изкуствена невронна мрежа, съхранена в базата данни [NIRA2012][ANMO2011] (ляво) и екран за изтриване на изкуствена невронна мрежа от базата данни [NIRA2012][ANMO2011] (дясно).*

Съществуващите записи на изкуствени невронни мрежи в базата данни могат да бъдат зареждани и визуализирани в административната система (Фиг. 3.5 - ляво). Записани изкуствени невронни мрежи в базата данни могат да бъдат изтривани с помощта на административната система (Фиг. 3.5 - дясно).

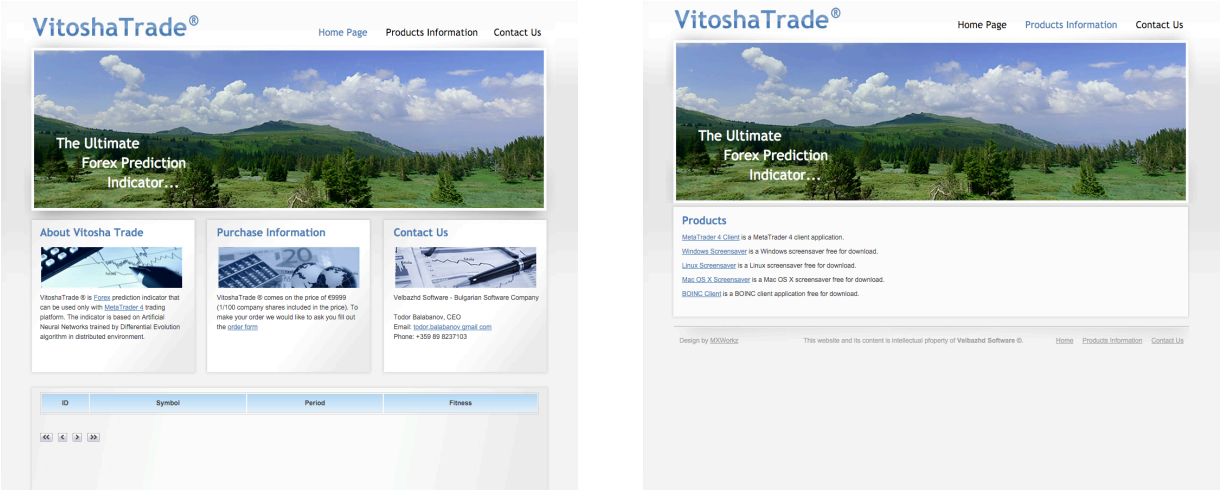


Фиг. 3.6 *Екран за настройка на опциите в административния модул [NIRA2012][ANMO2011] (ляво) и екран за визуализация на изкуствена невронна мрежа [NIRA2012][ANMO2011] (дясно).*

За визуализацията на различните компоненти от изкуствените невронни мрежи (тегла и неврони с различно предназначение) се използват набор от опции, които могат да се настройват в специално изработена диалогова кутия (Фиг. 3.6 - ляво). Най-съществената възможност на административната система е визуализирането на изкуствени невронни мрежи с всички връзки, както под формата на тегла, така и под формата на активности. Невроните в мрежата се изобразяват с различни цветове, спрямо ролята, която изпълняват в мрежата (входни, изходни, вътрешни или отместващи). Разположение на невроните върху визуалното пространство може да се променя, с помощта на мишката и клавиатурата (Фиг. 3.6 - дясно).

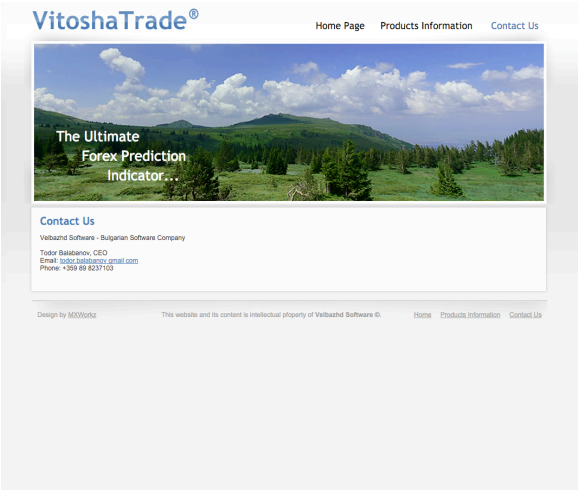
3.7 Публичен модул за информация от системата

Публичният модул представлява уеб сайт, разработен с HTML, CSS, PHP. Целта на публичния модул е да представя информация от системата, която е за публично ползване. Публичната информация включва списък на различните изкуствени невронни мрежи в базата данни, различната ефективност до която отделните мрежи са обучени и техния служебен идентификатор в рамките на базата данни (Фиг. 3.7 - ляво).



Фиг. 3.7 *Основна уеб страница за публикуване на публично достъпна информация от системата (ляво) и уеб страница със списък на всички публично достъпни модули от системата (дясно).*

Цялостната система включва различни модули, част от които са достъпни за публично ползване. Списък с публично достъпните модули е представен в допълнителна уеб страница (Фиг. 3.7 - дясно).



Фиг. 3.8 *Уеб страница с информация за контакт.*

Публичният модул съдържа и уеб страница с информация за контакт (Фиг. 3.8).

3.8 Обобщение

Като кратко обобщение може да се отбележи, че предложеното софтуерно решение за обучение на изкуствени невронни мрежи има редица предимства, като - ниска себестойност за опериране на системата, висока надеждност за непрекъсната работа на системата и висока степен на разширяемост.

В резултат на научноизследователската работа изложена в тази глава са постигнати следните приноси:

1. Разработен е обектно-ориентиран програмен код за пресмятане от страната на клиента;
2. Разработена е оптимизирана релационна база данни за ефективно съхранение на изчислените резултати;
3. Разработен е изчистен и ефективен за използване графичен потребителски интерфейс;

Програмният код на системата е достъпен за свободен достъп, а части от системата са представени в публикации референции [ANMO2011] и [NIRA2012].

Глава 4 - Приложение на системата за прогнозиране и анализ на експериментални резултати

Сложността при прогнозирането на валутните пазари се дължи на динамиката в изменението на пазарните процеси и множеството фактори, оказващи влияние върху цената. Предложеният теоретичен модел и конкретната му софтуерна реализация ще послужат за изследване на проблемите разгледани по-долу в текста.

При нарастване броя на възлите в изкуствените невронни мрежи броят на теглата се увеличава значително което може допълнително да затрудни процеса на обучение, особено при по-сложна топология на изкуствени невронни мрежи. Първото направление за изследване е каква топология на изкуствени невронни мрежи би била най-ефективна за целите на прогнозирането. Разгледани са класическа трислойна изкуствена невронна мрежа, многослойна изкуствена неврона мрежа с обратни връзки и пълно-свързана изкуствена невронна мрежа. Освен броят слоеве и конкретният набор връзки между тях от съществено значение е и общият брой неврони.

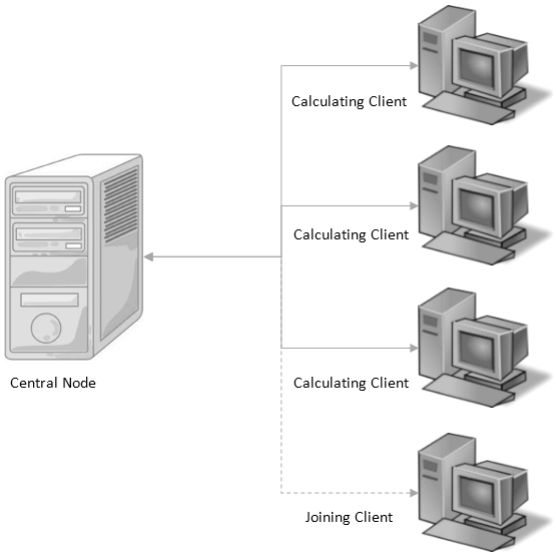
Съществен момент в изследването е изборът на вида на входно-изходните данни при работа с изкуствени невронни мрежи. По отношение на изхода, желателно е да се получава директно прогнозна стойност (конкретно число или разлика на прогнозирания момент, спрямо текущия). По отношение на входните данни са на лице следните възможности: 1. Да се подава стойност за време; 2. Да се подава стойност на величината от предходен момент; 3. Да се подава съвкупност от стойности на величината в предходни моменти; 4. Да се подават разлики в стойността на величината от предходни моменти; 5. Да се подава комбинация от предходни стойности на величината и информация за времето.

Освен с данни за време и стойност на величина изкуствените невронни мрежи може да се обучава за класифициране на „истински” и „фалшиви” сигнали подавани от финансови индикатори и/или осцилатори. При този вариант самото прогнозиране се получава от комбинация индикатор/осцилатор и изкуствени невронни мрежи която потвърждава

достоверността на подаваните сигнали.

4.1 Инцидентно участие на възли в система за разпределени изчисления

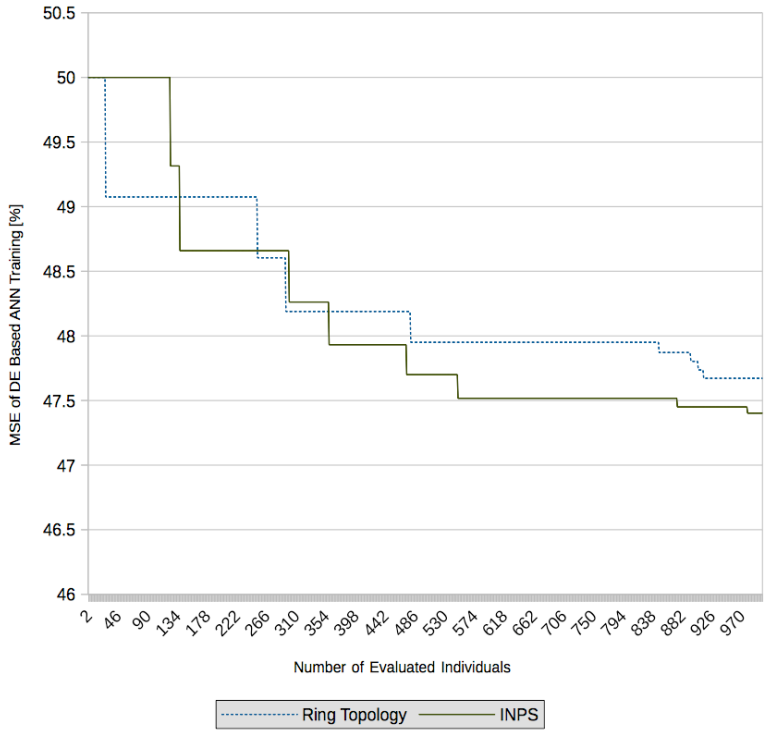
В предложената система за разпределени изчисления, основната популация се намира на един централен възел (сървър). Всеки клиент се закача към системата и получава подмножество от глобалната популация (Фиг. 4.1). След тази първоначална инициализация оптимизацията протича на локално ниво като към централния възел се изпращат само индивиди от локалната популация които имат по-добра жизнена стойност спрямо вече известната най-добра глобална жизнена стойност. По този начин на сървъра се образува един своеобразен елит от индивиди. В същото време сървърът разполага с механизми да подбира индивиди от този елит, но също така да инициализира и случайна локална популация.



***Фиг. 4.1** Инцидентно включване на възли в разпределената система.*

На практика, централният възел служи за хранилище на индивидите с най-добра постигната жизненост от локалните процедури за оптимизация (своеобразна форма на елитаризъм). Тази част от системата за обмен на индивиди между отделните изчислителни

възли е разгледана в изолиран вариант в [TBIZ2015] като е направено сравнение между възли с инцидентно включване и изключване и обмен на информация по кръгова топология.



Фиг. 4.2 *Сравнение между инцидентно включване на възли и кръгова топология.*

Като експериментални данни са използвани котировките EUR/USD, при дневен интервал, за периода 01.01.2014 до 31.03.2014 (Приложение А). От проведените експерименти се вижда, че при инцидентно включване на възлите сходимостта на оптимизационният процес е по-добра (Фиг. 4.2).

4.2 Сравнение между C++ и JavaScript, при правия пас на изкуствена невронна мрежа

В настоящото изследване [BAGE2016a] е представен модел за обучение на изкуствени невронни мрежи с помощта на обучаващ алгоритъм - диференциална еволюция който е от групата на еволюционните алгоритми. Обучението на изкуствени невронни мрежи се осъществява в разпределена среда като мрежата е представена във вид на JavaScript

програмен код, а комуникацията с централния възел се извършва с помощта на AJAX асинхронни заявки. Целта за обучение на изкуствени невронни мрежи е прогнозиране на стойностите за различни валутни двойки на глобалния валутен пазар.

За целите на прогнозирането се използва математически модел базиран на изкуствени невронни мрежи, чието обучение се извършва с диференциална еволюция, под формата на изчисления в разпределена среда. Топологията на изкуствените невронни мрежи е обект на научно изследване и поради тази причина в разработената система топологията се въвежда от оператора на системата. Допустимите видове топологии са: многослойни, многослойни с обратна връзка и пълно-свързани. Моделът е базиран на класическа изкуствена невронна мрежа, каквито се използват в моделите с обратно разпространение на грешката. Като предавателна функция се използва линейна предавателна функция:

(12) $$u[i] = sum(w[i][j]*x[j])$$

Предавателната функция определя по какъв начин входните сигнали в комбинация с тегловните коефициенти, ще влияят върху активността на съответния неврон. Макар и да са възможни модели с друг вид предавателни функции на този етап предпочитанията са в ползата на най-опростения модел базиран на линейността.

Резултатът от предавателната функция е необходимо да се нормализира с помощта на прагова функция (в избрания модел сигмоидна функция със стойности в интервала от 0 до 1), тъй като различният брой връзки при различните неврони би влиял различно, ако не се нормализира.

(13) $$x[i] = 1 / (1 + exp(-u[i]))$$

Сигмоидната функция е предпочитана за прагова функция поради нейните подходящи свойства по отношение на диференцируемост и асимптотичност за плюс безкрайност и минус безкрайност. Възможно е използването на двоична функция или линейна функция (с цел подобряване на бързодействието), но техните свойства влошават резултатите, които изкуствената невронна мрежа може да постигне. Ако мрежата работи със стойности от -1 до

+1 то може да се използва хиперболичен тангенс като прагова функция.

Обучението на изкуствени невронни мрежи се осъществява с помощта на диференциална еволюция. В рамките на алгоритъма диференциална еволюция всяка хромозома представя един комплект тегла за конкретна топология изкуствена невронна мрежа към която се отнасят определен брой обучаващи примери. Последователно отделните хромозоми (комплекти тегла) се зареждат в структурата на изкуствената невронна мрежа, след което се подават обучаващите примери, изчислява се грешката, която мрежата допуска за всеки пример, сумират се грешките от отделните примери и се определя коефициент за жизнеспособност на хромозомата (комплекта тегла). Начинът по който се определя коефициентът на жизнеспособност е един от най-ключовите проблеми от които зависи успехът на цялостното прогнозиране. При времевите редове е разумно обучаващите примери да се подават в хронологична последователност което би било проблем при мрежи обучавани с алгоритъм за обратно разпространение на грешката. Също така, разумно е хронологично по-старите обучаващи примери да оказват по-малко влияние при формирането на коефициента за жизнеспособност (схеми за „стареене“ на данните). След като бъдат оценени отделните хромозоми те влизат в изчислителната схема на диференциалната еволюция като над тях се извършва селекция, кръстосване и мутация.

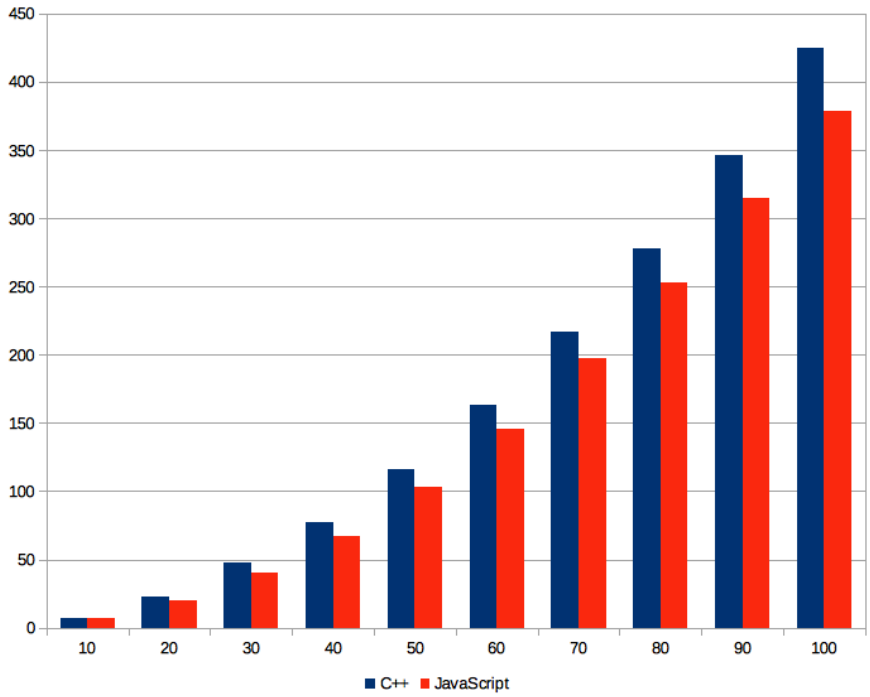
При паралелния вариант на алгоритъма за диференциална еволюция се създават локални копия на изкуствените невронни мрежи и на диференциалната еволюция. Обучението протича локално като всяка хромозома в популацията обозначава отделен индивид в пространството на търсенето. Макар и различни индивидите се „скупчват“ относително близко един до друг в това пространство. При разделяне на изчисленията върху множество изчислителни машини подобно „скупчване“ на индивидите може да доведе до локално изследване на различни области от пространството на решенията. В този контекст от съществено значение е политиката за синхронизиране. Процесът на синхронизиране включва излъчването на най-добрите индивиди и събирането им на едно общо централизирано място (клиент-сървър архитектура). Това множество на глобално най-устойчивите индивиди се използва при създаването на следващи локални популации. Освен клиент-сървър архитектура е възможно да се реализират и решения от тип „точка-в-точка“. При такива решения отсъства централизиран възле, а всяко локално работещо приложение се грижи за комуникацията с

други локално работещи приложения. Основно предимство на предложената разпределена система е нейната изключително висока степен на разширяемост.

Предимствата на предложения модел се състоят в това, че чрез използването на диференциална еволюция за обучението на изкуствени невронни мрежи се избяга опасността от катастрофално забравяне което е възможно при обучение с обратно разпространение на грешката. Второто предимство е възможността чрез диференциална еволюция да се обучават изкуствени невронни мрежи с разнообразни връзки. Третото предимство е възможността чрез диференциална еволюция да се обучават изкуствени невронни мрежи без да е от значение редът по който се подават обучаващите примери (избягва се заучаването на реда за подаване на примерите). Четвъртото предимство е възможността да се обучават различни копия на изкуствени невронни мрежи и това да става паралелно водещо до подобряване на бързодействието и по-добро покритие на пространството за търсене.

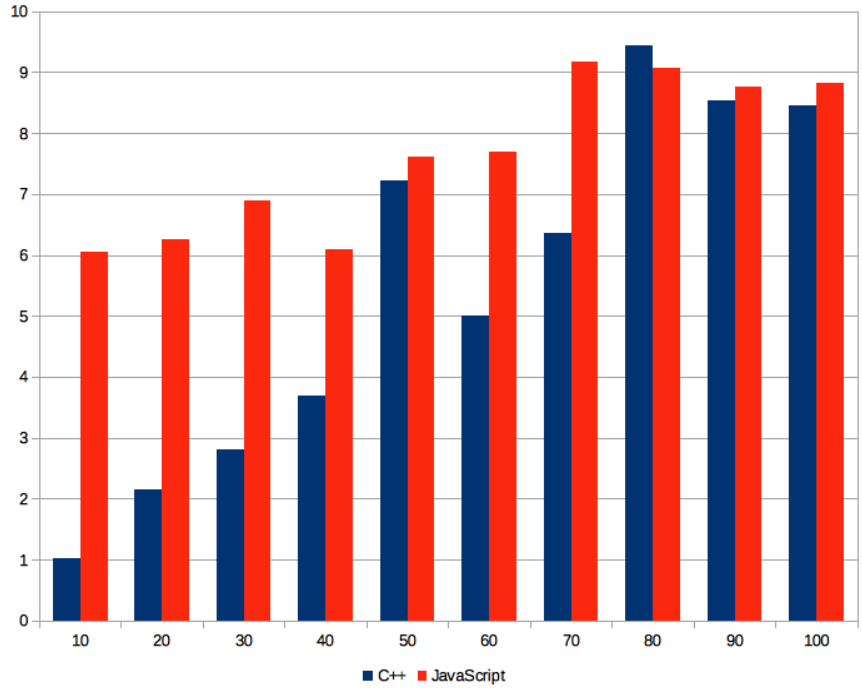
Използването на изкуствени невронни мрежи независимо за решението на каква задача е свързано с много бавен и труден процес на обучение (основен недостатък). Макар и много ефективни след като бъдат обучени изкуствените невронни мрежи за обучението им се изискват значителни изчислителни ресурси. Използването на диференциална еволюция забавя процеса на обучение в сравнение с алгоритъма за обратно разпространение на грешката, но поради множеството изброени предимства диференциалната еволюция е предпочитан пред обратно разпространение на грешката. Хибридният модел в който диференциална еволюция се комбинира с обратно разпространение на грешката води до компромис по отношение топологията на изкуствените невронни мрежи. Чисто технологично разработването на системи за изчисления в разпределена среда е значително по-сложно от писането на линейни програми и дори по-сложно от писането на паралелни програми.

Бързодействието, при изпълнението на JavaScript, е основният аргумент езикът да се избягва при извършването на големи по обем изчисления или при изчисления, които изискват голяма точност. JavaScript попада в групата на скриптовите езици, чийто код не се компилира до инструкции за процесора, а се интерпретира от програмни модули наречени интерпретатори.



Фиг. 4.3 *Сравнение в бързодействието при разпространение на информацията в изкуствена невронна мрежа, по време на правия пас, между C++ и JavaScript. По оста X са броят неврони, а по оста Y е бързодействието в милисекунди.*

Във всеки един момент интерпретаторът може да прекрати изпълнението на програмния код и поради тази причина не може да се гарантира достатъчна надеждност на изчисленията. За да се провери разликата в бързодействието на предложеното AJAX-JavaScript решение са извършени серия експерименти с разпространение на информацията в изкуствена невронна мрежа при правия пас. Алгоритъмът за правия пас е кодиран в два отделни модула на системата VitoshaTrade - съответно модул на C++ и модул на JavaScript. Както е видно от Фиг. 4.3, бързодействието е съизмеримо за мрежи от 10 до 100 неврона. Всеки експеримент е изпълнен 30 пъти и средните стойности са представени на фигурата [BAKE2016]. Като експериментални данни са използвани котировките EUR/USD, при дневен интервал, за периода 01.01.2015 до 31.03.2015 (Приложение А).



Фиг. 4.4 *Стандартно отклонение по време при разпространение на информацията в изкуствена невронна мрежа, по време на правия пас, между C++ и JavaScript. По оста X е броят неврони, а по оста Y е стандартното отклонение.*

По отношение на бързодействието езиците C++ и JavaScript са съизмерими, но устойчивостта на изчислителния процес е по-добре при C++, което е видно на Фиг. 4.4, която отразява средното квадратично отклонение на времето нужно за изчисление. Тази разлика се дължи основно на наличието на интерпретатор и уеб браузър, нещо което не присъства при изчисленията в C++. При езици от категорията на C++ програмата първоначално се транслира до асемблерен език, а след това и до машинен код.

Реализацията на изчисления в разпределена среда, под формата на JavaScript-AJAX уеб базирана система води до още по-висока степен на разширяемост в системата спрямо C/C++ варианта. Практически, разпределените изчисления могат да се стартират на всяко устройство поддържащо съвременен уеб браузър способен да стартира JavaScript и AJAX. Тъй като изчислението се извършва в рамките на уеб браузъра, който от своя страна е процес в адресното пространство на операционната система, а тя от своя страна се изпълнява на физическия хардуер, макар и съизмерими по бързодействие, изчисленията са по-ненадеждни (наличие на интерпретатор) отколкото биха били при реализация на езици като C/C++ или

Assembler.

4.3 Сравнение между различни топологии на изкуствени неверонни мрежи

При прогнозирането на времеви редове с изкуствени невронни мрежи двата най-актуални модела мрежи са класическа трислойна мрежа, без обратни връзки и мрежа с обратни връзки. Проведените експерименти са с трислойни мрежи, които имат различен размер на скрития слой и пълно свързани мрежи (всеки възел е свързан с всеки останал, включително и със себе си). При пълната свързаност се цели ефект на кратковременна памет в мрежата. Като експериментални данни са използвани собствени данни за брой изминати крачки на ден (Приложение А). Тестовете на топологиите са реализирани в изолиран софтуерен проект поради сложността на експерименталната постановка.

4.3.1 Сравнение между пълно-свързана и трислойна изкуствена невронна мрежа

За сравнението между пълно-свързана и трислойна мрежа е избрана трислойна топология 28-12-5. Пълно свързаната мрежа се състои от 45 възела. И двата вида мрежи са обучени с генетични алгоритми при следните параметри за генетичния алгоритъм:

Таб. 4.1 *Параметри на генетичния алгоритъм.*

Размер на популацията:	37
Вероятност за кръстосване:	0.9
Вероятност за мутация:	0.03
Дял елитни елементи:	0.1
Аргументи за състезателна селекция:	2
Вид кръстосване:	бинарно
Вид мутация:	случаен делта вектор

Условие за край:	24 часа

И двете мрежи са обучени, при равни други условя. Условието за край на обучението е изтичането на фиксиран период от време, а именно 24 часа. В резултат на обучението пълно-свързаната мрежа показва по-добри параметри за по-малко епохи обучение:

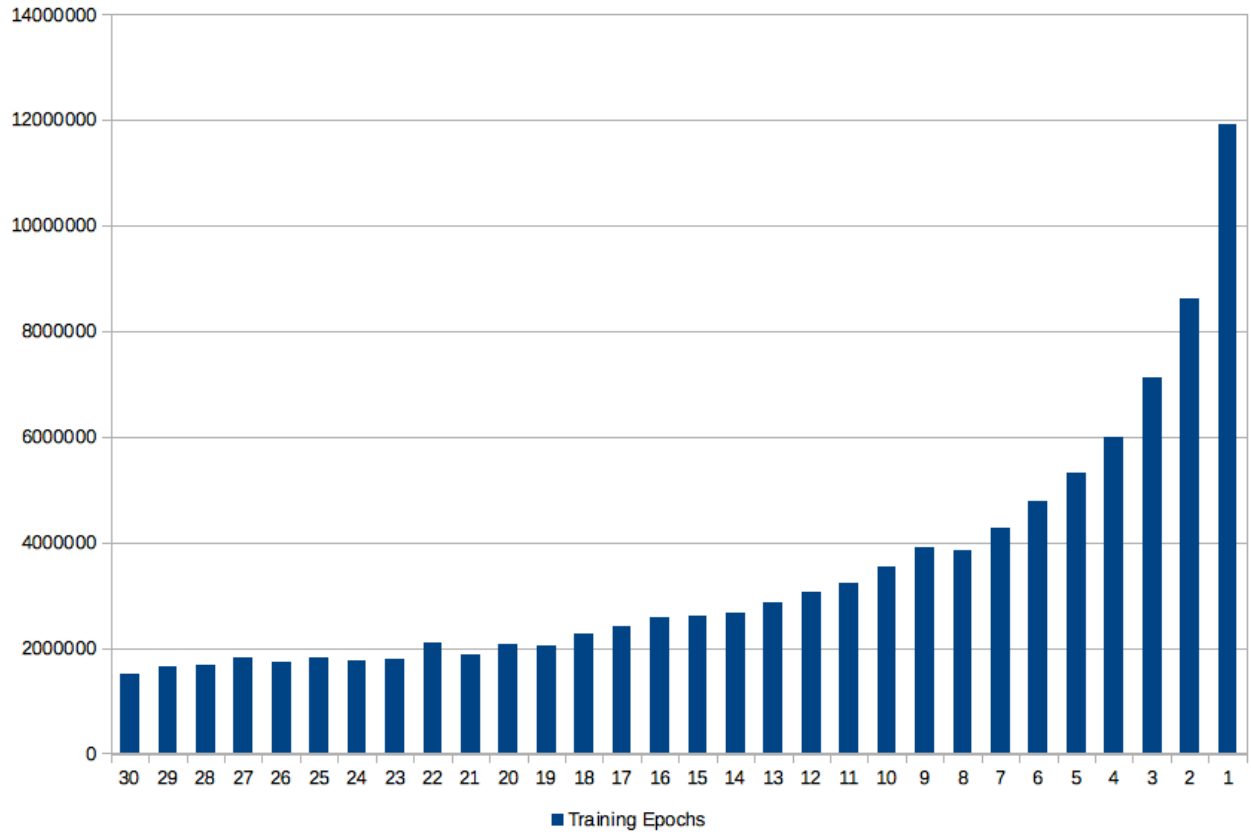
Таб. 4.2 *Сравнение между трислойна и пълно-свързана изкуствена невронна мрежа.*

Вид мрежа:	Трислойна	Пълно-свързана
Брой епохи:	296037	294003
Жизненост на най-добрия индивид в популацията:	656.1699030329493	656.169903010489
Грешка на ИНМ при тестовото множество:	317.7286234453099	317.0226357675591

Наличието на повече тегла при една пълно-свързана мрежа води до известно забавяне на обучението, но наличието на обратни връзки позволява формирането на кратковременна памет в изкуствените невронни мрежи и при по-дълго обучение това дава предимство на този модел.

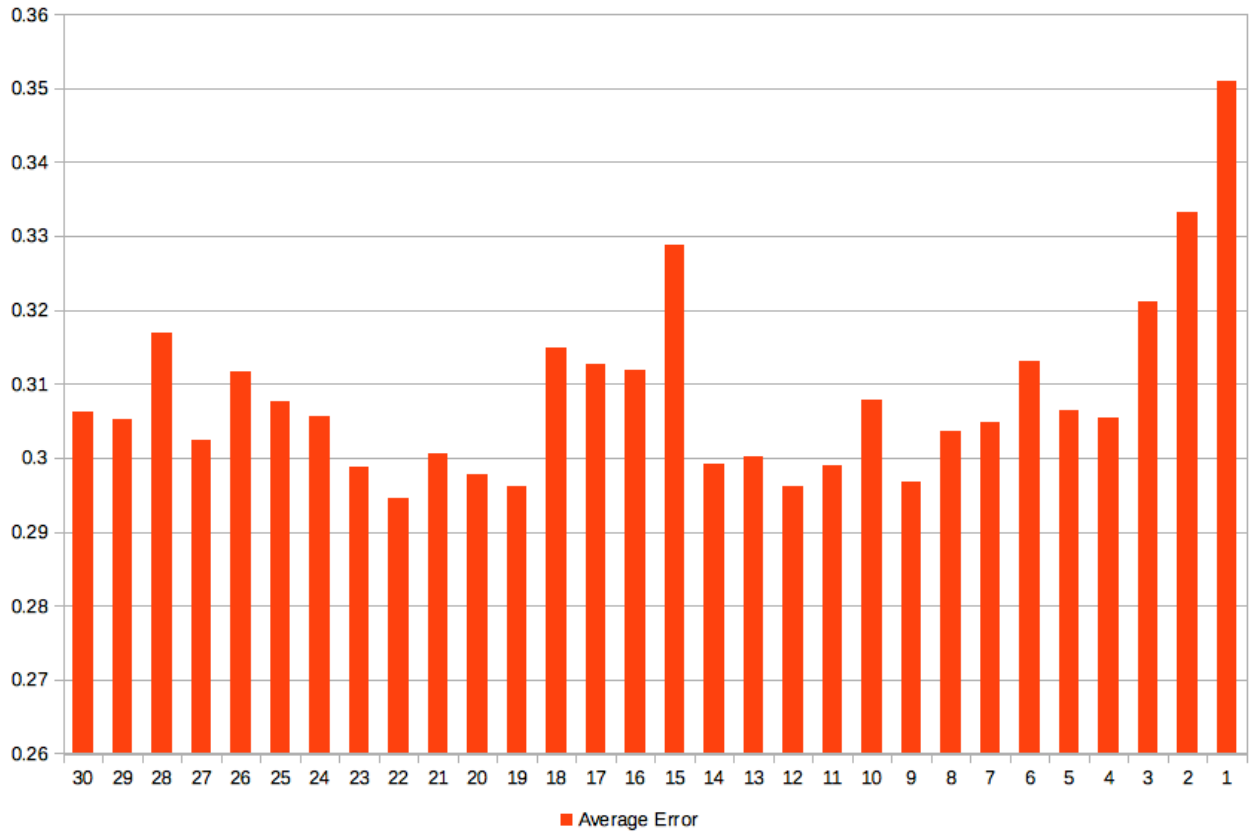
4.3.2 Сравнение между трислойни изкуствени невронни мрежи с различен размер в скрития слой

Проведените експерименти са извършени с трислойни мрежи, като на входа се подават 25 стойности от изминал период, а на изхода се очакват 5 стойности в бъдещ период. За целите на експеримента данните условно са разделени в три под множества: 1. Данни за обучение; 2. Данни за валидация; 3. Данни за тестване. Изпробвани са топологии с размер на скрития слой от 30 възела до 1 възел. Всички експерименти са извършени при еднакъв интервал от врем. Фиг. 4.5 показва разликата в броя епохи, спрямо размера на скрития слой.



Фиг. 4.5 *Брой епохи за обучение, при еднакъв времеви интервал, за обучението, но различен размер на скрития слой.*

В експериментите е оценена средната грешка, която допускат мрежите с различна топология. Целта е да се установи при кои стойности за размера на скрития слой мрежата е най-ефективна. Изчисляването на грешката е извършено само с използване на данните от подмножеството за тестване. Това са данни, които не са използвани в процеса за обучение и валидация.

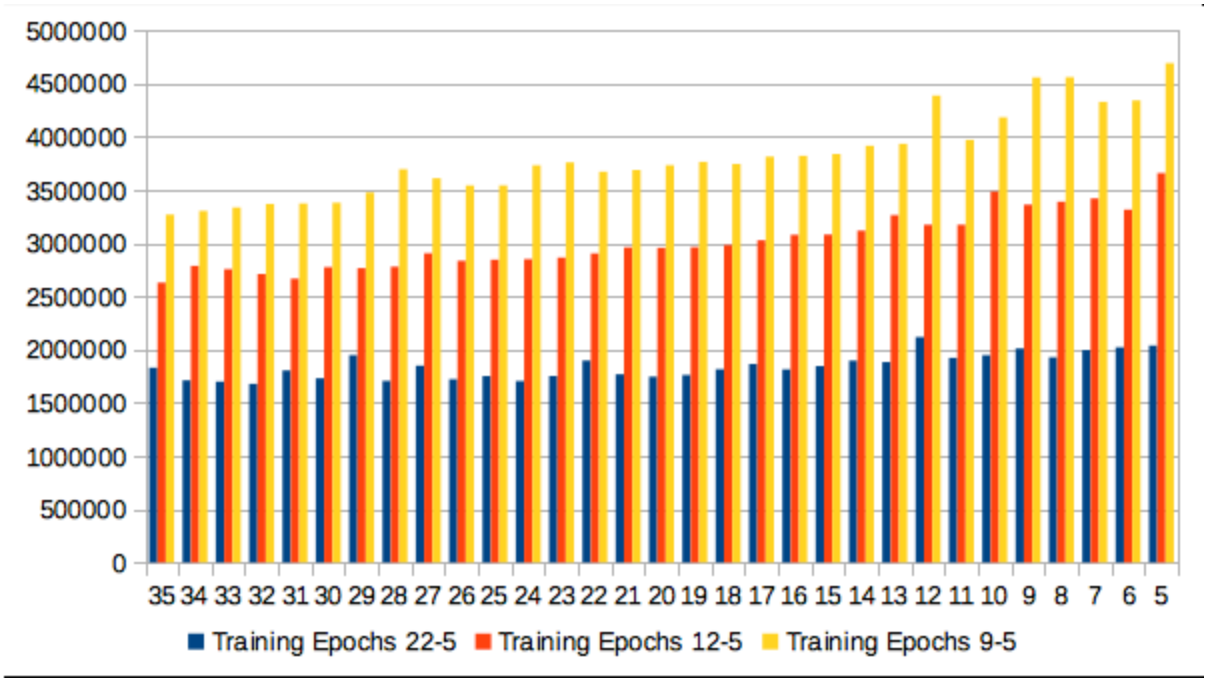


Фиг. 4.6 *Средна грешка, допускана от изкуствената невронна мрежа, при обучение на мрежи с топология от три слоя, в които размерът на скрития слой варира от 30 до 1.*

От Фиг. 4.6 ясно се вижда, че най-перспективни са моделите с 22, 12 и 9 елемента в скрития слой което съответства на топологии мрежи: 1. 25-22-5; 2. 25-12-5; 3. 25-9-5.

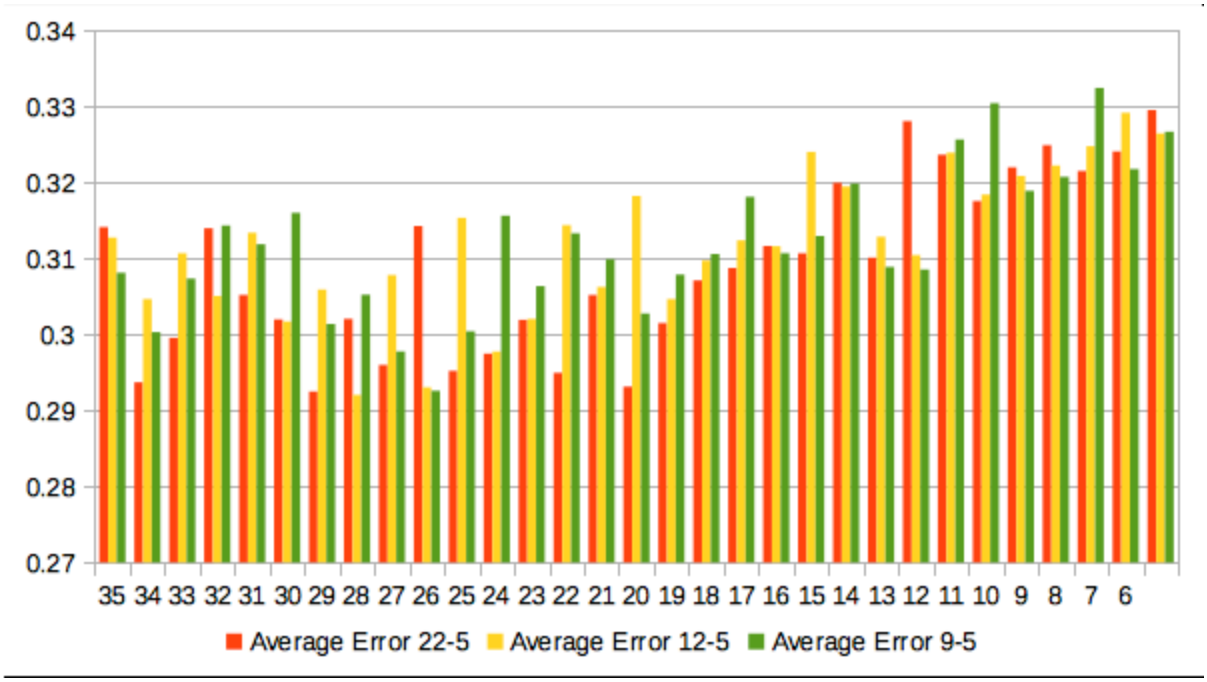
4.4 Сравнение за ефективност на прогнозирането при различен прозорец от входни данни

За трите най-ефективни модела, определени на Фиг. 4.6, са извършени допълнителни експерименти с изменение на размера който има времевия прозорец от отминалите периоди. Размерът на времевия прозорец е изследван от 35 до 5 стойности като прозореца за прогнозиране на бъдещите резултати е фиксиран само на 5 стойности.



Фиг. 4.7 *Брой епохи за обучение, при еднакъв времеви интервал, за обучението, но различен размер на входния слой (прозорец от данни в миналото).*

От Фиг. 4.7 ясно се вижда, че размерът на входния слой почти не оказва влияние в броя епохи през които протича обучението за фиксиран интервал от време (в случая 1 час). Също така ясно се забелязва, че размерът на скрития слой оказва пряко влияние на броя епохи за обучение при фиксиран интервал от време. Най-бавно протича обучението при скрит слой от 22 елемента, а най-бърз протича обучението при скрит слой от 9 елемента.



Фиг. 4.8 *Средна грешка, допускана от изкуствената неверонна мрежа, при обучение на мрежи с топология от три слоя, в които размерът на входния слой варира от 35 до 5.*

На Фиг. 4.8 ясно се вижда при кои стойности на входния слой изкуствената невронна мрежа е постигнала най-малка грешка за предложената прогноза. За размер на скрития слой от 22 елемента най-ефективният входен слой е 29 елемента. При скрит слой от 12 елемента, най-ефективният входен слой е 28 елемента. И при скрит слой от 9 елемента, най-ефективният входен слой е 26 елемента. От трите възможности най-добри резултати се получават при скрит слой от 12 елемента, което е около половината от входния слой.

4.5 Обобщение

Като кратко обобщение може да се отбележи, че след извършените експерименти за обучение на изкуствени невронни мрежи в разпределена среда има подобрена ефективност когато обучението протича паралелно и е базирано на ефективна миграция между възлите.

В резултат на научноизследователската работа изложена в тази глава са постигнати следните приноси:

1. Представени са резултати за подобрено бързодействие на обучението при използване на алгоритъм за инцидентно включване на възли;

2. Представени са резултати за съизмеримо бързодействие при обучението на изкуствени невронни мрежи с пълна свързаност на невроните и трислойни изкуствени невронни мрежи;

Резултатите от проведените експерименти са представени на „XXII Международен симпозиум Управление на енергийни, индустриални и екологични системи“, „XXIV Международен симпозиум Управление на топло енергийни обекти и системи, Управление на енергийни, индустриални и екологични системи“ и International Conference on Large-Scale Scientific Computations, а публикациите са цитирани в дисертационния труд като референции [BATO2015], [BAZA2015] и [BAGE2016a].

Заключение

В дисертационната работа са представени основни резултати получени от проведените изследвания в областта на методите за прогнозиране на времеви редове, алгоритми и програмни системи за решаване на задачи в областта на прогнозирането. Целта на тези изследвания е разработката на евристичните подходи за машинно самообучение разширяващи възможностите за прогнозиране на времеви редове. На основата на тези евристични подходи са предложени евристични методи и алгоритми подобряващи бързодействието в машинното самообучение на изкуствени невронни мрежи. Ускоряването на бързодействието в обучението на изкуствени невронни мрежи е особено важно при прогнозирането на времеви редове, тъй като фазата на обучение е значително по-бавна отколкото фазата на самото прогнозиране. Характеристиките на тези методи и алгоритми, качеството на предложения дизайн както и добрата програмна реализация са в основата за създаването на една ефективна програмна система за прогнозиране на времеви редове в сферата на валутните пазари.

Основните резултати, получени при разработката на дисертационната работа, са докладвани на научни семинари в ИИКТ-БАН, както и на 9 специализирани конференции (някои от които с международно участие) по машинно самообучение.

* International Conference AUTOMATICS AND INFORMATICS, Sofia, Bulgaria, 2016

* International Conference InfoTech16, Varna - St. St. Constantine and Elena, Bulgaria. 2016

* XXIV Международен симпозиум по управление на топло енергийни обекти и системи, Управление на енергийни, индустриални и екологични системи, Банкя, България, 2016

* International Conference on Large-Scale Scientific Computing, Sozopol, Bulgaria, 2015

* International Scientific Conference UniTech15, Gabrovo, Bulgaria, 2015

* XXIII Международен симпозиум по управление на енергийни, индустриални и екологични системи, Банкя, България, 2015

* Първа национална тематична школа и борса за научни идеи в областта на

информационните и комуникационни технологии, Русенски университет "Ангел Кънчев", Русе, България, 2013

* International Conference on Large-Scale Scientific Computing, Sozopol, Bulgaria, 2011

* Anniversary Scientific Conference 40 Years Department of Industrial Automation, Sofia, Bulgaria, 2011

Изследванията от дисертационния труд са отразени в 8 публикации, 2 от които са самостоятелни. За самостоятелните публикации не са забелязани цитирания. За публикациите в съавторство са забелязани 4 цитирания.

Част от получените резултати са включени в отчетите на 2 научно-изследователски теми от вътрешния план на ИИТ-БАН и ИИКТ-БАН (приемник на ИИТ):

* Изследване, разработка и приложение на ефективни методи за трудно решими едно- и много критериални задачи. По договор с Институт по информационни и комуникационни технологии - БАН, 2010-2011

* Методи и системи за оптимизация. По договор с Института по информационни технологии - БАН. 2007-2009

Друга част от резултатите са отразени в отчетите на съвместните научно-изследователски проекти между ИИКТ-БАН, Фонд научни изследвания и Европейския социален фонд:

* ДФНИ И02/20 Ефективни паралелни алгоритми за големи изчислителни задачи, 2015-2016

* ДФНИ И02/5 Интеркритериален анализ - нов подход за вземане на решения, 2015-2016

* BG051PO001-3.3.06-00 Изграждане и развитие на млади висококвалифицирани изследователи за ефективно прилагане на биомедицинските изследвания за подобряване качеството на живот, Оперативна програма „Развитие на човешките ресурси” 2007-2013, МОНМ и Европейския социален фонд на Европейския съюз. 2012-2015

* BG051PO001-3.3.06-0048 по схемата за предоставяне на безвъзмездна финансова

помощ “Подкрепа за развитието на докторанти, постдокторанти, специализанти и млади учени”

* BG051PO001-3.3.04/40: Изграждане на висококвалифицирани млади изследователи по съвременни информационни технологии за оптимизация, разпознаване на образи и подпомагане вземането на решения. 2009-2011

* ДТК 02/71 УЕБ-базирана интерактивна система,подпомагаща построяването на модели и решаването на задачи за оптимизация и вземане на решения, 2009-2013

* МУ02-63-МИ-09-000 Консолидиране и обработване на параметри за пожари в горски масиви на територията на България чрез прилагане на модела WEATHER RESEARCH AND FORECASTING MODEL – FIRE (WRF-FIRE), 2010-2011

Приноси

Научни приноси

1. Предложен е евристичен подход за обучение на пълно свързани невронни мрежи в разпределена среда, базиран на инцидентно включване на възли;

2. Предложен е евристичен подход за обучение на пълно свързани невронни мрежи в разпределена среда, базиран на диференциална еволюция;

3. Разработен е метод за машинно обучение на пълно свързани невронни мрежи в разпределена среда, базиран на предложения евристичен подход за обучение с инцидентно включване на възли;

4. Разработен е метод за машинно обучение на пълно свързани невронни мрежи в разпределена среда, базиран на предложения евристичен подход за обучение с диференциална еволюция;

Научно-приложни приноси

5. Предложени са алгоритми за прогнозиране на времеви редове с използване на пълно

свързани невронни мрежи, обучавани с предложения евристичен подход с инцидентно включване на възли;

6. Предложени са алгоритми за прогнозиране на времеви редове с използване на пълно свързани невронни мрежи, обучавани с предложения евристичен подход с диференциална еволюция;

Приложни приноси

7. Изследвани са и са експериментално оценени предложените подходи, методи и алгоритми за прогнозиране на времеви редове с данни от валутни пазари.

Планове за бъдеща разработка

Постигнатите резултати в дисертационната работа очертават следните насоки за бъдещи изследвания:

* Да се изследват предложените евристични подходи за търсене на прогнози в други области, а не само по отношение на валутните пазари.

* Да се разшири програмната система, като се включат и други методи за прогнозиране на времеви редове, с цел по-ефективен сравнителен анализ.

* Да се разширят възможностите на уеб базирания интерфейс в системата и дори осъществяване на изчисления в разпределена среда, базиращи се на пресмятания върху мобилни устройства [BAGE2016b].

Благодарности

Благодаря на колегите от ИИТ-БАН и ИИКТ-БАН, които бяха редом до мен през годините и спомогнаха за моята работа.

Бих искал да благодаря на трима човека, моето семейство, без подкрепата на които тази работа не би била възможна. На първо място, това са моите родители Здравка и Димитър. Без всичките техни усилия и десетките години на безкористна подкрепа аз не бих имал шансовете да постигна всичко, което съм постигнал в живота. На второ място, това е моят брат Лъчезар, който е единственият човек, на когото съм разчитал в трудни моменти.

Смятам и трима ви за част от този труд.

Благодаря!

Декларация за оригиналност на резултатите

Декларирам, че настоящата дисертация съдържа оригинални резултати, получени при проведени от мен научни изследвания (с подкрепата и съдействието на консултанта и научния ми ръководител). Резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани в библиографията.

Настоящата дисертация не е прилагана за придобиване на научна степен в друго висше училище, университет или научен институт.

Подпис:

Тодор Димитров Балабанов

Библиография

1. [JDXL2011] J. P. Donate, X. Li, G. G. Sanchez, A. S. de Miguel, Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm, Neural Comput & Applic, DOI 10.1007/s00521-011-0741-0, Springer-Verlag London Limited 2011, 14 Oct 2011.

2. [JPGG2010a] J. Peralta, G. Gutierrez, A. Sanchis, Time series forecasting by evolving artificial neural networks using genetic algorithms and estimation of distribution algorithms, In Proceedings of IJCNN. 2010, 1-8.

3. [PCJM1996a] P. Cortez, J, Machado, J. Neves, An Evolutionary Artificial Neural Network Time Series Forecasting System, International Conference on Artificial Intelligence, Expert Systems and Neural Networks, Honolulu, Hawaii, 1996.

4. [AMFO1999] A. Foka, Time Series Prediction Using Evolving Polynomial Neural Networks, A dissertation submitted to the University of Manchester Institute of Science and Technology for the degree of MSc, 1999.

5. [VBGM2006] V. Bevilacqua, G. Mastronardi, F. Menolascina, A. Paradiso, S. Tommasi, Genetic Algorithms and Artificial Neural Networks in Microarray Data Analysis: a Distributed Approach, Engineering Letters - Special Issue on Bioinformatics - ISSN: 1816-0948 13: 3. 335-343, 04 Nov 2006.

6. [VLOL2004] V. Olej, Distributed Genetic and Eugenic Algorithms for Prediction of Gross Domestic Product Development by Frontal Neural Network, Proc. of the IEEE 4-th International Conference on Intelligent Systems Design and Applications, ISDA 04, Budapest, Hungary, 2004, 7-13.

7. [MASE2005] M. Setati, Machine Learning for Decision-Support in Distributed Networks, A dissertation submitted to the Faculty of Engineering and the Build Environment, University of the

Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science, Dec 2005.

8. [NMMS2008] N. Mirarmandehi, M. Saboorian, A. Ghodrati, Time-Series Prediction using Neural Networks, http://ce.sharif.edu/~armandeh/Publish/ANN/TS.pdf.

9. [PCJM1996b] P. Cortez, J. Machado, J. Neves, Time Series Forecasting in a Distributed Environment, Ibero-American Conference On Artificial Intelligence (IBERAMIA 96), 5, Cholula, Puebla, Mexico, 1996. [S.l.] : Limusa Noriega Editores, 1996. p. 68-77.

10. [IBAT2012] Ismail B, Ataulla, M. Yunus, Time Series Forecasting Using Undecimated Wavelets, Neural Networks and Genetic Algorithm, International Journal of Electronics and Computer Science Engineering, Vol. 1, Num. 3, ISSN- 2277-1956, p. 1404-1415, 2012.

11. [RSQZ2006] R. Segall, Q. Zhang, Applications of Neural Network and Genetic Algorithm Data Mining Techniques in Bioinformatics Knowledge Discovery - A Preliminary Study, Proceedings of the Thirty-seventh Annual Conference of the Southwest Decision Sciences Institute, p. 278-285, 2006.

12. [CDTY2006] C. Dawson, T. Young, J. Allen, Machine Intelligence Genetic Algorithm Neural Network System for the Engineered Wood Industry, USDA SBIR Phase-I Final Report, Jul 2006.

13. [MVRB2004] M. Versace, R. Bhatt, O. Hinds, M. Shiffer, Predicting the exchange traded fund DIA with a combination of genetic algorithms and neural networks, Expert Systems with Applications 27 (2004) 417–425, p. 417-425.

14. [SOMF1992] S. Oliker, M. Furst, O. Maimon, A Distributed Genetic Algorithm for Neural Network Design and Training, Complex Systems 6 (1992) 459-477, p. 459-477.

15. [ITDG2008a] I. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing, 72(1-3) 269-277.

16. [RAVA2012] R. Abiyev, V. Abiyev, Differential Evaluation Learning of Fuzzy Wavelet Neural Networks for Stock Price Prediction, Journal of Information and Computing Science, Vol. 7, No. 2, p.121-130, 2012.

17. [DWFG2005] D. Wierstra, F. Gomez, J. Schmidhuber, Modeling Systems with Internal State using Evolino, In Proc. of the 2005 conference on genetic and evolutionary computation (GECCO), Washington, D. C., pp. 1795-1802, ACM Press, New York, NY, USA, 2005.

18. [MKMB2010] M. Khashei, M. Bijari, An artificial neural network (p, d,q) model for timeseries forecasting, Expert Systems with Applications: An International Journal Vol. 37 Issue 1, January, p. 479-489, 2010.

19. [ANWE1997] A. Weigend, Data Mining in Finance: Report From The Post-Nncm-96 Workshop on Teaching Computer Intensive Methods for Financial Modeling and Data Analysis, Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96), pp. 399-412, 1997.

20. [LMRB2009] L. Maciel, R. Ballini, Design a Neural Network for Time Series Financial Forecasting: Accuracy and Robustness Analisys, XXIII International Congress of Applied Economics - ASEPELT'09,Covilhp, Portugal, Jun 2009.

21. [ITDG2008b] I. Tsoulos, D. Gavrilis, E. Glavas, Neural Network Construction and Training using Grammatical Evolution, Neurocomputing Vol. 72 Issue 1-3, p. 269-277, Dec 2008 .

22. [KKMM1999] KW. Ku, MW. Mak, WC. Siu, Adding Learning to Cellular Genetic Algorithms for Training Recurrent Neural Networks, IEEE Trans Neural Netw. 10(2), p.239-252, 1999.

23. [DMLD1989] D. Montana, L. Davis, Training Feedforward Neural Networks Using Genetic Algorithms, IJCAI'89 Proceedings of the 11th international joint conference on Artificial intelligence - Vol. 1, p.762-767, 1989.

24. [EFKA2001] E. Kalyvas, Using Neural Networks and Genetic Algorithms to Predict Stock

Market Returns, A Thesis Submitted to the University of Manchester for The Degree of Master of Science in Advanced Computer Science in the Faculty of Science and Engineering, Oct 2001.

25. [SCYW1999] S. Chen, Y. Wu, B. L. Luk, Combined Genetic Algorithm Optimization and Regularized Orthogonal Least Squares Learning for Radial Basis Function Networks, IEEE Transactions on Neural Networks, Vol. 10, No. 5, p.1239-1243, Sep 1999.

26. [RSBA1998] R. Sexton, B. Alidaee, R. Dorsey, J. Johnson, Global Optimization for Artificial Neural Networks: A Tabu Search Application, European Journal of Operational Research, Vol. 106, Num. 2, pp. 570-584(15), 16 Apr 1998.

27. [BHKT2005] B. Hu and KW. Tsui, Distributed Evolutionary Monte Carlo with Applications to Bayesian Analysis, University of Wisconsin - Madison, Technical Report NO. 1112, Nov 10, 2005.

28. [ERHU2004] E. Hulthen, Improving Time Series Prediction Using Recurrent Neural Networks and Evolutionary Algorithms, Thesis For The Degree of Master of Science, Division of Mechatronics ,Department of Machine and Vehicle Systems, Chalmers University of Technology, Goteborg, Sweden, 2004.

29. [ZPLL2006] Z. Ping, L. Ling-yan, S. Hao-shan, A Hybrid Location Algorithm Based on BP Neural Networks for Mobile Position Estimation, IJCSNS International Journal of Computer Science and Network Security, Vol.6 No.7A, p.162-167, Jul 2006.

30. [JARG2001] J. Arifovica , R. Gencay, Using genetic algorithms to select architecture of a feedforward articial neural network, Physica A, Vol. 289, Issues 3-4, p.574-594, 15 Jan 2001.

31. [CHTA2009] C. Tan, Financial Time Series Forecasting Using Improved Wavelet Neural Network, Master's Thesis, promoter Prof. Christian Norgaard Storm Pedersen, Bioinformatics Research Center, Aarhus University, Denmark, 31 May 2009.

32. [LESP2006] L. Ekonomou SSP. Pappas, Implementation of Artificial Intelligence in the Time Series Prediction Problem, Proceedings of the 6th WSEAS International Conference on Simulation,

Modelling and Optimization, Lisbon, Portugal, September 22-24, 2006.

33. [JPGG2010b] J. Peralta, G. Gutierrez, A. Sanchis, Automatic Design of Artificial Neural Networks to Forecast Time Series, Actas del V Simposio de Teoria y Aplicaciones de Mineria de Datos, III Congreso Espanol de Informatica. CEDI, ISBN 978-84-92812-62-2, 2010.

34. [JORI1991] J. Riordan, A Neurla Network Based Distributed System for Forecasting the Stock Market, Subbmited to the Department of Electrical Engineering and Computer Science for the degree of Bachelor of Science in Computer Science and Engineering at the Massachusetts Institute of Technology, May 1991.

35. [ASIC2002] A. Skabar, I. Cloete , Neural Networks, Financial Trading and the Efficient Markets Hypothesis, In ACSC '02 Proceedings of the twenty-fifth Australasian conference on Computer science - Vol. 4, p.241-249, Australia 2002.

36. [RSNS2001] R. Sexton, N. Sikander, Data Mining Using a Genetic Algorithm Trained Neural Network, Int. Syst. in Accounting, Finance and Management, p.201-210, 2001.

37. [JRVC1998] J. Riley, V. Ciesielski, An Evolutionary Approach to Training Feed-Forward and Recurrent Neural Networks, In Proceedings of the Second International Conference on Knowledge Based Intelligent Electronic Systems (KES'98), p.596-602, 1998.

38. [ARSH2003] A. Shapiro , Capital Market Applications of Neural Networks, Fuzzy Logic and Genetic Algorithms, 13th International AFIR Colloquium, Vol. 1, p.493-514, 2003.

39. [MDMP2006] M. Delgado, MC. Pegalajar, MP. Cuellar, Evolutionary Training for Dynamical Recurrent Neural Networks: An Application in Finantial Time Series Prediction, Mathware & Soft Computing, Vol. 13, Num. 2, p.89-110, 2006.

40. [JMAP2011] JJ. Moreno, A. Pol, P. Gracia, Artifi cial neural networks applied to forecasting time series, Psicothema, ISSN: 0214-9915, Vol. 23, Num. 2, pp. 322-329, 2011.

41. [TGTT2009] T. Glezakos, T. Tsiligiridis, L. Iliadis, C. Yialouris, F. Maris, K. Ferentinos, Feature Extraction for Time Series Data: an Artificial Neural Network Evolutionary Training Model for the Management of Mountainous Watersheds, Neurocomputing, Vol. 73, Issues 1-3, p.49–59, Dec 2009.

42. [CHSL1990] C. Slim, A genetic Neural Network approach applied to Forecast the volatility of financial time series, CiteSeerX, 10.1.1.89.7183.

43. [KYKI2006] K. Kim, Artificial neural networks with evolutionary instance selection for financial forecasting, Expert Systems with Applications, Vol .30, Issue 3, p.519–526, Apr 2006.

44. [HAMN2002] H. Alfares, M. Nazeeruddin, Electric load forecasting: literature survey and classi®cation of methods, International Journal of Systems Science, vol. 33, Num. 1, p.23-34, 2002.

45. [MAMA1995] M. Mandischer, Evolving Recurrent Neural Networks with Non-binary Encoding, In Proceedings of the 2nd IEEE Conference on Evolutionary Computation (ICEC), p.584-589, 1995.

46. [LSSO1999] L. See, S. Openshaw, Applying soft computing approaches to river level forecasting, Hydrological Sciences-Journal-des Sciences Hydrologiques, 44(5), p.763-778, Oct 1999.

47. [SKYS2010] S. Khmylovyy, Y. Skobtsov, Feature Selection for Time-series Prediction in Case of Nondetermined Estimation, In Proceedings of Donetsk National Technical University, Num. 1, p.63-74, 2010.

48. [DEDH2002] D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, R. Porter, J. Theiler, Genetic Algorithms and Support Vector Machines for Time Series Classication, In Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 4787, pp. 74–85, 2002.

49. [MOAW2010] M. Awad, Optimization RBFNNs Parameters Using Genetic Algorithms: Applied on Function Approximation, International Journal of Computer Science and Security, Vol.4, Issue 3,

p.295-307, 2010.

50. [MZPL2004] M. Zhong, P. Lingras, S. Sharma, Estimation of missing traffic counts using factor, genetic, neural, and regression techniques, Transportation Research Part C: Emerging Technologies, No. 12, p.139–166, 2004.

51. [ANBG2011] A. Nor, B. Gharleghi, K. Omar, T. Sarmidi, A Novel Artificial Neural Network Model for Exchange Rate Forecasting, In Proceeding of 2nd International Conference on Business and Economic Research (2nd ICBER 2011), p.99-108, 2011.

52. [RSRD1998] R. Sexton, R. Dorsey, J. Johnson, Toward Global Optimization of Neural Networks: A Comparison of the Genetic Algorithm and Backpropagation, Decision Support Systems, Vol. 22, No. 2, p.171-185, 1998.

53. [BZHM1993] BT. Zhang, H. Muhlenbein, Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor, Complex Systems, Vol. 7, Num. 3, p.199-220, 1993.

54. [IDAD1998] I. De Falco, A. Della Cioppa, A. Iazzetta, P. Natale, E. Tarantino, Optimizing Neural Networks for Time Series Prediction, In Procedding of Third World Conference on Soft Computing (WSC3): Engineering Design and Manifacturing, Jul 1998.

55. [RSRD1999] R. Sexton, R. Dorsey, J. Johnson, Optimization of Neural Networks: A Comparative Analysis of the Genetic Algorithm and Simulated Annealing, European Journal of Operational Research, Vol. 114, Issue 3, p.589-601, 1999.

56. [ARGH2012] A. Ghosh, Comparative study of Financial Time Series Prediction By Artificial Neural Network with Gradient Descent Learning, International Journal Of Scientific & Engineering Research, Vol. 3 Issue 1, Jan 2012.

57. [SSST2006] S. Sirakaya, S. Turnovsky, M. Alemdar, Feedback Approximation of the Stochastic Growth Model by Genetic Neural Networks, Computational Economics, Vol. 27, p.185-206, 2006.

58. [SMGM1996] S. Mahfoud, G. Mani, Financial Forecasting Using Genetic Algorithms, Applied Artificial Intelligence, Vol. 10, p.543-565, 1996.

59. [MHRK2009] M. Hlavacek, R. Kalous, F. Hakl, Neural network with cooperative switching units with application to time series forecasting, Computer Science and Information Engineering, 2009 WRI World Congress on Computer Science and Information Engineering, Vol. 5, p.676-682, 2009.

60. [HNTH2004] H. Niska, T. Hiltunen, A. Karppinen, J. Ruuskanen, M. Kolehmainen, Evolving the neural network model for forecasting air pollution time series, Engineering Applications of Artificial Intelligence, Vol. 17, Num. 2, p.159–167, 2004.

61. [MEVO2011] M. Vora, Genetic Algorithm for Trading Signal Generation: Solution to trader's dilemma: Is it right time to trade?, 2010 International Conference on Business and Economics Research, Vol.1,IACSIT Press, Kuala Lumpur, Malaysia, 2011.

62. [RPSL2005] R. Patuelli, S. Longhi, A. Reggiani, P. Nijkamp, Forecasting Regional Employment in Germany by Means of Neural Networks and Genetic Algorithms, Computational Economics 0511002, EconWPA, 2005.

63. [KMKC2000] KZ. Mao, KC. Tan, W. Ser, Probabilistic Neural-Network Structure Determination for Pattern Classification, IEEE Transactions on Neural Networks, Vol. 11, No. 4, Jul 2000.

64. [AYGU2009] A. Guven, Linear genetic programming for time-series modelling of daily flow rate, Journal of Earth System Science, Vol.118, No. 2, p.137–146, Apr 2009.

65. [SVEK2003] S. Eklund, Time Series Forecasting using Massively Parallel Genetic Programming, International Parallel and Distributed Processing Symposium (IPDPS'03), pp.143a, 2003.

66. [BCJB2003] B. Chramcov, J. Balati, Prediction of the Heat Supply Daily Diagram via Artificial

Neural Network, In Proceedings of the 4th International Carpathian Control Conference ICCC 2003, p.703-706, 2003.

67. [LHBB2002] L. Hamm, B. Brorsen, Global Optimization Methods, Paper presented at the Western Agricultural Economics Association Annual Meetings, Long Beach, CA, Jul 2002.

68. [ABEB2012] A. Barbulescu, E. Bautu, A Hybrid Approach for Modeling Financial Time Series, The International Arab Journal of Information Technology, Vol. 9, No. 4, p.327-335, Jul 2012.

69. [PDAA2006] P. Doganis, A. Alexandridis, P. Patrinos, H. Sarimveis, Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing, Journal of Food Engineering, Vol. 75(2), p.196–204, 2006.

70. [ERFO1993] E. Foster, Commodity Futures Price Prediction, an Artificial Intelligence Approach, A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree Master of Science Athens, Georgia.

71. [JVGM2002] J. Valdes, G. Mateescu, Time Series Model Mining with Similarity-Based Neuro-Fuzzy Networks and Genetic Algorithms: A Parallel Implementation, In Proceedings of the RSCTC'02, Special Session on Distributed andCollaborative Data Mining, Pennsylvania. October 2002. NRC 44933.

72. [JSKU1998] J. Kutsurelis, Forecasting Financial Markets Using Neural Networks: An Analysis of Methods and Accuracy, Submitted in partial fulfillment of the requirements for the degree of Master of Science in Management from the Naval Postgraduate School, Sep 1998.

73. [SMSK2001] SW. Moon, SG. Kong, Block-Based Neural Networks, IEEE Transactions On Neural Networks, Vol. 12, No. 2, Mar 2001.

74. [RSRD2000] R. Sexton, R. Dorsey, Reliable classification using neural networks: a genetic algorithm and backpropagation comparison, Decision Support Systems, Vol. 30, p.11–22, 2000.

75. [MUAB2005] M. Abdella, The Use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database, A research report submitted to the Faculty of Engineering and Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering by coursework and research report, Jan 2005.

76. [RPTL2002] R. Prudencio, T. Ludermir, Neural Network Hybrid Learning: Genetic Algorithms & Levenberg-Marquardt, In Proceedings of Between Data Science And Applied Data Analysis, Part III, p.464, 2002.

77. [YAHU2009] Y. Huang, Advances in Artificial Neural Networks – Methodological Development and Application, Algorithms, Vol.2, No.3, p.973-1007, 2009.

78. [EUVO2005] E. Vogel, Complex Indicators: Nonlinear Pricing and Reflexivity, Department of Informatics, St. Petersburg State University of Economics and Finance, 191023, Saint-Petersburg, Russia, 2005.

79. [IVTL2011] I. Valenca, T. Lucas, T. Ludermir, Selecting Variables With Search Algorithms and Neural Networks to Improve the Process of Time Series Forecasting, International Journal of Hybrid Intelligent Systems - Feature and algorithm selection with Hybrid Intelligent Techniques, Vol. 8, Issue 3, p.129-141, Aug 2011.

80. [YCBY2005] Y. Chen, B. Yang, J. Dong, A. Abraham, Time-series forecasting using flexible neural tree model, Information Sciences: an International Journal, Vol. 174, Issue 3-4, p. 219-235, 11 Aug 2005.

81. [PGSC2007] P. Gautham, S. Chandrasekar, O. Piyush, Artificial Neural Network Based Short Term Load Forecasting for the Distribution Utilities, National Conference on Distribution Automation, CPRI, Bangalore, 2007.

82. [BLLE1997] B. LeBaron, An Evolutionary Bootstrap Approach to Neural Network Pruning and Generalization, Unpublished working paper, SSRI working papers, 9718, 1997.

83. [JYJH1999] JM. Yang, JT. Horng, CY. Kao, Incorporation Family Competition into Gaussian and Cauchy Mutations to Training Neural Networks Using an Evolutionary Algorithm, In Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99), Vol. 3, p.1994-2001, Jul 1999.

84. [PPAM2011] P. Padhiary, A. Mishra, Development of Improved Artificial Neural Network Model for Stock Market Prediction, International Journal of Engineering Science and Technology (IJEST), Vol. 3, No. 2, p.1576-1581, Feb 2011.

85. [AOAC2007] A. Ozuna, A. Cifter, Nonlinear Combination of Financial Forecast with Genetic Algorithm, MPRA Paper No. 2488, posted 07, Nov 2007.

86. [GDAM2008] G. Dahl, A. McAvinney, T. Newhall, Parallelizing Neural Network Training for Cluster Systems, PDCN'08 Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, p.220-225, 2008.

87. [XJHA2008] X. Jiang, H. Adeli, Neuro-genetic algorithm for non-linear active control of structures, International Journal for Numerical Methods in Engineering, Vol. 75, Issue 7, p.770-786, Aug 2008.

88. [MVDL2008] M. Vasquez, D. Losada, F. Osorio, Exploiting Stock Data: A Survey Of Computational Techniques for Generating Portfolio Beliefs, Aceptado para publicar en Vol. 28, No. 1. Revista Ingenieria e Investigacion Facultad de Ingenieria, Apr 2008.

89. [EHMW2004] E. Hulthen, M. Wahde, Improved time series prediction using evolutionary algorithms for the generation of feedback connections in neural networks, In Proceedings of Computational Finance and its Applications, ISBN/ISSN: 1-85312-709-4, 2004.

90. [IEMH2011] I. Espino, M. Hernandez, Short-Term Wind Speed Forecasting Model based on Banks of Support Vector Regressors, 14th Conferencia de la Asociacion Española Para la Inteligencia Artificial (CAEPIA'11) San Cristobal de La Laguna, Tenerife, Espana, Vol. 1, Nov.

2011.

91. [DIPI2002] D. Pissarenko, Neural Networks For Financial Time Series Prediction: Overview Over Recent Research, A project completed as part of the requirements for the BSc (Hons) Computer Studies at the University of Derby in Austria, 2001-2002.

92. [CDAH1999] C. Dunis, A. Harris, S. Leong, P. Nacaskul, Optimising Intraday Trading Models with Genetic Algorithms, Neural Network World, No 9/3, p.193-223, 1999.

93. [MAHL2009] M. Hlavacek, Seasonal Time Series Modeling via Neural Networks with Switching Units, PhD Thesis, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University, Prague, Sep 2009.

94. [NWZM2007] N. Wagner, Z. Michalewicz, M. Khouja, RR. McGregor, Time Series Forecasting for Dynamic Environments: the DyFor Genetic Program Model, IEEE Transactions on Evolutionary Computation, Vol. 11, Issue 4, p.433-452, Aug 2007.

95. [LOBO2004] L. Bodis, Financial Time Series Forecasting Using Artificial Neural Networks, Master Thesis, Babes-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, 2004.

96. [MKMB2011] M. Khashei, M. Bijari, Which Methodology is Better for Combining Linear and Nonlinear Models for Time Series Forecasting?, Journal of Industrial and Systems Engineering, Vol. 4, No. 4, p.265-285, 2011.

97. [MDMB2012] M. Djennas, M. Benbouziane, M. Djennas, A Neural Network and Genetic Algorithm Hybrid Model for Modeling Exchange Rate: The Dollar – Kuwaiti Dinar Case, International Research Journal of Finance and Economics, Issue 92, p.141-162, Jun 2012.

98. [SKHK2008] S. Kim, HS. Kim, Neural networks and genetic algorithm approach for nonlinear evaporation and evapotranspiration modeling, Journal of Hydrology, Vol. 351, Issues 3–4, p.299–317, Apr 2008.

99. [JERI1997] J. Riley, An Evolutionary Approach to Training Feed-Forward and Recurrent Neural Networks, A minor thesis submitted in partial fulfilment of the requirements for the degree of Master of Applied Science in Information Technology, Department of Computer Science, Royal Melbourne Institute of Technology, Australia, Nov 1997.

100. [XYYL1997] X. Yao, Y. Liu, A New Evolutionary System for Evolving Artificial Neural Networks, IEEE Transactions On Neural Networks, Vol. 8, No. 3, May 1997.

101. [MMMH2010] M. Majumder, MD. Hussian, Forecasting of Indian Stock Market Index Using Artificial Neural Network, National Stock Exchange of India Limited, NSE Newsletter, Jul 2010.

102. [MSMJ2009] M. Shahi, M. Jaksa, H. Maier, State of the Art of Artificial Neural Networks in Geotechnical Engineering, Advances in Artificial Neural Systems (Bouquet 08), Vol. 2009, Article No. 5, 2009.

103. [WLRP2002] W. Leigh, R. Purvis, J. Ragusa, Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support, Decision Support Systems, Vol. 32, Issue 4, p.361–377, Mar 2002.

104. [JIHU2004] J. Huang, Genetic Algorithms with Functional Mutation and Mating Operators in Time Series Data Mining, A Thesis in Computer Science Submitted to the Graduate Faculty of Texas Tech University in Partial Fulfillment of the Requirements for the Degree of Master of Science, 2004.

105. [MPAC2008] ME. Pedersen, A. Chipperfield, Tuning Differential Evolution For Artificial Neural Networks, Hvass Laboratories Technical Report no. HL0803 (Nova Publishers 2011), 2008.

106. [BSDJ2009] B. Subudhi, D. Jena, An Improved Differential Evolution Trained Neural Network Scheme for Nonlinear System Identification, International Journal of Automation and Computing, Vol. 6, No. 2, p.137-144, May 2009.

107. [GNYZ2007] G. Ning, Y. Zhou, A Modified Differential Evolution Algorithm for Optimization Neural Network, In Proceedings of ISKE-2007, Advances in Intelligent Systems Research, Oct 2007.

108. [ABWD2008] AS. Wdaa, Differential Evolution for Neural Networks Learning Enhancement, A project report submitted in partial fulfillment of the requirements for the award of the degree of Master of Science (Computer Science), Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Oct 2008.

109. [ASMB2008] A. Slowik, M. Bialko, Training of Artificial Neural Networks Using Differential Evolution Algorithm, Conference on Human System Interactions (HSI08), Krakow, Poland, p.60-65, May 2008.

110. [HMAB2009] HM. Abdul Kader, Neural Networks Training Based on Differential Evolution Algorithm Compared with Other Architectures for Weather Forecasting34, IJCSNS International Journal of Computer Science and Network Security, Vol.9 No.3, p.92-99, Mar 2009.

111. [BYXH2006] B. Yu , X. He, Training Radial Basis Function Networks with Differential Evolution, In Proceeding of IEEE International Conference on Granular Computing (GrC'06), p.369-372, May 2006.

112. [JIJK2003] J. Ilonen, JI. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Process Letters, Vol.17, No.1, p.93–105, 2003.

113. [TTSS2009] T. Takahama, S. Sakai, A. Hara, N. Iwane, Predicting Stock Price Using Neural Networks Optimized by Differential Evolution with Degeneration, International Journal of Innovative Computing, Information and Control, Vol. 5, No. 12(B), p.5021–5031, Dec 2009.

114. [BSDJ2008] B. Subudhi, D. Jena, M. Gupta, Memetic Differential Evolution Trained Neural Networks For Nonlinear System Identification, In Proceedings of the IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems (ICIIS 2008), Kharagpur, India, p.1-6, Dec 2008.

115. [YWWL2011] YC. Wu, WP. Lee, CW. Chien, Modified the Performance of Differential Evolution Algorithm with Dual Evolution Strategy, In Proceeding of 2009 International Conference on Machine Learning and Computing IPCSIT, Vol.3 (2011), p.57-63, IACSIT Press, Singapore, 2011.

116. [HKMS2012] H. Kader, M. Salam, Evaluation of Differential Evolution and Particle Swarm Optimization Algorithms at Training of Neural Network for Stock Prediction, International Arab Jurnal of e-Technology, Vol. 2, No. 3, p.145-151, Jan 2012.

117. [VPDS1998a] V. Plagianakos, D. Sotiropoulos, M. Vrahatis, Integer weight training by differential evolution algorithms, Recent Advances in Circuits and Systems, p.327-331, 1998.

118. [ABWD2011] AS. Wdaa, Differential evolution for neural networks learning enhancement, Jurnal of university of anbar for pure science, Vol.5, No.2, p.79-84, 2011.

119. [SDAA2008] S. Das, A. Abraham, A. Konar, Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives, Advances of Computational Intelligence in Industrial Systems, Studies in Computational Intelligence (SCI), Vol.116, p.1-38, 2008.

120. [SLKG2009] S. Lahiri, K.Ghanta, Artificial neural network model with the parameter tuning assisted by a differential evolution technique: The study of the hold up of the slurry flow in a pipeline, Chemical Industry & Chemical Engineering Quarterly, Vol. 15, No. 2, p.103−117, 2009.

121. [KBWK2009] K. Bandurski, W. Kwedlo, Training neural networks with a hybrid differential evolution algorithm, Zeszyty Naukowe Politechniki Bialostockiej, Informatyka, Z. 4, p5-17, Bibliogr. 16 poz., Wykr. 2009.

122. [JLJL2005] J. Liu, J. Lampinen, A Differential Evolution Based Incremental Training Method for RBF Networks, In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO'05), p.881-888, Jun 2005.

123. [JDDH2007] JX. Du, DS. Huang, XF. Wang, X. Gu, Shape recognition based on neural networks trained by differential evolution algorithm, Neurocomputing, Vol. 70, Issue 4-6, p.896–903, Jan 2007.

124. [HSRG2011] H. Shah, R. Ghazali, N. Nawi, Using Artificial Bee Colony Algorithm for MLP Training on Earthquake Time Series Data Prediction, In Proceedings of CoRR, Vol. 3, Issue 6, p.135-142, 2011.

125. [CLCW2011] CJ. Lin, CF. Wu, CY. Lee, Design of a Recurrent Functional Neural Fuzzy Network Using Modified Differential Evolution, International Journal of Innovative Computing, Information and Control, Vol. 7, No. 2, p.669-683, Feb 2011.

126. [NAFK2011] N. Amjady, F. Keynia, H. Zareipour, Short-term wind power forecasting using ridgelet neural network, Electric Power Systems Research, Vol. 81, No.12, p.2099–2107, Dec 2011.

127. [VPMV2002] V. Plagianakos, M. Vrahatis, Parallel evolutionary training algorithms for “hardware-friendly” neural networks, Natural Computing, Vol. 1, Issue 2-3, p.307–322, Jun 2002.

128. [VPDS1998b] V. Plagianakos, D. Sotiropoulos, M. Vrahatis, Evolutionary Algorithms for Integer Weight Neural Network Training, Technical Report No.TR98-04, University of Patras, Greece, 1998.

129. [HARS2001] H. Abbass, R. Sarker, Simultaneous Evolution of Architectures and Connection Weights in ANNs, In Proceedings of 5th Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES 2001), Dunedin, New Zealand, p.16-21, 2001.

130. [VPMV2000] V. Plagianakos, M. Vrahatis, Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights, In Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Vol. 5, p.161-166, 2000.

131. [HFJL2003] HY. Fan, J. Lampinen, G. Dulikravich, Improvements to Mutation Donor

Formulation of Differential Evolution, International Congress on Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems EUROGEN03, p. 1-12, 2003.

132. [HTBM2011] H. Thethi, B. Majhi, G. Panda, Efficient System Identification using a Low Complexity Nonlinear Network with Differential Evolution and its variant based Training Schemes, International Journal of Computer Applications, Vol. 31, No. 8, p.38-46, Oct 2011.

133. [DXSL2007] D. Xu, S. Li, F. Qian, An Improved Differential Evolution Algorithm and its Application in Reaction Kinetic Parameters Estimation, ISKE-2007 Proceedings, Advances in Intelligent Systems Research, Oct 2007.

134. [HIAL2004] H. Al-Hindi, Approximation of a Discrete Event Stochastic Simulation Using an Evolutionary Artificial Neural Network, Journal of King Abdulaziz University, Engineering Sciences, Vol. 15, No. 1, p.125-138, 2004.

135. [RGAC2010] R. Giri, A. Chowdhury, A. Ghosh, S. Das, A. Abraham, V. Snasel, A Modified Invasive Weed Optimization Algorithm for Training of Feed-Forward Neural Networks, In Proceedings of Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC10), p.3166-3173, 2010.

136. [YLYL2011] YC. Lin, YC. Lin, WC. Chang, KL. Su, Evolutionary Neural Networks with Mixed-Integer Hybrid Differential Evolution, Journal of Computers, Vol. 6, No. 8, p.1591-1596, Aug 2011.

137. [WGAF2010] W. Gong, A. Fialho, Z. Cai, Adaptive Strategy Selection in Differential Evolution, In Proceedings of Genetic and Evolutionary Computation Conference (GECCO10), p.409-416, 2010.

138. [JNJS2011] J. Novo, J. Santos, M. Penedo, Optimization of Topological Active Nets with Differential Evolution, In Proceedings of the 10th international conference on Adaptive and natural computing algorithms (ICANNGA11), Vol. 6593, p.350-360, 2011.

139. [DTNP2004] D. Tasoulis, N. Pavlidis, V. Plagianakos, M. Vrahatis, Parallel Differential Evolution, In Proceedings of IEEE Congress on Evolutionary Computation (CEC 2004), p. 2023-2029, 2004.

140. [KKAS2006] K. Kozlov, A. Samsonov, New migration scheme for parallel differential evolution, In Proceedings of the Fifth International Conference on Bioinformatics of Genome Regulation and Structure, p.141–144, 2006.

141. [SGNB2011] S. Gonzalez, N. Barriga, Fully Parallel Differential Evolution, In Competition of GPUs for Genetic and Evolutionary Computation at the 2011 Genetic and Evolutionary Computation Conference (GECCO'2011), Dublin, Ireland, Jul 2011.

142. [DZDP2003] D. Zaharie, D. Petcu, Parallel implementation of multi-population differential evolution, In Proceedings of the NATO Advanced Research Workshop on Concurrent Information Processing and Computing (IOS Press, 2003), p. 223-232, 2003.

143. [MWFN2011] M. Weber, F. Neri, V. Tirronen, Shuffle Or Update Parallel Differential Evolution for Large Scale Optimization, Soft Computing, Vol. 15, Issue 11, p. 2089-2107, Nov 2011.

144. [PEBU2011] P. Bujok, Parallel Models of Adaptive Differential Evolution Based on Migration Process, Aplimat - Journal of Applied Mathematics, Vol. 4, No. 2, p.347-356, 2011.

145. [RSKP1995] R. Storn, K. Price, Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, ICSI, Berkeley, Mar 1995.

146. [PKJP2011a] P. Kromer, J. Platos, V. Snasel, A. Abraham, A Comparison of Many-threaded Differential Evolution and Genetic Algorithms on CUDA, In Proceedings of Third World Congress on Nature and Biologically Inspired Computing (NaBIC), p.509-514, 2011.

147. [DAZA2004a] D. Zaharie, A multipopulation differential evolution algorithm for multimodal

optimization, In Proceedings of Mendel, 10th International Conference on Soft Computing, p. 17-22, 2004.

148. [LRCC2011] L. Ramirez-Chavez, C. Coello, E. Rodriguez-Tello, A GPU-Based Implementation of Differential Evolution for Solving the Gene Regulatory Network Model Inference Problem, In Proceedings of the Fourth International Workshop on Parallel Architectures and Bioinspired Algorithms (WPABA'11), Oct 2011.

149. [SREN2010] S. Enaganti, Solving Correlation Matrix Completion Problems using Parallel Differential Evolution, A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in The University Of British Columbia, Nov 2010.

150. [JOTT2011] J. Olensek, T. Tuma, J. Puhan, A. Burmen, A new asynchronous parallel global optimization method based on simulated annealing and differential evolution, Applied Soft Computing, Vol. 11, Issue 1, p. 1481-1489, Jan 2011 .

151. [KITA2012] K. Tagawa, A Statistical Study of Concurrent Differential Evolution on Multi-core CPUs, In Proceedings of the Italian Workshop on Artificial Life, Evolution and Complexity (WIVACE 2012), 2012.

152. [DAZA2003] D. Zaharie, Control of Population Diversity and Adaptation in Differential Evolution Algorithms, In Proceedings of the 9th International Conference on Soft Computing (MENDEL 2003), p.41–46, Jun 2003.

153. [DXLD2012] D. Xie, L. Ding, X. Du, Y. Hu, S. Wang, Self-adaptive Pseudo-parallel Differential Evolution Algorithm, Journal of Computational Information Systems, Vol. 8, No. 8, p.3403-3411, 2012.

154. [ASML2007] A. Sutton, M. Lunacek, L. Whitley, Differential Evolution and Non-separability: Using selective pressure to focus search, In Proceedings of Genetic and Evolutionary Computation Conference (GECCO-07), p. 1428-1435, Jul 2007.

155. [MAWE2010] M. Weber, Parallel Global Optimization, Structuring Populations in Differential Evolution, Dissertation in Faculty of Information Technology of the University of Jyvaskyla, Nov 2010.

156. [KTTI2010a] K. Tagawa, T. Ishimizu, Concurrent Differential Evolution Based on MapReduce, International Journal Of Computers, Vol. 4, No. 4, p. 161-168, 2010.

157. [MEVP2008] M. Epitropakis, V. Plagianakos, M. Vrahatis, Balancing the exploration and exploitation capabilities of the Differential Evolution Algorithm, In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2008), p. 2686-2693, Jun 2008.

158. [SRHT2008] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition-Based Differential Evolution, In Proceedings of IEEE Transactions On Evolutionary Computation, Vol. 12, No. 1, p. 64-79, Feb 2008.

159. [RCAD2010] R. Cabido, A. Duarte, A. Montemayor, J. Pantrigo, Differential Evolution for Global Optimization on GPU, In Proceedings of the 3rd International Conference on Metaheuristics and Nature Inspired Computing (META'10), Oct 2010.

160. [NPVP2006] N. Pavlidis, V. Plagianakos, D. Tasoulis, M. Vrahatis, Human Designed Vs. Genetically Programmed Differential Evolution Operators, In Proceedings of IEEE Congress on Evolutionary Computation (CEC 2006), p.1880-1886, Jul 2006.

161. [DDJG2012] D. Davendra, J. Gaura, M. Bialic-Davendra, R. Senkerik, CUDA Based Enhanced Differential Evolution: a Computational Analysis, In Proceedings of the 26th European Conference on Modelling and Simulation (ECMS 2012), May 2012.

162. [PKJP2011b] P. Kromer, J. Platos, V. Snasel, A. Abraham, An Implementation of Differential Evolution for Independent Tasks Scheduling on GPU, In Proceedings of the 6th international conference on Hybrid artificial intelligent systems (HAIS'11), Vol. 1, p. 372-379, 2011.

163. [DLHL2012] DH. Lim, H. Luong, C. Ahn, A Novel Differential Evolution Using Multiple-

Deme Based Mutation, International Journal of Innovative Computing, Information and Control (IJICIC), Vol. 8, No. 5A. p. 3049-3060, May 2012.

164. [YGSG2011] Y. Gao, S. Gong, Ge Zhao , A Novel and Robust Evolution Algorithm for Optimizing Complicated Functions, In Proceedings of the Fourth International Workshop on Advanced Computational Intelligence (IWACI), p. 370-373, Oct 2011.

165. [NNHI2010] N. Noman, H. Iba, Cellular Differential Evolution Algorithm, In Proceedings of the 3rd Australasian Joint Conference Adelaide, Advances in Artificial Intelligence, p. 293-302, Dec 2010.

166. [NPDT2005] N. Pavlidis, D. Tasoulis, V. Plagianakos, M. Vrahatis, Spiking Neural Network Training Using Evolutionary Algorithms, In Proceedings of International Joint Conference on Neural Networks (IJCNN'05), Poznan University of Technology, Institute of Control and Information Engineering, p. 2190-2194, 2005.

167. [RMPS2010] R. Mallipeddi, P. Suganthan, Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies, In Proceedings of the First International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO10), Vol. 6466, p. 71-78, Dec 2010.

168. [KTTI2010b] K. Tagawa, T. Ishimizu, Concurrent Implementation of Differential Evolution, In Proceedings of the 10th WSEAS international conference on Systems theory and scientific computation (ISTASC'10), p. 65-70, 2010.

169. [KTTI2011] K. Tagawa, T. Ishimizu, Concurrent Differential Evolution for Uncertain Optimization Problems, In Proceedeings of The Fifth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP11), p. 48-53, Nov 2011.

170. [MDRT2011] M. Depolli, R. Trobec, B. Filipic, Parallel Evolutionary Algorithm for Heterogeneous Computer Systems, In Proceedings of PARNUM 2011 International Workshop on Parallel Numerics 11, Oct 2011.

171. [TDDA2010] T. Desell, D. Anderson, M. Magdon-Ismail, H. Newberg, B. Szymanski, C. Varela, An Analysis of Massively Distributed Evolutionary Algorithms, In Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2010), p. 1-8, Jul 2010.

172. [EZMT2011] E. Zaferani, M. Teshnehlab, The Parallel Evaluation Strategy Approach to Solving Optimization Problems, In Proceedings of the International Conference on Computer and Software Modeling (IPCSIT11), Vol. 14, p.123-127, 2011.

173. [DAZA2004b] D. Zaharie, Extensions of Differential Evolution Algorithms for Multimodal Optimization, In Proceedings of the 6th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'04), Vol. 42, No. 3, p. 523-534, 2004.

174. [FLHL2003] F. Leung, H. Lam, S. Ling, P. Tam, Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm, IEEE Transactions on Neural Networks, Vol. 14, No. 1, p. 79-88, Jan 2003.

175. [DZGC2006] D. Zaharie, G. Ciobanu, Distributed Evolutionary Algorithms Inspired by Membranes in Solving Continuous Optimization Problems, In Proceedings of the 7th international conference on Membrane Computing (WMC'06), p. 536-553, 2006.

176. [FFRK2012] F. Fabris, R. Krohling, A Co-evolutionary Differential Evolution algorithm for solving Min-Max problem: An implementation on the GPU with C-CUDA, Expert Systems with Applications, Vol. 39, No. 12, p. 10324–10333, Sep 2012.

177. [SDAA2009] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential Evolution Using a Neighborhood-based Mutation Operator, IEEE Transactions on Evolutionary Computation, Vol. 13, No. 3, p. 526-553, Jun 2009.

178. [EMJV2006] E. Mezura-Montes, J. Velazquez-Reyes, C. Coello-Coello, A Comparative Study of Differential Evolution Variants for Global Optimization, In Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO'06), p. 485-492, 2006.

179. [AIXL2011] A. Iorio, X. Li, Improving the performance and scalability of Differential Evolution on problems exhibiting parameter interactions, Soft Computing, Vol. 15, No. 9, p. 1769-1792, Sep 2011.

180. [WGZC2011] W. Gong, Z. Cai, C. Ling, H. Li, Enhanced Differential Evolution with Adaptive Strategies for Numerical Optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 41, No. 2, p. 397-413, 2011.

181. [YWZC2011] Y. Wang, Z. Cai, Q. Zhang, Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters, IEEE Transactions on Evolutionary Computation, Vol. 15, No. 1, p. 55-66, 2011.

182. [AMJT2012] A. Moraglio, J. Togelius, S. Silva, Geometric Differential Evolution for Combinatorial and Programs Spaces, Evolutionary Computation Journal (accepted for publication), Technical Report in School of Computing, University of Kent, Canterbury, 2012 .

183. [ZAKA2007] Z. Kajee-Bagdadi, Differential Evolution Algorithms for Constrained Global Optimization, A thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg in fulfillment of the requirements for the degree of Master of Science, 2007.

184. [MATO1999] M. Tomassini, Parallel and Distributed Evolutionary Algorithms: A Review, Chapter 7 in Miettinen et al. (Eds) Evolutionary Algorithms in Engineering and Science, Wiley, p. 113-133, 1999.

185. [WRWI2006] W. Wickramasinghe, Adaptive Distributed Evolutionary Algorithms, A thesis submitted for the degree of Masters in Parallel and Distributed Computer Systems, Department of Computer Science Vrije Universiteit, Amsterdam, The Netherlands, Jul 2006.

186. [MICE2006] M. Cervenka, Distributed Evolutionary Algorithms, Doctoral Thesis, Tomas Bata University in Zlin Faculty of Applied Informatics, Zlin, 2006.

187. [EAGL2004] E. Alba, G. Luque, Growth Curves and Takeover Time in Distributed Evolutionary Algorithms, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004), p. 864-876, 2004.

188. [JMAM2007] J. Merelo, A. Mora-García, J. Laredo, J. Lupion, F. Tricas, Browser-based Distributed Evolutionary Computation: Performance and Scaling Behavior, In Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation (GECCO'07), p 2851-2858, 2007.

189. [SIHA2006] S. Harding, A Distributed Evolutionary Algorithm Using C# and Mono, Technical Report, Department Of Computer Science at Memorial University, Jun 2006.

190. [LAJM2009] L. Araujo, J. Merelo-Guervos, Multikulti Algorithm: using genotypic differences in adaptive distributed evolutionary algorithm migration policies, In Proceedings of the IEEE Congress on Evolutionary Computation (CEC'09), p. 2858-2865, May 2009.

191. [MGJM2011] M. Garcia-Arenas, J. Merelo, A. Mora, P. Castillo, G. Romero, J. Laredo, Assessing Speed-ups In Commodity Cloud Storage Services For Distributed Evolutionary Algorithms, In Proceedings of the IEEE Congress on Evolutionary Computation, p. 304-311, 2011.

192. [WWMS2007] W. Wickramasinghe, M van Steen, A. Eiben, Peer-to-peer evolutionary algorithms with adaptive autonomous selection, In Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO'07), p. 1460-1467, 2007.

193. [PCPV2011] P. Chitra, P. Venkatesh, R. Rajaram, Comparison of evolutionary computation algorithms for solving bi-objective task scheduling problem on heterogeneous distributed computing systems, Journal Sadhana, Indian Academy of Sciences, Vol. 36, No. 2, p. 167–180, Apr 2011.

194. [SOCL2002] SK. Oh, CY. Lee, JJ. Lee, A New Distributed Evolutionary Algorithm for Optimization in Nonstationary Environments, In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Vol. 1, p. 378-383, May 2002.

195. [JDKM2003] J. Digalakis, K. Margaritis, Parallel Evolutionary Algorithms on Message-Passing Clusters, In Proceedings of Parallel Computing Conference, 2003.

196. [KSSL2008] K. Sullivan, S. Luke, C. Larock, S. Cier, S. Armentrout, Opportunistic Evolution: Efficient Evolutionary Computation on Large-Scale Computational Grids, In Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation (GECCO'08), p. 2227-2232, 2008.

197. [MDCG2006] M. Dubreuil, C. Gagne, M. Parizeau, Analysis of a Master-Slave Architecture for Distributed Evolutionary Computations, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 36, No. 1, p. 229-235, Feb 2006.

198. [CGMP2003a] C. Gagne, M. Parizeau, M. Dubreuil, Distributed Beagle: An Environment for Parallel and Distributed Evolutionary Computations, In Proceedings of the 17 th Annual International Symposium on High Performance Computing Systems and Applications (HPCS), p. 201-208, May 2002.

199. [JMMG2011] J. Merelo, M. Garcia-Arenas, A. Mora, P. Castillo, G. Romero, J. Laredo, Cloud-based Evolutionary Algorithms: An algorithmic study, In Proceedings of Computing Research Repository (CORR), Vol. abs/1105.6, 2011.

200. [JLAE2008] J. Laredo, A. Eiben, M. van Steen, P. Castillo, A. Mora, J. Merelo, P2P Evolutionary Algorithms: A Suitable Approach for Tackling Large Instances in Hard Optimization Problems, In Proceedings of Euro-Par 2008 - Parallel Processing, 14th International Euro-Par Conference, Vol. 5168, p. 622-631, 2008.

201. [RTSY2007] R. Tinos, S. Yang, Self-Adaptation of Mutation Distribution in Evolutionary Algorithms, In Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007), p. 79-86, Sep 2007.

202. [JLPC2008] J. Laredo, P. Castillo, A. Mora, C. Fernandes, J. Merelo, Resilience to Churn of a

Peer-to-Peer Evolutionary Algorithm, International Journal of High Performance Systems Architecture, Vol. 1, No. 4, p. 260-268, Mar 2008.

203. [TBWK2003] T. Burczynski, W. Kus, Distributed Evolutionary Algorithm with Master – Co-Evolutionary Algorithm and Slaves – Fitness Function Evaluators, In Proceedings of the 6th National Conference on Evolutionary Computation and Global Optimization, p. 33-39, May 2003.

204. [GWDG2005] G. Winter, D. Greiner, B. Galvan, Parallel Evolutionary Computation, CEANI (Evolutionary Computation and Applications) Division of the Institute of Intelligent Systems and Numerical Applications in Engineering (IUSIANI), University of Las Palmas de Gran Canaria, Spain, 2005.

205. [JULA2010] J. Laredo, Peer-to-Peer Evolutionary Computation: A Study of Viability, Tesis Doctoral, Universidad de Granada, Departamento de Arquitectura y Tecnologia de Computadores, Granada, 2010.

206. [ATPG2007] A. Tallon-Ballesteros, P. Gutierrez-Pena, C. Hervas-Martinez, Distribution of the search of evolutionary product unit neural networks for classification, In Proceedings of the IADIS International Conference Applied Computing, p. 266-273, 2007.

207. [NCTD2010] N. Cole, T. Desell, D. Gonzalez, F. de Vega, M. Magdon-Ismail, H. Newberg, B. Szymanski, C. Varela, Evolutionary Algorithms on Volunteer Computing Platforms: The MilkyWay@Home Project, Parallel and Distributed Computational Intellligence, Vol. 269, p. 63-90, 2010.

208. [CGMP2003b] C. Gagne, M. Parizeau, M. Dubreuil, A Robust Master-Slave Distribution Architecture for Evolutionary Computations, In Proceedings of Genetic and Evolutionary Computation COnference (GECCO'03), p. 80-87, Jul 2003.

209. [TDMM2010] T. Desell, M. Magdon-Ismail, B. Szymanski, C. Varela, H. Newberg, D. Anderson, Validating Evolutionary Algorithms on Volunteer Computing Grids, In Proceedings of the 10th IFIP international conference on distributed applications and interoperable systems

(DAIS), p. 29-41, Jun 2010.

210. [EAJT1999] E. Alba, J. Troya, A Survey of Parallel Distributed Genetic Algorithms, Complexity, Vol. 4, No. 4, p. 31-52, 1999.

211. [JMAM2012] J. Merelo, A. Mora, C. Fernandes, M. Arenas, A. Esparcia-Alcazar, Using pool-based evolutionary algorithms for scalable and asynchronous distributed computing, In Proceedings of the 12th International Conference on Parallel Problem Solving From Nature (PPSN 2012), Sep 2012.

212. [TCPL2009] T. Chen, P. Lehre, K. Tang, X. Yao, When Is an Estimation of Distribution Algorithm Better than an Evolutionary Algorithm?, In Proceedings of the 10th IEEE Congress on Evolutionary Computation (CEC 2009), p. 1470-1477, May 2009.

213. [TJGC2007] T. Jumonji, G. Chakraborty, H. Mabuchi, M. Matsuhara, A Novel Distributed Genetic Algorithm Implementation with Variable Number of Islands, In Proceedings of IEEE Congress on Evolutionary Computation, p. 4698-4705, 2007.

214. [MATF2004] M. Aydin, T. Fogarty, A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimisation Problems, Journal of Heuristics, Vol. 10, No. 3, p.269–292, May 2004.

215. [EAMT2002] E. Alba, M. Tomassini, Parallelism and Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 6, No. 5, p. 443-462, Oct 2002.

216. [WOKW2004] W. Kwedlo, A Parallel Evolutionary Algorithm for Discovery of Decision Rules, Parallel Processing and Applied Mathematics Lecture Notes in Computer Science, Vol. 3019, p 580-585, 2004.

217. [MTMQ2015] MetaTrader 4, MetaQuotes Software Corp., (2015, October 20) http://www.metatrader4.com/

218 [TBIZ2015] Balabanov, T., Zankinski, I., Barova, M., Distributed Evolutionary Computing Migration Strategy by Incident Node Participation. Large-Scale Scientific Computing, Lecture Notes in Computer Science, 9374, Springer International Publishing Switzerland, ISBN:978-3-319-26520-9, DOI:10.1007, p 203-209, 2015.

219 [ATHA2004] Attiya, Hagit and Welch, Jennifer (2004). Distributed Computing: Fundamentals, Simulations, and Advanced Topics. Wiley-Interscience. ISBN 0471453242.

220 [JLAM2002] Bibliography on DE from 1995 to 2002, http://www.lut.fi/~jlampine/debiblio.htm

221 [GEJE1976] Box, George; Jenkins, Gwilym (1976), Time series analysis: forecasting and control, rev. ed., Oakland, California: Holden-Day

222 [STOR1996] DE Algorithms' Home Page, http://www.icsi.berkeley.edu/~storn/code.html

223 [DUWI2002] Dunis, C.L. & Williams, M., Modelling and trading the eur/usd exchange rate: Do neural network models perform better? Derivatives Use, Trading and Regulation, 8(3), pp. 211–239, 2002.

224 [FEOK2006] Feoktistov V., Differential Evolution In Search of Solutions, 2006, XII, 196 p. 73 illus., Hardcover, ISBN: 978-0-387-36895-5.

225 [GAVI2002] Garg, Vijay K. (2002). Elements of Distributed Computing. Wiley-IEEE Press. ISBN 0471036005.

226 [GENE2000] Gershenfeld, Neil (2000), The nature of mathematical modeling, Cambridge: Cambridge Univ. Press, ISBN 978-0521570954, OCLC 174825352.

227 [GILA2001] Giles, C.L., Lawrence, S. & Tsoi, A.C., Noisy time series prediction using a recurrent neural network and grammatical inference. Machine Learning, 44(1/2), pp. 161–183, 2001.

228 [HAYK1999] Haykin, S., Neural Networks, A Comprehensive Foundation. Prentice-Hall, Inc., 2nd edition, 1999.

229 [HOLL1975] Holland, J., Adaptation In Natural and Artificial Systems. The University of Michigan Press, 1975.

230 [MOOD1995] Moody, J.E., Economic forecasting: Challenges and neural network solutions. Proceedings of the International Symposium on Artificial Neural Networks, Hsinchu, Taiwan, 1995.

231 [PRST2005] Price, Kenneth, Storn, Rainer M., Lampinen, Jouni A., Differential Evolution A Practical Approach to Global Optimization, 2005, XX, 538 p. 292 illus. With CD-ROM., Hardcover, ISBN: 978-3-540-20950-8.

232 [ROBA2009] Roman M. Balabin, Ekaterina I. Lomakina (2009). "Neural network approach to quantum-chemistry data: Accurate prediction of density functional theory energies". J. Chem. Phys. 131 (7): 074104. doi:10.1063/1.3206326.

233 [TEGE1994] Tel, Gerard (1994). Introduction to Distributed Algorithms. Cambridge University Press.

234 [BILL1998] The Machine Learning Dictionary. http://www.cse.unsw.edu.au/~billw/mldict.html#activnfn.

235 [WERB1990] Werbos, P., Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), pp. 1550–1560, 1990.

236 [YAOX1999] Yao, X., Evolving artificial neural networks. Proc of the IEEE, 87(9), pp. 1423–1447, 1999.

237 [NIRA2012] Николова, Р., Разработка на система за графично изобразяване и визуално манипулиране на изкуствени невронни мрежи, НБУ-София, 2012.

238 [ANMO2011] Аначков, М., Система за визуализиране и манипулиране на изкуствени невронни мрежи“, ТУЕС към ТУ-София, 2011.

239 [SINI2011] Симеонов, Н., Мултимедиен скрийнсейвър за Windows, подходящ за вграждане в проекти за разпределени изчисления, ТУЕС към ТУ-София, 2011.

240 [HREL2011] Христова, Е., Мултимедиен скрийнсейвър за Linux, подходящ за вграждане в проекти за разпределени изчисления, ТУЕС към ТУ-София, 2011.

Списък с публикациите по дисертационната работа

1 [BAZA2010] Балабанов Т., Занкински И., Симеонова В., Прогнозиране на времеви редове с изкуствени невронни мрежи и диференциална еволюция в разпределена среда, Работни статии на ИИТ, IIT/WP-268B, ISSN:1310-652X, 2010.

2 [BATO2011] Балабанов. Т., Прогнозиране с евристични подходи в разпределена среда, Proceedings of Anniversary Scientific Conference 40 Years Department of Industrial Automation, ISBN:978-954-465-043-8, 163-166, 2011.

3 [BAZA2011] Balabanov. T., Zankinski I., Dobrinkova N., Time Series Prediction by Artificial Neural Networks and Differential Evolution in Distributed Environment, Large-Scale Scientific Computing, Lecture Notes in Computer Science, vol. 7116, ISBN:978-3-642-29842-4, DOI:10.1007/978-3-642-29843-1_22, 198-205, 2011.

4 [BATO2015] Балабанов, Т., Избягване на локални оптимуми при евристични популационни алгоритми в разпределена среда, Сборник с доклади от XXII Международен симпозиум Управление на енергийни, индустриални и екологични системи, ISSN:1313-2237, 83 - 86, 2015.

5 [BAZA2015] Balabanov, T., Zankinski, I., Barova, M.. Distributed Evolutionary Computing Migration Strategy by Incident Node Participation. Large-Scale Scientific Computing, Lecture Notes in Computer Science, 9374, Springer International Publishing Switzerland, ISBN:978-3-319-26520-9, DOI:10.1007, 203 - 209, 2015.

6 [BAGE2016a] Балабанов, Т., Генова, К., AJAX разпределена система за обучение на изкуствени невронни мрежи с еволюционни алгоритми, Сборник с доклади от XXIV Международен симпозиум Управление на топло енергийни обекти и системи, Управление на енергийни, индустриални и екологични системи, ISSN:1313-2237, 49 - 52, 2016.

7 [BAKE2016] Balabanov T., Keremedchiev D., Goranov I., Web Distributed Computing For

Evolutionary Training Of Artificial Neural Networks, International Conference InfoTech-2016, Varna - St. St. Constantine and Elena resort, Bulgaria, ISSN:1314-1023, 210-216, 2016.

8 [BAGE2016b] Балабанов, Т., Генова, К., Разпределена система за обучение на изкуствени невронни мрежи, базирана на мобилни устройства, Proceedings of International Conference Automatics and Informatics'2016, ISSN:1313-1850, 49 - 52, 2016.

Списък с цитирания и реферирания

Balabanov, T., Zankinski, I., Barova, M., Strategy for Individuals Distribution by Incident Nodes Participation in Star Topology of Distributed Evolutionary Algorithms, Cybernetics and Information Technologies, Vol. 16 No 1, ISSN: 1314-4081, 2016.

> 1. Tomov, P., Monov, V., Artificial Neural Networks and Differential Evolution Used for Time Series Forecasting in Distributed Environment, Proceedings of International Conference Automatics and Informatics, Sofia, Bulgaria, ISSN 1313-1850, pp. 129-132, 2016.

Balabanov, T., Zankinski, I., Barova, M., Distributed Evolutionary Computing Migration Strategy by Incident Node Participation, International Conference on Large-Scale Scientific Computing, 10th International Conference, LSSC 2015, Sozopol, Bulgaria, June 8-12, ISBN 978-3-319-26519-3, pp. 203-209, 2015.

> 2. Tomov, P., Monov, V., Artificial Neural Networks and Differential Evolution Used for Time Series Forecasting in Distributed Environment, Proceedings of International Conference Automatics and Informatics, Sofia, Bulgaria, ISSN 1313-1850, pp. 129-132, 2016.
>
> 3. Ташо Ташев, Арсений Баканов, Радостина Ташева. Верхняя граница пропускной способности коммутатора с матричным переключателем для входящего трафика типа модифицированной модели Чанг-а. Доклади на Уиверситетската годишна научна конференция на Национален Военен Университет «В.Левски» 2016, 20-21 Октомври 2016, Велико Търново, България. Издагелски комплекс НВУ, 2016, том 2, стр.107-115. ISSN 1314-1937.

Balabanov, T., Zankinski, I., Dobrinkova, N., Time Series Prediction by Artificial Neural Networks and Differential Evolution in Distributed Environment, International Conference on Large-Scale Scientific Computing, 8th International Conference, LSSC 2011, Sozopol, Bulgaria, June 6-10, ISBN 978-3-642-29842-4, pp. 198-205, 2011.

> 4. Keremedchiev, D., Barova, M., Tomov, P., Mobile Application as Distributed Computing System for Artificial Neural Networks Training Used in Perfect Information Games, Proceedeings of International Scientific Conference UniTech, Gabrovo, Bulgaria, ISSN 1313-230X, vol. 2, pp. 389-393, 2016.

Приложение А – Експериментални данни

Данни използвани за експериментите с кръгова топология и инцидентно включване на възли

Съотношение на EUR/USD в периода 01.01.2014 до 31.03.2014 при дневен период на отчитане (източник https://www.investing.com/currencies/eur-usd-historical-data):

Date	Price	Open	High	Low	Change %
Mar 31, 2014	1.3771	1.3755	1.3807	1.3734	0.13%
Mar 28, 2014	1.3753	1.3740	1.3773	1.3705	0.09%
Mar 27, 2014	1.3740	1.3781	1.3798	1.3729	-0.31%
Mar 26, 2014	1.3783	1.3827	1.3829	1.3777	-0.32%
Mar 25, 2014	1.3827	1.3839	1.3848	1.3750	-0.09%
Mar 24, 2014	1.3839	1.3792	1.3876	1.3760	0.33%
Mar 21, 2014	1.3794	1.3779	1.3813	1.3767	0.11%
Mar 20, 2014	1.3779	1.3832	1.3845	1.3749	-0.39%
Mar 19, 2014	1.3833	1.3933	1.3935	1.3809	-0.72%
Mar 18, 2014	1.3934	1.3921	1.3942	1.3880	0.09%
Mar 17, 2014	1.3922	1.3911	1.3949	1.3879	0.06%
Mar 14, 2014	1.3914	1.3866	1.3937	1.3847	0.32%
Mar 13, 2014	1.3869	1.3903	1.3968	1.3844	-0.25%
Mar 12, 2014	1.3904	1.3859	1.3914	1.3841	0.32%
Mar 11, 2014	1.3859	1.3874	1.3880	1.3832	-0.13%
Mar 10, 2014	1.3877	1.3875	1.3898	1.3861	-0.01%
Mar 07, 2014	1.3878	1.3860	1.3917	1.3854	0.12%
Mar 06, 2014	1.3861	1.3734	1.3874	1.3723	0.93%
Mar 05, 2014	1.3733	1.3742	1.3746	1.3706	-0.08%
Mar 04, 2014	1.3744	1.3735	1.3782	1.3720	0.07%
Mar 03, 2014	1.3735	1.3786	1.3793	1.3725	-0.49%
Feb 28, 2014	1.3803	1.3708	1.3824	1.3693	0.69%
Feb 27, 2014	1.3709	1.3687	1.3727	1.3642	0.15%
Feb 26, 2014	1.3688	1.3744	1.3758	1.3661	-0.42%
Feb 25, 2014	1.3746	1.3735	1.3767	1.3714	0.08%
Feb 24, 2014	1.3735	1.3735	1.3770	1.3708	-0.04%
Feb 21, 2014	1.3740	1.3720	1.3760	1.3704	0.16%
Feb 20, 2014	1.3718	1.3733	1.3763	1.3685	-0.11%
Feb 19, 2014	1.3733	1.3757	1.3774	1.3724	-0.18%
Feb 18, 2014	1.3758	1.3706	1.3770	1.3694	0.36%
Feb 17, 2014	1.3708	1.3688	1.3725	1.3687	0.11%

Date	Price	Open	High	Low	Change %
Feb 14, 2014	1.3693	1.3681	1.3718	1.3674	0.09%
Feb 13, 2014	1.3681	1.3594	1.3694	1.3586	0.64%
Feb 12, 2014	1.3594	1.3639	1.3655	1.3563	-0.33%
Feb 11, 2014	1.3639	1.3648	1.3686	1.3631	-0.06%
Feb 10, 2014	1.3647	1.3634	1.3653	1.3609	0.09%
Feb 07, 2014	1.3635	1.3590	1.3642	1.3553	0.33%
Feb 06, 2014	1.3590	1.3533	1.3621	1.3485	0.42%
Feb 05, 2014	1.3533	1.3519	1.3555	1.3498	0.10%
Feb 04, 2014	1.3519	1.3525	1.3539	1.3494	-0.04%
Feb 03, 2014	1.3525	1.3493	1.3536	1.3476	0.28%
Jan 31, 2014	1.3487	1.3557	1.3570	1.3480	-0.51%
Jan 30, 2014	1.3556	1.3662	1.3666	1.3543	-0.78%
Jan 29, 2014	1.3663	1.3668	1.3685	1.3603	-0.04%
Jan 28, 2014	1.3669	1.3673	1.3689	1.3629	-0.02%
Jan 27, 2014	1.3672	1.3677	1.3718	1.3654	-0.04%
Jan 24, 2014	1.3678	1.3697	1.3741	1.3665	-0.12%
Jan 23, 2014	1.3695	1.3547	1.3699	1.3530	1.09%
Jan 22, 2014	1.3548	1.3559	1.3584	1.3535	-0.10%
Jan 21, 2014	1.3561	1.3552	1.3571	1.3518	0.07%
Jan 20, 2014	1.3552	1.3543	1.3570	1.3508	0.08%
Jan 17, 2014	1.3541	1.3620	1.3623	1.3517	-0.58%
Jan 16, 2014	1.3620	1.3604	1.3650	1.3582	0.11%
Jan 15, 2014	1.3605	1.3680	1.3684	1.3582	-0.54%
Jan 14, 2014	1.3679	1.3670	1.3699	1.3648	0.05%
Jan 13, 2014	1.3672	1.3669	1.3685	1.3637	0.01%
Jan 10, 2014	1.3670	1.3609	1.3689	1.3577	0.45%
Jan 09, 2014	1.3609	1.3577	1.3636	1.3550	0.25%
Jan 08, 2014	1.3575	1.3617	1.3636	1.3554	-0.30%
Jan 07, 2014	1.3616	1.3627	1.3657	1.3596	-0.09%
Jan 06, 2014	1.3628	1.3589	1.3653	1.3571	0.29%
Jan 03, 2014	1.3588	1.3672	1.3674	1.3582	-0.61%
Jan 02, 2014	1.3672	1.3756	1.3777	1.3630	-0.60%
Jan 01, 2014	1.3754	1.3742	1.3769	1.3742	0.06%

Данни за сравнение на правия пас при обучение на изкуствена невронна мрежа със C++ и JavaScript

Съотношение на EUR/USD в периода 01.01.2015 до 31.03.2015 при дневен период на отчитане (източник https://www.investing.com/currencies/eur-usd-historical-data):

Date	Price	Open	High	Low	Change %
Mar 31, 2015	1.0731	1.0833	1.0846	1.0713	-0.95%
Mar 30, 2015	1.0834	1.0889	1.0897	1.0811	-0.50%
Mar 27, 2015	1.0888	1.0886	1.0949	1.0802	0.04%
Mar 26, 2015	1.0884	1.0969	1.1052	1.0856	-0.80%
Mar 25, 2015	1.0972	1.0926	1.1016	1.0901	0.42%
Mar 24, 2015	1.0926	1.0947	1.1031	1.0892	-0.19%
Mar 23, 2015	1.0947	1.0837	1.0972	1.0768	1.16%
Mar 20, 2015	1.0822	1.0661	1.0883	1.0650	1.52%
Mar 19, 2015	1.0660	1.0866	1.0915	1.0613	-1.90%
Mar 18, 2015	1.0867	1.0598	1.1033	1.0581	2.54%
Mar 17, 2015	1.0598	1.0566	1.0650	1.0551	0.28%
Mar 16, 2015	1.0568	1.0494	1.0621	1.0459	0.70%
Mar 13, 2015	1.0495	1.0637	1.0639	1.0462	-1.32%
Mar 12, 2015	1.0635	1.0547	1.0684	1.0495	0.83%
Mar 11, 2015	1.0547	1.0699	1.0718	1.0511	-1.41%
Mar 10, 2015	1.0698	1.0851	1.0856	1.0692	-1.42%
Mar 09, 2015	1.0852	1.0832	1.0908	1.0823	0.07%
Mar 06, 2015	1.0844	1.1033	1.1035	1.0842	-1.70%
Mar 05, 2015	1.1031	1.1079	1.1115	1.0989	-0.43%
Mar 04, 2015	1.1079	1.1178	1.1187	1.1062	-0.87%
Mar 03, 2015	1.1176	1.1183	1.1219	1.1155	-0.08%
Mar 02, 2015	1.1185	1.1187	1.1242	1.1159	-0.09%
Feb 27, 2015	1.1195	1.1199	1.1248	1.1178	-0.03%
Feb 26, 2015	1.1198	1.1361	1.1381	1.1184	-1.44%
Feb 25, 2015	1.1362	1.1340	1.1389	1.1334	0.19%
Feb 24, 2015	1.1341	1.1336	1.1360	1.1289	0.08%
Feb 23, 2015	1.1332	1.1385	1.1414	1.1296	-0.43%
Feb 20, 2015	1.1381	1.1370	1.1431	1.1280	0.13%
Feb 19, 2015	1.1366	1.1397	1.1450	1.1355	-0.27%
Feb 18, 2015	1.1397	1.1412	1.1418	1.1334	-0.13%
Feb 17, 2015	1.1412	1.1354	1.1450	1.1322	0.52%
Feb 16, 2015	1.1353	1.1385	1.1429	1.1317	-0.28%
Feb 13, 2015	1.1385	1.1403	1.1443	1.1380	-0.16%
Feb 12, 2015	1.1403	1.1334	1.1423	1.1302	0.61%
Feb 11, 2015	1.1334	1.1321	1.1349	1.1280	0.11%
Feb 10, 2015	1.1321	1.1327	1.1347	1.1274	-0.03%
Feb 09, 2015	1.1324	1.1315	1.1359	1.1271	0.08%
Feb 06, 2015	1.1315	1.1477	1.1488	1.1313	-1.41%
Feb 05, 2015	1.1477	1.1345	1.1500	1.1304	1.15%
Feb 04, 2015	1.1347	1.1482	1.1486	1.1317	-1.17%
Feb 03, 2015	1.1481	1.1343	1.1534	1.1313	1.23%
Feb 02, 2015	1.1342	1.1284	1.1364	1.1280	0.48%
Jan 30, 2015	1.1288	1.1321	1.1366	1.1279	-0.28%
Jan 29, 2015	1.1320	1.1287	1.1369	1.1261	0.28%
Jan 28, 2015	1.1288	1.1382	1.1385	1.1276	-0.82%
Jan 27, 2015	1.1381	1.1237	1.1423	1.1224	1.26%

Date	Price	Open	High	Low	Change %
Jan 26, 2015	1.1239	1.1186	1.1297	1.1099	0.28%
Jan 23, 2015	1.1208	1.1366	1.1374	1.1114	-1.39%
Jan 22, 2015	1.1366	1.1610	1.1648	1.1317	-2.10%
Jan 21, 2015	1.1610	1.1553	1.1677	1.1542	0.51%
Jan 20, 2015	1.1551	1.1606	1.1616	1.1540	-0.47%
Jan 19, 2015	1.1606	1.1551	1.1641	1.1545	0.31%
Jan 16, 2015	1.1570	1.1633	1.1650	1.1462	-0.52%
Jan 15, 2015	1.1631	1.1789	1.1795	1.1570	-1.34%
Jan 14, 2015	1.1789	1.1772	1.1846	1.1726	0.14%
Jan 13, 2015	1.1772	1.1833	1.1860	1.1753	-0.52%
Jan 12, 2015	1.1834	1.1838	1.1871	1.1786	-0.06%
Jan 09, 2015	1.1841	1.1793	1.1848	1.1763	0.41%
Jan 08, 2015	1.1793	1.1839	1.1848	1.1753	-0.39%
Jan 07, 2015	1.1839	1.1892	1.1898	1.1803	-0.42%
Jan 06, 2015	1.1889	1.1933	1.1969	1.1884	-0.38%
Jan 05, 2015	1.1934	1.2003	1.2008	1.1883	-0.57%
Jan 02, 2015	1.2003	1.2106	1.2111	1.2001	-0.83%
Jan 01, 2015	1.2104	1.2100	1.2107	1.2094	0.05%

Данни за сравнение между трислойни изкуствени неверонни мрежи и мрежи с пълна свързаност

Дневен брой на изминати крачки, с цел прогнозно определяне на изразходваната енергия (източник https://github.com/TodorBalabanov/FullyConnectedVsTriLayersANN20160430).

3938, 1317, 4021, 10477, 9379, 7707, 9507, 4194, 2681, 3522, 5599, 5641, 6737, 7781, 2044, 1501, 6586, 4915, 5918, 6132, 9394, 2113, 935, 9729, 5236, 8815, 3169, 5888, 5722, 191, 9539, 3384, 6006, 7139, 7285, 136, 1843, 5094, 3795, 5985, 5566, 3545, 965, 14, 3738, 4645, 8439, 6390, 13842, 7754, 11440, 7572, 4876, 3206, 5577, 2734, 1169, 20, 5049, 6612, 2685, 7000, 6711, 4091, 26, 5383, 5516, 7185, 6118, 4484, 2178, 754, 8104, 8209, 6159, 11137, 8994, 5172, 425, 8082, 5337, 5712, 7157, 6385, 3343, 4196, 5957, 8581, 3686, 0, 254, 1819, 1071, 876, 3509, 2777, 1474, 4945, 3971, 21, 5466, 5509, 1316, 5653, 2775, 797, 22, 5601, 6177, 5662, 5132, 6543, 1700, 4361, 6951, 7734, 3451, 5385, 6358, 6838, 19, 6460, 5813, 6839, 6335, 2105, 8, 6, 9530, 1250, 5668, 5595, 6008, 2315, 1712, 8553, 5570, 5979, 4818, 6745, 5250, 43, 5727, 7416, 5888, 6270, 4931, 0, 31, 6190, 11164, 5768, 7307, 5412, 2716, 35, 8391, 6054, 2796, 5081, 6646, 4597, 1978, 7570, 5909, 9581, 3571, 6740, 1702, 1080, 6719, 963, 6781, 7544, 7708, 1993, 597, 2394, 5516,

12966, 723, 6528, 2476, 86, 5956, 5820, 6995, 6682, 2460, 2479, 56, 7095, 7255, 6310, 9971, 3725, 5400, 452, 6018, 5803, 6673, 6098, 9476, 692, 20, 7855, 11970, 10557, 5696, 7765, 3847, 47, 6020, 6037, 5684, 7089, 6372, 970, 861, 3590, 7672, 3730, 10689, 9428, 1514, 2062, 6154, 5234, 6160, 5134, 879, 1079, 9164, 6338, 6687, 8195, 6351, 1123, 4216, 3759, 9372, 7782, 3143, 4773, 6993, 849, 906, 6385, 7512, 8824, 8150, 12464, 7726, 8745, 13594, 6589, 6524, 2784, 0, 1785, 688, 7998, 6797, 8289, 10815, 10280, 4839, 3928, 10935, 4588, 5785, 6771, 7628, 2908, 11391, 6637, 5585, 7454, 5828, 8259, 6644, 2436, 7055, 7206, 7873, 7368, 6239, 3595, 3166, 1846, 2301, 21, 1600, 2390, 1894, 1469, 9097, 8401, 2034, 3244, 8811, 2979, 20, 7808, 7698, 11031, 4556, 7149, 3745, 5563, 9673, 8149, 12158, 7043, 6273, 1855, 80, 10729, 5880, 9327, 6343, 7227, 3522, 1244, 6382, 7186, 4964, 6162, 7435, 10524, 2449, 7437, 11970, 6661, 6122, 7323, 6707, 25, 2270, 5117, 6676, 5317, 7032, 7689, 4891, 8051, 5699, 4927, 11553, 6418, 2968, 11338, 7662, 9976, 5526, 14341, 4331, 10026, 1672, 5199, 4699, 7774, 7958, 7720, 2499, 10745, 19609, 15896, 5705, 6207, 7699, 2543, 32, 3642, 6307, 7491, 6236, 8644, 2121, 1448, 7838, 5434, 5945, 6074, 6962, 5441, 42, 7424, 5818, 8877, 5743, 7980, 3140, 3046, 8329, 8186, 5994, 2931, 7309, 862, 145, 8141, 6252, 9536, 6213, 7150, 2718, 1687, 5000, 6068, 5918, 10652, 12257, 1505, 2421, 10518, 2368, 7341, 8137, 7997, 3437, 2009, 5468, 3947, 5836, 8567, 11039, 3726, 746, 3417, 8649, 8016, 7652, 8298, 1306, 4031, 5525, 6203, 11847, 7688, 10911, 1080, 1001, 12315, 6084, 6529, 4074, 8526, 3161, 2184, 7400, 4916, 4521, 1523, 398, 1364, 925, 38, 2580, 1039, 6556, 2040, 1166, 825, 7672, 7177, 6104, 7928, 6240, 1420, 1214, 10638, 10726, 2323, 6113, 8112, 2757, 3761, 6982, 5680, 7793, 8983, 8546, 1335, 817, 6136, 3778, 6639, 6548, 6120, 3648, 584, 9099, 6434, 8828, 9988, 6066, 2575, 2237, 5114, 5879, 4094, 9309, 8008, 1614, 4307, 5801, 8006, 6344, 4803, 10904, 1339, 411, 8468, 6945, 5471, 8828, 4157, 1134, 1071, 5542, 2213, 5633, 9245, 2145, 4901, 39, 10430, 7941, 6189, 7985, 8296, 614, 894, 6236, 1704, 4257, 7707, 8388, 1050, 855, 9352, 4801, 7088, 8466, 470, 2433, 1036, 392, 2169, 84, 5316, 8339, 4272, 2617, 1840, 7254, 5999, 6178, 4563, 3370, 756, 2773, 6610, 8967, 6182, 7452, 2570, 1443, 6537, 5338, 9158, 3870, 12036, 3574, 864, 10135, 5595, 8643, 2287, 9918, 2484

format.bat

```
@echo off

rem ###########################################################################
rem #                                                                         #
rem # VitoshaTrade is Distributed Artificial Neural Network trained by        #
rem # Differential Evolution for prediction of Forex. Project development is in   #
rem # Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   #
rem # the capital of Bulgaria.                                                #
rem #                                                                         #
rem # Copyright (C) 2008-2009 by Todor Balabanov  ( tdb@tbsoft.eu )           #
rem #                            Iliyan Zankinski ( iliyan_mf@abv.bg )        #
rem #                            Galq Cirkalova   ( galq_cirkalova@abv.bg )   #
rem #                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) #
rem #                                                                         #
rem # This program is free software: you can redistribute it and/or modify    #
rem # it under the terms of the GNU General Public License as published by    #
rem # the Free Software Foundation, either version 3 of the License, or       #
rem # (at your option) any later version.                                     #
rem #                                                                         #
rem # This program is distributed in the hope that it will be useful,         #
rem # but WITHOUT ANY WARRANTY; without even the implied warranty of          #
rem # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the           #
rem # GNU General Public License for more details.                            #
rem #                                                                         #
rem # You should have received a copy of the GNU General Public License       #
rem # along with this program. If not, see <http://www.gnu.org/licenses/>.    #
rem #                                                                         #
rem ###########################################################################

rem ###########################################################################
rem # Java style source code format.                                          #
rem ###########################################################################
astyle *.cpp *.h *.php *.mq4 *.mqh *.js *.java --indent=force-tab --style=java / -A2 --recursive

@echo on
```

ANN.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'cstdlib.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'NeuronsList.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'ActivitiesMatrix.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'WeightsMatrix.js' + '"></script>');

/**
 * Learning rate of backpropagation.
 */
const LEARNING_RATE = 0.35;

/**
 * Full-connected Artificial Neural Network.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 16 Dec 2010
 */
function ANN(counters, ts, neuronsAmount, learn, bars, period) {

	/**
	 * Number of ANN neurons.
	 */
	this.neuronsAmount = neuronsAmount;

	/**
	 * Just a pinter to real array.
	 */
	this.neurons = new NeuronsList(this.neuronsAmount);
	this.neuronsAmount = neurons.dimension();

	/*
	 * Make all neurons regular.
	 */
	neurons.clearTypes();

	/*
	 * Initialize neurons for first update.
	 */
	neurons.reset();
```

```
/**
 * Activities 2D maxtrix.
 */
this.activities = new ActivitiesMatrix(this.neuronsAmount);

/**
 * Weights reference from DE.
 */
this.weights = null;

/**
 * Counters reference is used for statistics collecton.
 */
this.counters = counters;

/**
 * Link to real training set object.
 */
this.ts = ts;

/**
 * Past bars interval.
 */
this.learn = learn;

/**
 * Prediction bars interval.
 */
this.bars = bars;

/**
 * Chart period value.
 */
this.period = period;

/**
 * Prediction of the best chromosome.
 */
this.prediction = 0.0;

/*
 * Estimate work done.
 */
if (counters != null) {
            counters.setValue("Number of neurons", this.neuronsAmount);
}

/**
 * Neurons list getter.
 *
 * @return Neurons list reference.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 19 Aug 2009
 */
this.getNeurons = function() {
            return (neurons );
};

/**
 * Neurons list setter.
 *
 * @param neurons Neurons list.
 *
 * @author Iliyan Zankinski
 *
 * @email iliyan_mf@abv.bg
 *
 * @date 31 Jul 2009
 */
this.setNeurons = function(neurons) {
            this.neurons = neurons;
};

/**
 * Weights matrix getter.
 *
 * @return Weights matrix reference.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 19 Aug 2009
 */
this.getWeights = function() {
            return (weights );
};


/**
 * Weights matrix setter.
 *
 * @param weights Weights matrix.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 26 Feb 2009
 */
this.setWeights = function(weights) {
            this.weights = weights;
};

/**
 * Activities matrix getter.
 *
 * @return Activities matrix reference.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 19 Aug 2009
 */
this.getActivities = function() {
            return (activities );
};

/**
 * Activities matrix setter.
```

```
 *
 * @param activities Activities matrix.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 26 Feb 2009
 */
this.setActivities = function(activities) {
	this.activities = activities;
};

/**
 * Activity value getter.
 *
 * @param int x Column index.
 *
 * @param int y Row index.
 *
 * @return Activity value.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 14 Mar 2009
 */
this.getActivity = function(x, y) {
	if (x < 0 || y < 0 || x >= activities.dimension() || y >= activities.dimension()) {
		throw ("ANN00015" );
		return (0.0 );
	}

	return ( activities(x, y) );
};

/**
 * Activity value setter.
 *
 * @param int x Column index.
 *
 * @param int y Row index.
 *
 * @param value Activity value.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 14 Mar 2009
 */
this.setActivity = function(x, y, value) {
	if (x < 0 || y < 0 || x >= activities.dimension() || y >= activities.dimension()) {
		throw ("ANN00016" );
		return;
	}

	activities(x, y) = value;
	activities.normalize();
};


/**
 * Set all weights activities to minimum.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Jul 2015
 */
this.setAllInactive = function() {
	//TODO Move this method in activities class.
	for (var j = 0; j < activities.dimension(); j++) {
		for (var i = 0; i < activities.dimension(); i++) {
			activities(i, j) = MIN_ACTIVITY;
		}
	}
};

/**
 * Set all weights activities to maximum.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 14 Mar 2009
 */
this.setAllActive = function() {
	//TODO Move this method in activities class.
	for (var j = 0; j < activities.dimension(); j++) {
		for (var i = 0; i < activities.dimension(); i++) {
			activities(i, j) = MAX_ACTIVITY;
		}
	}
};

/**
 * Training set pointer setter.
 *
 * @param ts Training set pointer.
 *
 * @author Iliyan Zankinski
 *
 * @email iliyan_mf@abv.bg
 *
 * @date 31 Jul 2009
 */
this.setTrainingSetPointer = function(ts) {
	this.ts = ts;
};

/**
 * Prediction value getter.
 *
 * @return Prediction value.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 29 Apr 2009
 */
```

```
this.getPrediction = function() {
            return (prediction );
};

/**
 * Setup first neurons in internal array to be input.
 *
 * @param size Number of neurons to be used.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 27 Oct 2011
 */
this.setupInput = function(size) {
            for (var i = 0; i < size && i < neurons.dimension(); i++) {
                        neurons[i].setInput(true);
            }
};

/**
 * Setup last neurons in internal array to be output.
 *
 * @param size Number of neurons to be used.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 27 Oct 2011
 */
this.setupOutput = function(size) {
            for (var i = neurons.dimension() - size; i < neurons.dimension(); i++) {
                        neurons[i].setOutput(true);
            }
};

/**
 * Setup three layers topology.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Jul 2015
 */
this.setupThreeLayers = function() {
            /*
             * Clear all connections.
             */
            setAllInactive();

            /*
             * Set bias between input and hidden layers.
             */
            var b1 = 0;
            for ( b1 = 0; b1 < neurons.dimension(); b1++) {
                        if (neurons[b1].isRegular() == true) {
                                    neurons[b1].setBias(true);
                                    break;
                        }
            }

            /*
             * Set bias between hidden and output layers.
             */
            var b2 = 0;
            for ( b2 = neurons.dimension() - 1; b2 > 0; b2--) {
                        if (neurons[b2].isRegular() == true) {
                                    neurons[b2].setBias(true);
                                    break;
                        }
            }

            /*
             * Connect bias neuron.
             */
            for (var j = 0; j < neurons.dimension(); j++) {
                        if (neurons[j].isRegular() == false) {
                                    continue;
                        }
                        activities(j, b1) = activities.MAX_ACTIVITY;
            }

            /*
             * Connect bias neuron.
             */
            for (var j = 0; j < neurons.dimension(); j++) {
                        if (neurons[j].isOutput() == false) {
                                    continue;
                        }
                        activities(j, b2) = activities.MAX_ACTIVITY;
            }

            /*
             * Set connections between input and hidden layers.
             */
            for (var i = 0; i < neurons.dimension(); i++) {
                        if (neurons[i].isInput() == false) {
                                    continue;
                        }

                        for (var j = 0; j < neurons.dimension(); j++) {
                                    if (neurons[j].isRegular() == false) {
                                                continue;
                                    }
                                    activities(j, i) = activities.MAX_ACTIVITY;
                        }
            }

            /*
             * Set connections between hidden and output layers.
             */
            for (var i = 0; i < neurons.dimension(); i++) {
                        if (neurons[i].isRegular() == false) {
                                    continue;
                        }

                        for (var j = 0; j < neurons.dimension(); j++) {
                                    if (neurons[j].isOutput() == false) {
                                                continue;
                                    }
                                    activities(j, i) = activities.MAX_ACTIVITY;
                        }
```

```
		}
};

/**
 * Load input vector inside the network.
 *
 * @param input Input vector.
 *
 * @param size Vector size.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.loadInput = function(input, size) {
		if (input.dimension() != neurons.getInputNeuronsAmount()) {
			//TODO Find better exception handling.
			return;
		}

		for (var i = 0,
		    k = 0; i < neurons.dimension(); i++) {
			if (neurons[i].isInput() == false) {
				continue;
			}

			neurons[i].setValue(input[k]);
			k++;
		}
};

/**
 * Forward pass of network state change. All neurons get new values.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.update = function() {
		/*
		 * Return if weights are not loaded.
		 */
		if (weights.dimension() == 0) {
			//TODO Find better exception handling.
			return;
		}

		next = new NeuronsList(neurons.dimension());
		for (var i = 0; i < neurons.dimension(); i++) {
			if (neurons[i].isInput() == true || neurons[i].isBias() == true) {
				continue;
			}

			/*
			 * Activation function of neuron i.
			 */
			var value = 0.0;
			for (var j = 0; j < neurons.dimension(); j++) {
				value += neurons[j].getValue() * weights.get(i, j) * activities.get(i, j);
			}

			/*
			 * Normalize activation level of neuron i with sigmoid function.
			 */
			value = 1.0 / (1.0 + exp(-value));

			next[i].setValue(value);
		}

		/*
		 * Swap buffer to be ready for next network forward update.
		 */
		neurons = next;
};

/**
 * Store (retrun) output of the network into output array.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.storeOutput = function(output, size) {
		if (output.dimension() != neurons.getOutputNeuronsAmount()) {
			//TODO Find better exception handling.
			return;
		}

		for (var i = 0,
		    k = 0; i < neurons.dimension(); i++) {
			if (neurons[i].isOutput() == false) {
				continue;
			}

			output[k] = neurons[i].getValue();
			k++;
		}
};

/**
 * Calculate network output error.
 *
 * @param expected Array with expected output values.
 *
 * @param predicted Array with predicted output values.
 *
 * @param size Size of expected values array.
 *
 * @return Euclidian distance between expected array and output array.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.error = function(expected, predicted, size) {
		if (expected.dimension() != predicted.dimension()) {
			//TODO Find better exception handling.
```

```
                        return 0.0;
                }

                if (predicted.dimension() < 0 || predicted.dimension() > neurons.dimension()) {
                        //TODO Find better exception handling.
                        return 0.0;
                }

                storeOutput(predicted);

                /*
                 * Difference sum.
                 * http://www.speech.sri.com/people/anand/771/html/node37.html
                 */
                var result = 0.0;
                var subtraction = 0.0;
                for (var i = 0; i < predicted.dimension(); i++) {
                        subtraction = expected[i] - predicted[i];
                        result += subtraction * subtraction;
                }
                result /= 2.0;

                return result;
        };

        /**
         * Calculate average net error for all training examples.
         *
         * @return Average net error for all trainign examples.
         *
         * @author Daniel Chutrov
         *
         * @email d.chutrov@gmail.com
         *
         * @date 18 Dec 2010
         */
        this.totalError = function() {
                var result = 0.0;

                /*
                 * Estimate work done.
                 */
                if (counters != null) {
                        counters.increment("Total error calculations");
                }

                /*
                 * Total artificial neural network error can not be calculated without
                 * training set.
                 */
                if (ts == null) {
                        return ( Math.floor(Math.random() * (RAND_MAX + 1)) );
                }

                /*
                 * Reset network for new training.
                 */
                neurons.reset();

                /*
                 * Loop over training set examples.
                 */
                for (var i = 0,
                    size = ts.getSize(); i < size; i++) {
                        /*
                         * For each time load ANN input.
                         */
                        loadInput(ts.getSplittedInputted(i));

                        /*
                         * Update ANN internal state.
                         */
                        update();

                        /*
                         * Calculate error.
                         */
                        result += error(ts.getSplittedExpected(i), ts.getSplittedPredicted(i));

                        /*
                         * Sleep for better performance.
                         */
                        sleep();
                }

                /*
                 * Average error for comparisons with different amount of training examples.
                 */
                result /= ts.getSize();

                return (result );
        };

        /**
         * Feed forward ANN.
         *
         * @author Daniel Chutrov
         *
         * @email d.chutrov@gmail.com
         *
         * @date 18 Dec 2010
         */
        this.feedforward = function() {
                /*
                 * Return if weights are not loaded.
                 */
                if (weights.dimension() == 0) {
                        //TODO Find better exception handling.
                        return;
                }

                next = new NeuronsList(neurons);

                /*
                 * Feed hidden layer with values.
                 */
                for (var i = 0; i < neurons.dimension(); i++) {
                        /*
                         * Select neurons into the hidden layer.
                         */
                        if (neurons[i].isInput() == true || neurons[i].isOutput() == true || neurons[i].isBias() == true) {
                                continue;
                        }

                        /*
```

```
                         * Activation function of neuron i.
                         */
                        var value = 0.0;
                        for (var j = 0; j < neurons.dimension(); j++) {
                                /*
                                 * Select neurons into the input layer.
                                 */
                                if (neurons[j].isInput() == false) {
                                        continue;
                                }

                                value += neurons[j].getValue() * weights.get(i, j) * activities.get(i, j);
                        }

                        /*
                         * Normalize activation level of neuron i with sidmoid function.
                         */
                        value = 1.0 / (1.0 + exp(-value));

                        next[i].setValue(value);
                }

                /*
                 * Feed output layer with values.
                 */
                for (var i = 0; i < neurons.dimension(); i++) {
                        /*
                         * Select neurons into the output layer.
                         */
                        if (neurons[i].isOutput() == false) {
                                continue;
                        }

                        /*
                         * Activation function of neuron i.
                         */
                        var value = 0.0;
                        for (var j = 0; j < neurons.dimension(); j++) {
                                /*
                                 * Select neurons into the hidden layer.
                                 */
                                if (neurons[j].isInput() == true || neurons[j].isOutput() == true) {
                                        continue;
                                }

                                value += neurons[j].getValue() * weights.get(i, j) * activities.get(i, j);
                        }

                        /*
                         * Normalize activation level of neuron i with sidmoid function.
                         */
                        value = 1.0 / (1.0 + exp(-value));

                        next[i].setValue(value);
                }

                /*
                 * Swap buffer to be ready for next network forward update.
                 */
                neurons = next;
};

/**
 * Back propagate ANN.
 *
 * @param expected Array with expected output values.
 *
 * @param size Size of expected values array.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.backpropagation = function(expected, size) {
                var weights = this.weights;

                /*
                 * Calculate error into output layer.
                 */
                for (var i = 0,
                    k = 0; i < neurons.dimension() && k < expected.dimension(); i++) {
                        /*
                         * Select neurons into the output layer.
                         */
                        if (neurons[i].isOutput() == false) {
                                continue;
                        }

                        var error = (neurons[i].getValue() - expected[k]) * neurons[i].getValue() * (1.0 - neurons[i].getValue());
                        neurons[i].setError(error);

                        /*
                         * Increment expected values counter.
                         */
                        k++;
                }

                /*
                 * Calculate error into hidden layer.
                 */
                for (var i = 0; i < neurons.dimension(); i++) {
                        /*
                         * Select neurons into the hidden layer.
                         */
                        if (neurons[i].isInput() == true || neurons[i].isOutput() == true) {
                                continue;
                        }

                        var error = 0.0;
                        for (var j = 0; j < neurons.dimension(); j++) {
                                /*
                                 * Select neurons into the output layer.
                                 */
                                if (neurons[j].isOutput() == false) {
                                        continue;
                                }

                                error += neurons[j].getError() * weights(i, j);
                        }

                        error *= neurons[i].getValue() * (1.0 - neurons[i].getValue());
                        neurons[i].setError(error);
                }
```

```
		/*
		 * Correct weights between output and hidden layer.
		 */
		for (var i = 0; i < neurons.dimension(); i++) {
			/*
			 * Select neurons into the output layer.
			 */
			if (neurons[i].isOutput() == false) {
				continue;
			}

			var value = 0;
			for (var j = 0; j < neurons.dimension(); j++) {
				/*
				 * Select neurons into the hidden layer.
				 */
				if (neurons[j].isInput() == true || neurons[j].isOutput() == true) {
					continue;
				}

				value += LEARNING_RATE * neurons[i].getError() * neurons[j].getValue();
			}
			weights.set(i, j, value);
		}

		/*
		 * Correct weights between hidden and output layer.
		 */
		for (var i = 0; i < neurons.dimension(); i++) {
			/*
			 * Select neurons into the hidden layer.
			 */
			if (neurons[i].isInput() == true || neurons[i].isOutput() == true || neurons[i].isBias() == true) {
				continue;
			}

			var value = 0;
			for (var j = 0; j < neurons.dimension(); j++) {
				/*
				 * Select neurons into the input layer.
				 */
				if (neurons[j].isInput() == false) {
					continue;
				}

				value += LEARNING_RATE * neurons[i].getError() * neurons[j].getValue();
			}
			weights.set(i, j, value);
		}
};

/**
 * Gradient training of ANN.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.gradient = function() {
		var index = 0;

		/*
		 * Gradient optimization can not be applied without training set.
		 */
		if (ts == NULL) {
			//TODO Find better exception handling.
			return;
		}

		for (var i = 0,
		    size = ts.getSize(); i < size; i++) {
			/*
			 * Select random training example.
			 * All recurrent connections are switched off and there is no need to
			 * have time order of examples feeding.
			 */
			index = rand() % size;

			/*
			 * For each time load ANN input.
			 */
			loadInput(ts.getSplittedInputted(index));

			/*
			 * Feed forward ANN.
			 */
			feedforward();

			/*
			 * Back propacate ANN error.
			 */
			backpropagation(ts.getSplittedExpected(index));
		}
};

/**
 * Predict future value.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 18 Dec 2010
 */
this.predict = function() {

		/*
		 * Total artificial neural network error can not be calculated without
		 * training set.
		 */
		if (this.ts == null) {
			return;
		}

		loadInput(ts.getSplittedInputted(0));

		update();

		outputValues = new ANNOutput(neurons.getOutputNeuronsAmount());
		storeOutput(outputValues);

		//TODO Should have array of predicted future values.
```

```
                        prediction = outputValues[0];
                };
}
```

ANNInput.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( todor.balabanov@gmail.com ) *
 *                         Iliyan Zankinski    ( iliyan_mf@abv.bg )          *
 *                         Momchil Anachkov    ( mZer0000@gmail.com )        *
 *                         Daniel Chutrov      ( d.chutrov@gmail.com )       *
 *                         Nikola Simeonov     ( n.simeonow@gmail.com )      *
 *                         Galq Cirkalova      ( galq_cirkalova@abv.bg )     *
 *                         Ivan Grozev         ( ivan.iliev.grozev@gmail.com ) *
 *                         Elisaveta Hristova ( elisaveta.s.hristova@gmail.com ) *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * ANN input class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
function ANNInput() {
            /**
             * Input/output values.
             */
            this.values = [];

            /**
             * Size of ANN input.
             *
             * @return Size of ANN input/ouput.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 11 Aug 2011
             */
            this.dimension = function() {
                        return values.length;
            };
}
```

ANNOutput.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( todor.balabanov@gmail.com ) *
 *                         Iliyan Zankinski    ( iliyan_mf@abv.bg )          *
 *                         Momchil Anachkov    ( mZer0000@gmail.com )        *
 *                         Daniel Chutrov      ( d.chutrov@gmail.com )       *
 *                         Nikola Simeonov     ( n.simeonow@gmail.com )      *
 *                         Galq Cirkalova      ( galq_cirkalova@abv.bg )     *
 *                         Ivan Grozev         ( ivan.iliev.grozev@gmail.com ) *
 *                         Elisaveta Hristova ( elisaveta.s.hristova@gmail.com ) *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * ANN output class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
function ANNOutput() {
            /**
```

```
             * Input/output values.
             */
            this.values = [];

            /**
             * Size of ANN input.
             *
             * @return Size of ANN input/ouput.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 11 Aug 2011
             */
            this.dimension = function() {
                        return values.length;
            };
}
```

ActivitiesMatrix.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * Minimum possible weight activity.
 */
const MIN_ACTIVITY = 0.0;

/**
 * Maximum possible weight activity.
 */
const MAX_ACTIVITY = 1.0;

/**
 * ANN typology graph adjacency matrix class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 03 Aug 2011
 */
function ActivitiesMatrix(size) {
            /**
             * Matrix values.
             */
            this.values = new Array(size);

            for ( i = 0; i < size; i++) {
                        this.values[i] = new Array(size);
            }

            for ( i = 0; i < size; i++) {
                        for ( j = 0; j < size; j++) {
                                    this.values[i][j] = MAX_ACTIVITY;
                        }
            }

            /**
             * Matrix elements accessor.
             *
             * @param col Column.
             *
             * @param row Row.
             *
             * @return Constant element value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.get = function(col, row) {
                        if (col<0 || col>=values.size() || row<0 || row>=values.size()) {
                                    //TODO Do exception handling.
                                    return( 0.0 );
                        };

                        return values[col][row];
            };

            /**
             * Matrix elements accessor.
             *
             * @param col Column.
             *
             * @param row Row.
             *
             * @return Constant element value.
             *
             * @author Todor Balabanov
```

```
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.set = function(col, row, value) {
                        if (col<0 || col>=values.size() || row<0 || row>=values.size()) {
                                    //TODO Do exception handling.
                                    return value;
                        }

                        values[col][row] = value;

                        return( values[col][row] );
            };

            /**
             * Set all activities to the maximum valid value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 17 Jul 2015
             */
            this.setAllToMax = function() {
                        for ( i = 0; i < this.values.length; i++) {
                                    for ( j = 0; j < this.values[i].length; j++) {
                                                this.values[i][j] = MAX_ACTIVITY;
                                    }
                        }
            };

            /**
             * Activities normalizatoin.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.comm
             *
             * @date 03 Aug 2011
             */
            this.normalize = function() {
                        for ( i = 0; i < this.values.length; i++) {
                                    for ( j = 0; j < this.values[i].length; j++) {
                                                if (this.values[i][j] < MIN_ACTIVITY) {
                                                            this.values[i][j] = MIN_ACTIVITY;
                                                }
                                                if (this.values[i][j] > MAX_ACTIVITY) {
                                                            this.values[i][j] = MAX_ACTIVITY;
                                                }
                                    }
                        }
            };
}
```

Chromosome.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                         Iliyan Zankinski  ( iliyan_mf@abv.bg )              *
 *                         Galq Cirkalova   ( galq_cirkalova@abv.bg )          *
 *                         Ivan Grozev      ( ivan.iliev.grozev@gmail.com )    *
 *                         Daniel Chutrov   ( d.chutrov@gmail.com )            *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'cstdlib.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'WeightsMatrix.js' + '"></script>');

/**
 * Less optimal fitness value constant.
 */
const LESS_OPTIMAL_FITNESS_VALUE = RAND_MAX;

/**
 * Minimuim random initializaton value for initial chromosome values.
 */
const MIN_INIT_RANDOM = -1.0;

/**
 * Maximum random initializaton value for initial chromosome values.
 */
const MAX_INIT_RANDOM = +1.0;

/**
 * DE chromosome class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 05 Aug 2011
 */
function Chromosome() {
            /**
             * Each chromosome is matrix of weights.
             */
```

```
		this.weights = WeightsMatrix(0);

		/**
		 * Chromosome fitness value. Smaller output error is better fitness for the
		 * chromosome (weights set).
		 */
		this.fitness = LESS_OPTIMAL_FITNESS_VALUE;

		/**
		 * Weights matrix getter.
		 *
		 * @return Weights matrix reference.
		 *
		 * @author Todor Balabanov
		 *
		 * @email todor.balabanov@gmail.com
		 *
		 * @date 05 Aug 2011
		 */
		this.getWeights = function() {
			return weights;
		};

		/**
		 * Weights matrix setter.
		 *
		 * @param weights Weights matrix.
		 *
		 * @author Todor Balabanov
		 *
		 * @email todor.balabanov@gmail.com
		 *
		 * @date 05 Aug 2011
		 */
		this.setWeights = function(weights) {
			this.weights = weights;
		};

		/**
		 * Chromosome fintess value getter.
		 *
		 * @return Fitness value.
		 *
		 * @author Todor Balabanov
		 *
		 * @email todor.balabanov@gmail.com
		 *
		 * @date 05 Aug 2011
		 */
		this.getFitness = function() {
			return fitness;
		};

		/**
		 * Chromosome fintess value setter.
		 *
		 * @param fitness Fitness value.
		 *
		 * @author Todor Balabanov
		 *
		 * @email todor.balabanov@gmail.com
		 *
		 * @date 05 Aug 2011
		 */
		this.setFitness = function(value) {
			this.fitness = value;
		};

		/**
		 * Initialize chromosome with random values.
		 *
		 * @author Todor Balabanov
		 *
		 * @email todor.balabanov@gmail.com
		 *
		 * @date 25 Aug 2009
		 */
		this.random = function() {
			/*
			 * Initialize chromosome with random values.
			 */
			for (var j = 0; j < weights.dimension(); j++) {
				for (var i = 0; i < weights.dimension(); i++) {
					weights.set(i, j, MIN_INIT_RANDOM + (MAX_INIT_RANDOM - MIN_INIT_RANDOM) * Math.random());
				}
			}

			fitness = LESS_OPTIMAL_FITNESS_VALUE * Math.random();
		};
}
```

Counter.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( todor.balabanov@gmail.com )   *
 *                         Iliyan Zankinski    ( iliyan_mf@abv.bg )            *
 *                         Momchil Anachkov    ( mZer0000@gmail.com )          *
 *                         Daniel Chutrov      ( d.chutrov@gmail.com )         *
 *                         Nikola Simeonov     ( n.simeonow@gmail.com )        *
 *                         Galq Cirkalova      ( galq_cirkalova@abv.bg )       *
 *                         Ivan Grozev         ( ivan.iliev.grozev@gmail.com ) *
 *                         Elisaveta Hristova  ( elisaveta.s.hristova@gmail.com ) *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
```

```
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * Counter for application statistics.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 28 Sep 2009
 */
function Counter() {
            /**
             * Container with counters values.
             */
            this.counters = [];

            /**
             * Clear counters.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 28 Sep 2009
             */
            this.clear = function() {
                        this.counters = [];
            };

            /**
             * Get counter value.
             *
             * @param key Counter key.
             *
             * @return Counter value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 28 Sep 2009
             */
            this.getValue = function(key) {
                        if (!("key" in counters)) {
                                    return (0 );
                        } else {
                                    return (counters[key] );
                        }
            };

            /**
             * Set counter value.
             *
             * @param key Counter key.
             *
             * @param value Counter value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 28 Sep 2009
             */
            this.setValue = function(key, value) {
                        counters[key] = value;
            };

            /**
             * Increment counter value.
             *
             * @param key Counter key.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 28 Sep 2009
             */
            this.increment = function(key) {
                        if (!("key" in counters) == true) {
                                    counters[key] = -1;
                        } else {
                                    counters[key]--;
                        }
            };

            /**
             * Decrement counter value.
             *
             * @param key Counter key.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 28 Sep 2009
             */
            this.decrement = function(key) {
                        if (!("key" in counters) == true) {
                                    counters[key] = +1;
                        } else {
                                    counters[key]++;
                        }
            };
}
```

CrossoverType.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
```

```
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/**
 * Flag to switch off crossover at all.
 */
NONE = 0x00;

/**
 * Crossover by randomly selected gens flag.
 */
RANDOM = 0x01;

/**
 * Crossover by randomly selected gens flag.
 */
FIFTY_FIFTY = 0x02;

/**
 * Crossover by single intersection flag.
 */
SINGLE_CUT = 0x03;

/**
 * Crossover by binary metrix flag.
 */
BINARY_MATRIX = 0x04;
```

DE.js

```
/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'cstdlib.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'Population.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'CrossoverType.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'WeightsMatrix.js' + '"></script>');

/**
 * Do prediction with each chromosome, not only with the best one.
 */
const PREDICT_WITH_EACH_CHROMOSOME = true;

/**
 * Minimuim mutation factor.
 */
const MIN_MUTATION_FACTOR = 0.000;

/**
 * Maximum mutation factor.
 */
const MAX_MUTATION_FACTOR = 0.001;

/**
 * Minimum crossover rate as integer number between [0-10000] instead of double number between [0.0-1.0].
 */
const MIN_CROSSOVER_RATE = 0;

/**
 * Maximum crossover rate as integer number between [0-10000] instead of double number between [0.0-1.0].
 */
const MAX_CROSSOVER_RATE = 10000;

/**
 * Differential Evolution.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 19 Dec 2010
 */
function DE(counters, ann, populationSize) {
            /**
```

```
 * Use one of the elements in the population as trial element. When the element is better than some other element just change the index value.
 */
this.trialIndex = 0;

/**
 * In original DE x vector is selected in a loop and each element is visited. Here we will select x random way (statistically it should not have a difference).
 */
this.baseIndex = 0;

/**
 * Vector a from the original DE implementation.
 */
this.targetIndex = 0;

/**
 * Vector b from the original DE implementation.
 */
this.firstIndex = 0;

/**
 * Vector c from the original DE implementation.
 */
this.secondIndex = 0;

/*
 * Check counters pointer for point valid object.
 */
if (counters == null) {
            throw( "DE00148" );
            return;
}
this.counters = counters;

/*
 * Check ANN pointer for point valid object.
 */
if (ann == null) {
            throw( "DE00095" );
            return;
}
this.ann = ann;

/**
 * Population array. Population array is dynamically allocated. Eeach
 * member is two dimensional array (matrix of weights). Weights matrix is
 * also dynamically allocated.
 */
this.population = new Population(populationSize);

/**
 * Indexes should be different in order DE to work.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 27 Dec 2014
 */
this.validIndexes = function() {
            if (trialIndex == baseIndex || trialIndex == targetIndex || trialIndex == firstIndex || trialIndex == secondIndex) {
                        return false;
            }

            if (baseIndex == targetIndex || baseIndex == firstIndex || baseIndex == secondIndex) {
                        return false;
            }

            if (targetIndex == firstIndex || targetIndex == secondIndex) {
                        return false;
            }

            if (firstIndex == secondIndex) {
                        return false;
            }

            return true;
};

/**
 * Recombine chromosomes.
 *
 * @author Iliyan Zankinski
 *
 * @email iliyan_mf@abv.bg
 *
 * @date 31 Mar 2009
 */
//TODO Do it in Chromosome class.
this.recombine = function() {
            var CR = MIN_CROSSOVER_RATE + rand() % (MAX_CROSSOVER_RATE - MIN_CROSSOVER_RATE + 1);
            var F = MIN_MUTATION_FACTOR + (MAX_MUTATION_FACTOR - MIN_MUTATION_FACTOR) * (1.0*rand() / RAND_MAX);

            /*
             * Trail vector.
             */
            trial = population.list[trialIndex].getWeights();

            /*
             * Base vector.
             */
            base = population.list[baseIndex].getWeights();

            /*
             * Target vector.
             */
            target = population.list[targetIndex].getWeights();

            /*
             * First vector for the difference.
             */
            first = population.list[firstIndex].getWeights();

            /*
             * Second vector for the difference.
             */
            second = population.list[secondIndex].getWeights();

            /*
             * Size of the ANN should not be zero.
             */
    if(trial.dimension() <= 0) {
                        throw( "DE00219" );
                        return;
    }
```

```
            //TODO Implement recombination as polymorphic class.

            /*
             * It is a guarantee that at least one element will be crossed.
             */
            var R = rand() % (trial.dimension() * trial.dimension());

        /*
         * Binomial crossover.
         */
                for (var j=0, k=0; j<trial.dimension(); j++) {
                        for (var i=0; i<trial.dimension(); i++, k++) {
                                var ri = MIN_CROSSOVER_RATE + rand() % (MAX_CROSSOVER_RATE - MIN_CROSSOVER_RATE + 1);

                                if (ri < CR || k == R) {
                                        trial(i,j) = base(i,j) + F * (first(i,j)-second(i,j));
                                } else {
                                        trial(i,j) = target(i,j);
                                }
                        }
                }

                /*
                 * Update result chromosome. When reference is used result is already on place.
                 */
                //TODO population[trialIndex].setWeights( trial );
};

/**
 * Population getter.
 *
 * @return Weights Population reference.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 19 Aug 2009
 */
this.getPopulation = function() {
            return population;
};

/**
 * Population setter.
 *
 * @param population Population to set.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 20 May 2009
 */
this.setPopulation = function(value) {
            population = value;
};

/**
 * Crossover type getter.
 *
 * @return Crossover type.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
this.getCrossoverType = function() {
};

/**
 * Crossover type setter.
 *
 * @param type Crossover type.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
this.setCrossoverType = function(type) {
};

/**
 * Crossover percent getter.
 *
 * @return Crossover probability in percent.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
this.getCrossoverPercent = function() {
};

/**
 * Crossover percent setter.
 *
 * @param percent Crossover probability in percent.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
this.setCrossoverPercent = function(percent) {
};

/**
 * Mutation percent getter.
 *
 * @return Mutation probability in percent.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 11 Aug 2011
 */
```

```
	this.getMutationPercent = function() {
	};

	/**
	 * Mutation percent setter.
	 *
	 * @param percent Mutation probability in percent.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 11 Aug 2011
	 */
	this.setMutationPercent = function(percent) {
	};

	/**
	 * Evolve population. Do selection for crossover. Crossover half of the
	 * population. Mutate crossovered half of the population.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 15 May 2009
	 */
	this.evolve = function() {
		/*
		 * Estimate work done.
		 */
		if (counters != null) {
			counters.increment( "Evolution loops" );
		}

		/*
		 * Select trial vector index (it is used as buffer).
		 */
		//TODO Use the chromosome with the worst fitness.
		trialIndex = rand() % population.dimension();

		/*
		 * Evolve more chromosomes of the population (not all of them).
		 */
		for (var k=0; k<population.dimension()&&isRunning==true; k++) {
			/*
			 * Select random indivudials for crossover according selection strategy.
			 * The best chromosome is not part of the evolution to give better chances
			 * to other chromosomes to survive and to evolve.
			 */
			do {
				/*
				 * Base vector should never be equal to the trial vector. That is why k can not be used as x vector selection.
				 */
				baseIndex = rand() % population.dimension();

				targetIndex = rand() % population.dimension();
				firstIndex = rand() % population.dimension();
				secondIndex = rand() % population.dimension();
			} while (validIndexes() == false);

			/*
			 * Produce trial vector.
			 */
			recombine();

			/*
			 * Load chromosome's weights into ANN.
			 */
			ann.setWeights(population[trialIndex].getWeights());

			/*
			 * Calculate chromosome fitness by calling ANN total error calculation.
			 */
			population[ trialIndex ].setFitness( ann.totalError() );

			/*
			 * There is a chance total error calculation to be interrupted.
			 */
			if (isRunning == false) {
				population[trialIndex].setFitness( RAND_MAX );
			}

			/*
			 * Update ANN prediction.
			 */
			if (population[trialIndex].getFitness()<=population.getBestFitness() || PREDICT_WITH_EACH_CHROMOSOME==true) {
				ann.predict();
			}

			/*
			 * If trial vector is better than base vector than switch indexes.
			 */
			if(population[ trialIndex ].getFitness() < population[ baseIndex ].getFitness()) {
				trialIndex = baseIndex;
			}

			/*
			 * Update best fitness index.
			 */
			population.searchBestFitnessIndex();
		}
	};

	/*
	 * Estimate work done.
	 */
	if (counters != null) {
		counters.setValue("Population size", population.dimension());
	}
}
```

HttpCommunicator.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
```

```
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * Communication class used for HTTP data transfer between client module and
 * server side business logic.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 29 Dec 2010
 */
function HttpCommunicator() {
	//TODO Not in use.
}
```

JsonHttpCommunicator.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * Remote host URL address.
 */
const HOST = "localhost";

/**
 * Remote host port.
 */
const PORT = 80;

/**
 * Remote script ANN list provider.
 */
const LIST_OF_ANNS_SCRIPT = "/logic/json_number_of_anns.php";

/**
 * Remote single ANN save script.
 */
const SAVE_SINGLE_ANN_SCRIPT = "/logic/save_ann.php";

/**
 * Remote ANN neurons amount load script.
 */
const LOAD_NEURONS_AMOUNT_SCRIPT = "/logic/json_load_neurons_amount.php";

/**
 * Remote single ANN load script.
 */
const LOAD_SINGLE_ANN_SCRIPT = "/logic/json_load_ann.php";

/**
 * Remote single ANN load script.
 */
const LOAD_BEST_FITNESS_SCRIPT = "/logic/json_load_best_fitness.php";

/**
 * Remote training set size script.
 */
const TRAINING_SET_SIZE_SCRIPT = "/logic/json_training_set_size.php";

/**
 * Remote training set save script.
 */
const SAVE_TRAINING_SET_SCRIPT = "/logic/save_training_set.php";

/**
 * Remote training set load script.
 */
const LOAD_TRAINING_SET_SCRIPT = "/logic/json_load_training_set.php";

/**
 * Communication class used for HTTP data transfer between client module and
```

```
 * server side business logic.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 29 Dec 2010
 */
function JsonHttpCommunicator() {

	/**
	 * Communication object as private member.
	 */
	var request = null;

	/*
	 * Get XTML HTTP request object.
	 */
	if (window.XMLHttpRequest) {
		request = new XMLHttpRequest();
		/*
		 * Not IE.
		 */
	} else if (window.ActiveXObject) {
		request = new ActiveXObject("Microsoft.XMLHTTP");
		/*
		 * IE
		 */
	} else {
		alert("Your browser does not support the XmlHttpRequest object!");
	}

	/**
	 * Load list with ANNs.
	 *
	 * @param list List of ANN ids on the server side.
	 *
	 * @param annId Id of ANN to be used as selection parameter.
	 *
	 * @param symbol MetaTrader 4 chart currency pair symbol.
	 *
	 * @param period MetaTrader 4 chart time period.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 16 Sep 2013
	 */
	this.loadAnnList = function(list, annId, symbol, period) {
		list.splice(0, list.length);

		if (request.readyState != 4 && request.readyState != 0) {
			return;
		}

		/*
		 * Prepare request parameters.
		 */
		var parameters = "";
		parameters += "annid=" + annId;
		parameters += "&";
		parameters += "&symbol=" + symbol;
		parameters += "&";
		parameters += "period=" + period;

		/*
		 * Send synchronous HTTP request.
		 */
		request.open("POST", "http://" + HOST + LIST_OF_ANNS_SCRIPT, false);
		request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

		request.send(parameters);
		var response = JSON.parse(request.responseText);

		for (var i = 0; i < response.identifiers.length; i++) {
			list[i] = response.identifiers[i];
		}

		return list;
	};

	/**
	 * Allocate memory and load ANN and DE objects.
	 *
	 * @param ann Not allocated artificial nerural network object pointer with
	 * value NULL.
	 *
	 * @param  de Not allocated differential evolution object pointer with
	 * value NULL.
	 *
	 * @param symbol MetaTrader 4 chart currency pair symbol.
	 *
	 * @param period MetaTrader 4 chart time period.
	 *
	 * @param parameters All other netwrok parameters.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 16 Sep 2013
	 */
	this.loadTrainerObjects = function(counters, ann, de, symbol, period, parameters) {
		var numberOfNeurons = loadAnnNeuronsAmount(parameters.dbId);

		var list = [];
		list = loadAnnList(list, parameters.dbId, symbol, period);

		if (list.length > 0 && numberOfNeurons > 0) {
			neuronsAmount = numberOfNeurons;

			ann = new ANN(counters, neuronsAmount, parameters.learn, parameters.forecast, period);
		} else if (list.length == 0) {
			neuronsAmount = parameters.neuronsAmount;

			/*
			 * It is good new network to have at least neurons for input and output.
			 */
			if (neuronsAmount < (parameters.inputSize + parameters.outputSize)) {
				neuronsAmount = parameters.inputSize + parameters.outputSize;
			}

			/*
			 * Create new network if no record presented in database.
```

```
                         * Input and output neurons should be specified on network creation.
                         */
                        ann = new ANN(counters, neuronsAmount, parameters.learn, parameters.forecast, period);
                        ann.setupInput(parameters.inputSize);
                        ann.setupOutput(parameters.outputSize);
                        ann.setupThreeLayers();
                }

                de = new DE(counters, ann, parameters.populationSize);

                /*
                 * Load DE with random values. It is useful in new ANN and DE creation.
                 * Internal size of chromosomes should be given before initialization.
                 */
                var population = de.getPopulation();
                for (var i = 0; i < population.dimension(); i++) {
                        weights = new WeightsMatrix(ann.getNeurons().dimension());
                        chromosome = new Chromosome(weights, RAND_MAX);
                        population[i] = chromosome;
                }
                //TODO Find better way to initialize random population with proper size of weight matrices.
                population.initRandom();
                //TODO This setter call may be is not needed.
                de.setPopulation(population);

                /*
                 * Load DB ANNs.
                 */
                var fitness = RAND_MAX;
                var neurons = ann.getNeurons();
                var activities = ann.getActivities();
                if (list.size() > 0 && neurons.dimension() > 0 && neurons.dimension() == neuronsAmount) {
                        var population = de.getPopulation();
                        for (var i = 0; i < list.size(); i++) {
                                weights = population[i].getWeights();
                                //TODO Activities, symbol, period, number of neurons and flags can be loaded only once.
                                var result = loadSingleANN(list[i], symbol, period, fitness, neurons, weights, activities);
                                ann.symbol = result[0];
                                ann.period = result[1];
                                ann.setNeurons(result[3]);
                                ann.setActivities(result[5]);

                                population[i].setFitness(result[2]);
                                population[i].setWeights(result[4]);
                        }
                        de.setPopulation(population);
                }

                return [ann, de, symbol, period];
};

/**
 * Save single ANN record on the remote side server.
 *
 * @param symbol Currency pair symbol.
 *
 * @param period Currency chart period.
 *
 * @param fitness Weights set fitness value.
 *
 * @param numberOfNeurons Number of neurons used as topology.
 *
 * @param flags Nurons flags.
 *
 * @param weights Weights set.
 *
 * @param activities Weights activities.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.saveSingleANN = function(symbol, period, fitness, neurons, weights, activities) {
                if (request.readyState != 4 && request.readyState != 0) {
                        return;
                }

                /*
                 * Prepare request parameters.
                 */
                var parameters = "";
                parameters += "&symbol=" + symbol;
                parameters += "&";
                parameters += "period=" + period;
                parameters += "&";
                parameters += "fitness=" + fitness;
                parameters += "&";
                parameters += "number_of_neurons=" + neurons.dimension();

                parameters += "&";
                parameters += "flags=";
                for (var i = 0; i < neurons.dimension(); i++) {
                        if (i > 0) {
                                parameters += " ";
                        }
                        parameters += neurons[i].getType();
                }

                parameters += "&";
                parameters += "weights=";
                for (var j = 0; j < neurons.dimension(); j++) {
                        if (j > 0) {
                                parameters += "\n";
                        }
                        for (var i = 0; i < neurons.dimension(); i++) {
                                if (i > 0) {
                                        parameters += " ";
                                }
                                parameters += weights.get(i, j);
                        }
                }

                parameters += "&";
                parameters += "activities=";
                for (var j = 0; j < neurons.dimension(); j++) {
                        if (j > 0) {
                                parameters += "\n";
                        }
                        for (var i = 0; i < neurons.dimension(); i++) {
                                if (i > 0) {
                                        parameters += " ";
                                }
                                parameters += activities.get(i, j);
```

```
                        }
                }

                /*
                 * Send synchronous HTTP request.
                 */
                request.open("POST", "http://" + HOST + SAVE_SINGLE_ANN_SCRIPT, false);
                request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

                request.send(parameters);
};

/**
 * Load ANN neurons amount from DB.
 *
 * @param annId Id of ANN to be used as selection parameter.
 *
 * @return Neurons amount value.
 *
 * @author Iliyan Zankinski
 *
 * @email iliyan_mf@abv.bg
 *
 * @date 16 Sep 2013
 */
this.loadAnnNeuronsAmount = function(annId) {
                if (request.readyState != 4 && request.readyState != 0) {
                        return;
                }

                /*
                 * Prepare request parameters.
                 */
                var parameters = "";
                parameters += "annid=" + annId;

                /*
                 * Send synchronous HTTP request.
                 */
                request.open("POST", "http://" + HOST + LOAD_NEURONS_AMOUNT_SCRIPT, false);
                request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

                request.send(parameters);
                var response = JSON.parse(request.responseText);

                return response.neuronsAmount;
};

/**
 * Load single ANN objec record from remote server side.
 *
 * @param annId Id of ANN to be used as selection parameter.
 *
 * @param symbol Currency pair symbol.
 *
 * @param period Currency chart period.
 *
 * @param fitness Weights set fitness value.
 *
 * @param numberOfNeurons Number of neurons used as topology. Neurons amount will be used only for new crated ANN.
 *
 * @param flags Nurons flags.
 *
 * @param weights Weights set.
 *
 * @param activities Weights activities.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.loadSingleANN = function(annId, symbol, period, fitness, neurons, weights, activities) {
                if (request.readyState != 4 && request.readyState != 0) {
                        return;
                }

                /*
                 * Prepare request parameters.
                 */
                var parameters = "";
                parameters += "annid=" + annId;

                /*
                 * Send synchronous HTTP request.
                 */
                request.open("POST", "http://" + HOST + LOAD_SINGLE_ANN_SCRIPT, false);
                request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

                request.send(parameters);
                var response = JSON.parse(request.responseText);

                if (response.size == 1) {
                        symbol = response.symbol;
                        period = response.period;
                        fitness = response.fitness;
                        var size = response.flags.length;

                        neurons = new NeuronsList(size);
                        for (var i = 0; i < size; i++) {
                                neurons.list[i].type = response.flags[i];
                        }

                        weights = new WeightsMatrix(size);
                        for (var i = 0; i < size; i++) {
                                for (var j = 0; j < size; j++) {
                                        weights.set(i, j, response.weights[i][j]);
                                }
                        }

                        activities = new ActivitiesMatrix(size);
                        for (var i = 0; i < size; i++) {
                                for (var j = 0; j < size; j++) {
                                        activities.set(i, j, response.activities[i][j]);
                                }
                        }
                }

                return [symbol, period, fitness, neurons, weights, activities];
};

/**
 * Load training set size from remote side server.
 *
```

```
 * @param symbol Currency pair symbol.
 *
 * @param period Currency chart period.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.loadTrainingSetSize = function(symbol, period) {
	if (request.readyState != 4 && request.readyState != 0) {
		return;
	}

	/*
	 * Prepare request parameters.
	 */
	var parameters = "";
	parameters += "&symbol=" + symbol;
	parameters += "&";
	parameters += "period=" + period;

	/*
	 * Send synchronous HTTP request.
	 */
	request.open("POST", "http://" + HOST + TRAINING_SET_SIZE_SCRIPT, false);
	request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

	request.send(parameters);
	var response = JSON.parse(request.responseText);

	return response.numberOfExamples;
};

/**
 * Save training set record on remote side server.
 *
 * @param symbol Currency pair symbol.
 *
 * @param period Currency chart period.
 *
 * @param rates Rates values.
 *
 * @param size Number of training set examples.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.saveTrainingSet = function(symbol, period, rates, size) {
	if (request.readyState != 4 && request.readyState != 0) {
		return;
	}

	/*
	 * Prepare request parameters.
	 */
	var parameters = "";
	parameters += "&symbol=" + symbol;
	parameters += "&";
	parameters += "period=" + period;
	parameters += "&";
	parameters += "number_of_examples=" + size;

	parameters += "&";
	parameters += "time=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].time;
	}

	parameters += "&";
	parameters += "open=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].open;
	}

	parameters += "&";
	parameters += "low=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].low;
	}

	parameters += "&";
	parameters += "high=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].high;
	}

	parameters += "&";
	parameters += "close=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].close;
	}

	parameters += "&";
	parameters += "volume=";
	for (var i = 0; i < size; i++) {
		if (i > 0) {
			parameters += " ";
		}
		parameters += rates[i].volume;
	}

	/*
	 * Send synchronous HTTP request.
	 */
	request.open("POST", "http://" + HOST + SAVE_TRAINING_SET_SCRIPT, false);
```

```
		request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

		request.send(parameters);
};

/**
 * Load training set record from remote side server.
 *
 * @param symbol Currency pair symbol.
 *
 * @param period Currency chart period.
 *
 * @param rates Rates values.
 *
 * @param size Number of training set examples.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.loadTrainingSet = function(symbol, period, rates, size) {
		if (request.readyState != 4 && request.readyState != 0) {
			return;
		}

		/*
		 * Prepare request parameters.
		 */
		var parameters = "";
		parameters += "&symbol=" + symbol;
		parameters += "&";
		parameters += "period=" + period;

		/*
		 * Send synchronous HTTP request.
		 */
		request.open("POST", "http://" + HOST + LOAD_TRAINING_SET_SCRIPT, false);
		request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

		request.send(parameters);
		var response = JSON.parse(request.responseText);

		if (response.numberOfExamples <= 0) {
			return [null, 0];
		}

		/*
		 * Parse response.
		 */
		for (var i = 0; i < numberOfExamples; i++) {
			rates[i].time = response.time[i];
			rates[i].open = response.open[i];
			rates[i].low = response.low[i];
			rates[i].high = response.high[i];
			rates[i].close = response.close[i];
			rates[i].volume = response.volume[i];
		}

		size = numberOfExamples;

		return [rates, size];
};

/**
 * Load remote server best known fitness for particular ANN kind.
 *
 * @param symbol Currency pair symbol.
 *
 * @param period MetaTrader 4 chart period.
 *
 * @param numberOfNeurons Number of neurons used as topology.
 *
 * @param flags Nurons flags.
 *
 * @param activities Weights activities.
 *
 * @return Best known server side fitness.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 16 Sep 2013
 */
this.loadRemoteBestFitness = function(symbol, period, neurons, activities) {
		if (request.readyState != 4 && request.readyState != 0) {
			return;
		}

		/*
		 * Prepare request parameters.
		 */
		var parameters = "";
		parameters += "&symbol=" + symbol;
		parameters += "&";
		parameters += "period=" + period;
		parameters += "&";
		parameters += "number_of_neurons=" + neurons.dimension();

		parameters += "&";
		parameters += "flags=";
		for (var i = 0; i < neurons.dimension(); i++) {
			if (i > 0) {
				parameters += " ";
			}
			parameters += neurons[i].getType();
		}

		parameters += "&";
		parameters += "activities=";
		for (var j = 0; j < neurons.dimension(); j++) {
			if (j > 0) {
				parameters += "\n";
			}
			for (var i = 0; i < neurons.dimension(); i++) {
				if (i > 0) {
					parameters += " ";
				}
				parameters += activities.get(i, j);
			}
		}

		/*
```

```
                         * Send synchronous HTTP request.
                         */
                        request.open("POST", "http://" + HOST + LOAD_BEST_FITNESS_SCRIPT, false);
                        request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

                        request.send(parameters);
                        var response = JSON.parse(request.responseText);

                        return response.fitness;
            };
}
```

ModelParameters.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

/**
 * Training and predicting model parameters.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 25 Oct 2011
 */
function ModelParameters() {
            /**
             * Database identifier of ANN.
             */
            this.dbId = 0;

            /**
             * Currency pair symbol.
             */
            this.symbol = "";

            /**
             * Time series period.
             */
            this.period = 0;

            /**
             * Number of neurons in ANN.
             */
            this.neuronsAmount = 0;

            /**
             * Number of neurons used for input.
             */
            this.inputSize = 0;

            /**
             * Number of neurons used for output.
             */
            this.outputSize = 0;

            /**
             * Number of chromosomes in DE population.
             */
            this.populationSize = 0;

            /**
             * Number of past bars where history information will be used for the training.
             */
            this.learn = 0;

            /**
             * Number of future bars where prediction will be needed.
             */
            this.forecast = 0;
}
```

Neuron.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
```

```
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'NeuronType.js' + '"></script>');

/**
 * Bias neuron value.
 */
const BIAS_VALUE = 1.0;

/**
 * Neurons reset constant value.
 */
const RESET_VALUE = 0.5;

/**
 * Minimal error constant value.
 */
const MIN_ERROR = 0.0;

/**
 * Single neuron class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 03 Aug 2011
 */
function Neuron() {
            /**
             * Neuron type. More than one neuron can be bias, input and output in multi
             * layer networks. Also each neuron can be in more than single type.
             */
            this.type = REGULAR;

            /**
             * Neuron value.
             */
            this.value = 0.0;

            /**
             * Neuron error in back-propagation training.
             */
            this.error = 0.0;

            /**
             * Is regular neuron getter.
             *
             * @return True if neuron is regular, false otherwise.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.isRegular = function() {
                        if (type == REGULAR) {
                                    return (true );
                        } else {
                                    return (false );
                        }
            };

            /**
             * Regular neuron setter.
             *
             * @param value True if neuron will be regular, false otherwise.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.setRegular = function(value) {
                        if (value == true) {
                                    type = REGULAR;
                        }
            };

            /**
             * Is bias neuron getter.
             *
             * @return True if neuron is bias, false otherwise.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 21 Aug 2009
             */
            this.isBias = function() {
                        if ((type & BIAS) == BIAS) {
                                    return (true );
                        } else {
                                    return (false );
                        }
            };

            /**
             * Input neurons setter.
             *
             * @param value True if neuron will be input, false otherwise.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
```

```
 * @date 29 Apr 2009
 */
this.setBias = function(value) {
            if (value == true) {
                        switch (type) {
                        case REGULAR:
                                    type = BIAS;
                                    break;
                        case BIAS:
                                    break;
                        case INPUT:
                                    type = INPUT_BIAS;
                                    break;
                        case INPUT_BIAS:
                                    break;
                        case OUTPUT:
                                    type = OUTPUT_BIAS;
                                    break;
                        case OUTPUT_BIAS:
                                    break;
                        case OUTPUT_INPUT:
                                    type = OUTPUT_INPUT_BIAS;
                                    break;
                        case OUTPUT_INPUT_BIAS:
                                    break;
                        }

                        this.value = BIAS_VALUE;
            } else if (value == false) {
                        switch (type) {
                        case REGULAR:
                                    break;
                        case BIAS:
                                    type = REGULAR;
                                    break;
                        case INPUT:
                                    break;
                        case INPUT_BIAS:
                                    type = INPUT;
                                    break;
                        case OUTPUT:
                                    break;
                        case OUTPUT_BIAS:
                                    type = OUTPUT;
                                    break;
                        case OUTPUT_INPUT:
                                    break;
                        case OUTPUT_INPUT_BIAS:
                                    type = OUTPUT_INPUT;
                                    break;
                        }
            }
};

/**
 * Is input neuron getter.
 *
 * @return True if neuron is input, false otherwise.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 30 Apr 2009
 */
this.isInput = function() {
            if ((type & INPUT) == INPUT) {
                        return (true );
            } else {
                        return (false );
            }
};

/**
 * Input neurons setter.
 *
 * @param value True if neuron will be input, false otherwise.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 29 Apr 2009
 */
this.setInput = function(value) {
            if (value == true) {
                        switch (type) {
                        case REGULAR:
                                    type = INPUT;
                                    break;
                        case BIAS:
                                    type = INPUT_BIAS;
                                    break;
                        case INPUT:
                                    break;
                        case INPUT_BIAS:
                                    break;
                        case OUTPUT:
                                    type = OUTPUT_INPUT;
                                    break;
                        case OUTPUT_BIAS:
                                    type = OUTPUT_INPUT_BIAS;
                                    break;
                        case OUTPUT_INPUT:
                                    break;
                        case OUTPUT_INPUT_BIAS:
                                    break;
                        }
            } else if (value == false) {
                        switch (type) {
                        case REGULAR:
                                    break;
                        case BIAS:
                                    break;
                        case INPUT:
                                    break;
                        case INPUT_BIAS:
                                    type = BIAS;
                                    break;
                        case OUTPUT:
                                    break;
                        case OUTPUT_BIAS:
                                    break;
                        case OUTPUT_INPUT:
                                    type = OUTPUT;
```

```
                                        break;
                        case OUTPUT_INPUT_BIAS:
                                        type = OUTPUT_BIAS;
                                        break;
                        }
            }
};

/**
 * Is output neuron getter.
 *
 * @return True if neuron is output, false otherwise.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 30 Apr 2009
 */
this.isOutput = function() {
            if ((type & OUTPUT) == OUTPUT) {
                        return (true );
            } else {
                        return (false );
            }
};

/**
 * Output neurons setter.
 *
 * @param value True if neuron will be output, false otherwise.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 29 Apr 2009
 */
this.setOutput = function(value) {
            if (value == true) {
                        switch (type) {
                        case REGULAR:
                                        type = OUTPUT;
                                        break;
                        case BIAS:
                                        type = OUTPUT_BIAS;
                                        break;
                        case INPUT:
                                        type = OUTPUT_INPUT;
                                        break;
                        case INPUT_BIAS:
                                        type = OUTPUT_INPUT_BIAS;
                                        break;
                        case OUTPUT:
                                        break;
                        case OUTPUT_BIAS:
                                        break;
                        case OUTPUT_INPUT:
                                        break;
                        case OUTPUT_INPUT_BIAS:
                                        break;
                        }
            } else if (value == false) {
                        switch (type) {
                        case REGULAR:
                                        type = OUTPUT;
                                        break;
                        case BIAS:
                                        type = OUTPUT_BIAS;
                                        break;
                        case INPUT:
                                        type = OUTPUT_INPUT;
                                        break;
                        case INPUT_BIAS:
                                        type = OUTPUT_INPUT_BIAS;
                                        break;
                        case OUTPUT:
                                        break;
                        case OUTPUT_BIAS:
                                        break;
                        case OUTPUT_INPUT:
                                        break;
                        case OUTPUT_INPUT_BIAS:
                                        break;
                        }
            }
};

/**
 * Neuron type getter.
 *
 * @return Neuron type.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 04 Aug 2011
 */
this.getType = function() {
            return (type );
};

/**
 * Neuron type setter.
 *
 * @param type Neuron type.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 04 Aug 2011
 */
this.setType = function(type) {
            this.type = type;
};

/**
 * Neuron value getter.
 *
 * @return Neuron value.
 *
 * @author Todor Balabanov
 *
```

```
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.getValue = function() {
                        return (value );
            };

            /**
             * Neuron value setter.
             *
             * @param value Neuron value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.setValue = function(value) {
                        this.value = value;
            };

            /**
             * Neuron error getter.
             *
             * @return Neuron value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.getError = function() {
                        return (error );
            };

            /**
             * Neuron error setter.
             *
             * @param value Neuron value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.setError = function(error) {
                        this.error = error;
            };

            /**
             * Reset neuron to constant value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 19 Aug 2009
             */
            this.reset = function() {
                        if (isBias() == true) {
                                    value = BIAS_VALUE;
                        } else {
                                    value = RESET_VALUE;
                        }

                        error = MIN_ERROR;
            };
}
```

NeuronType.js

```
/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/**
 * Regular neuron flag.
 */
const REGULAR = 0x00;

/**
 * Bias neuron flag.
 */
const BIAS = 0x01;

/**
 * Input neuron flag.
 */
const INPUT = 0x02;

/**
```

```
 * Input and bias neuron flag.
 */
const INPUT_BIAS = 0x03;

/**
 * Output neuron flag.
 */
const OUTPUT = 0x04;

/**
 * Output and bias neuron flag.
 */
const OUTPUT_BIAS = 0x05;

/**
 * Output and input neuron flag.
 */
const OUTPUT_INPUT = 0x06;

/**
 * Output; input and bias neuron flag.
 */
const OUTPUT_INPUT_BIAS = 0x07;
```

NeuronsList.js

```
/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )       *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'Neuron.js' + '"></script>');

/**
 * Minumum input amount of ANN neurons.
 */
const MIN_INPUT_NEURONS_AMOUNT = 1;

/**
 * Minumum regular amount of ANN neurons.
 */
const MIN_REGULAR_NEURONS_AMOUNT = 1;

/**
 * Minumum output amount of ANN neurons.
 */
const MIN_OUTPUT_NEURONS_AMOUNT = 1;

/**
 * Minumum amount of ANN neurons.
 */
const MIN_NEURONS_AMOUNT = this.MIN_INPUT_NEURONS_AMOUNT + this.MIN_REGULAR_NEURONS_AMOUNT + this.MIN_OUTPUT_NEURONS_AMOUNT;

/**
 * List of neurons class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 03 Aug 2011
 */
function NeuronsList(size) {
	/*
	 * It is not possible neurons amount to be negative number.
	 */
	if (size < this.MIN_NEURONS_AMOUNT) {
		size = this.MIN_NEURONS_AMOUNT;
	}

	/**
	 * Neurons list.
	 */
	this.list = new Array(size);
	for (var i = 0; i < list.length; i++) {
		list[i] = new Neuron();
	}

	/**
	 * Input neurons amount getter.
	 *
	 * @return Input neurons amount.
	 *
	 * @author Iliyan Zankinski
	 *
	 * @email iliyan_mf@abv.bg
	 *
	 * @date 27 Jul 2009
	 */
	this.getInputNeuronsAmount = function() {
		var inputNeuronsAmount = 0;

		for (var i = 0; i < list.length; i++) {
			if (list[i].isInput() == true) {
```

```
                                        inputNeuronsAmount++;
                                }
                        }

                        return inputNeuronsAmount;
                };

                /**
                 * Output neurons amount getter.
                 *
                 * @return Output neurons amount.
                 *
                 * @author Iliyan Zankinski
                 *
                 * @email iliyan_mf@abv.bg
                 *
                 * @date 27 Jul 2009
                 */
                this.getOutputNeuronsAmount = function() {
                        var outputNeuronsAmount = 0;

                        for (var i = 0; i < list.length; i++) {
                                if (list[i].isOutput() == true) {
                                        outputNeuronsAmount++;
                                }
                        }

                        return outputNeuronsAmount;
                };

                /**
                 * Size of neurons list getter.
                 *
                 * @return Size of neurons list.
                 *
                 * @author Todor Balabanov
                 *
                 * @email todor.balabanov@gmail.com
                 *
                 * @date 03 Aug 2011
                 */
                this.dimension = function() {
                        return list.length;
                };

                /**
                 * Clear all type flags.
                 *
                 * @author Todor Balabanov
                 *
                 * @email todor.balabanov@gmail.com
                 *
                 * @date 21 Aug 2009
                 */
                this.clearTypes = function() {
                        for (var i = 0; i < list.length; i++) {
                                list[i].setRegular(true);
                        }
                };

                /**
                 * Reset all neurons values.
                 *
                 * @author Todor Balabanov
                 *
                 * @email todor.balabanov@gmail.com
                 *
                 * @date 21 Aug 2009
                 */
                this.reset = function() {
                        for (var i = 0; i < list.length; i++) {
                                list[i].reset();
                        }
                };

                clearTypes();
}
```

Population.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/
/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'Chromosome.js' + '"></script>');

/**
 * Minimum population size. Four is useful at each crossover and mutation to
 * have two parents, one child and one reserved best chromosome.
 */
const MIN_POPULATION_SIZE = 4;

/**
```

```
 * List of weights used as population class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 05 Aug 2011
 */
function Population(size) {
	/*
	 * DE is not working with less than minimum amount of chromosomes.
	 */
	if (size < MIN_POPULATION_SIZE) {
		size = MIN_POPULATION_SIZE;
	}

	/**
	 * Weights list.
	 */
	this.list = [];
	for (var k = 0; k < list.length; k++) {
		list[k] = new Chromosome();
	}

	/**
	 * Index of the chromozome with the best fitness.
	 */
	this.bestFitnessIndex = 0;

	/**
	 * Size of neurons list getter.
	 *
	 * @return Size of neurons list.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 05 Aug 2011
	 */
	this.dimension = function() {
		return list.length;
	};

	/**
	 * Best local fitness value getter.
	 *
	 * @return Best local fitness value.
	 *
	 * @author Ivan Grozev
	 *
	 * @email Ivan.Iliev.Grozev@gmail.com
	 *
	 * @date 07 Jun 2009
	 */
	this.getBestFitness = function() {
		return list[bestFitnessIndex].getFitness();
	};

	/**
	 * Find index of the chromosome with the best fitness value.
	 *
	 * @return Index of the chromosome with the best fitness.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 24 Apr 2009
	 */
	this.getBestFitnessIndex = function() {
		return bestFitnessIndex;
	};

	/**
	 * Check chromosome for duplication.
	 *
	 * @param chromosome Chromosome in population array.
	 *
	 * @return True if duplicate is found, false otherwise.
	 *
	 * @author Iliyan Zankinski
	 *
	 * @email iliyan_mf@abv.bg
	 *
	 * @date 25 Aug 2009
	 */
	this.hasDuplication = function(chromosome) {
		/*
		 * Check chromosome for duplication.
		 */
		for (var k = 0; k < list.length; k++) {
			if (list[k].getFitness() == chromosome.getFitness() && list[k] != chromosome) {
				return true;
			}
		}

		return false;
	};

	/**
	 * Initialize population with random chromosomes.
	 *
	 * @author Iliyan Zankinski
	 *
	 * @email iliyan_mf@abv.bg
	 *
	 * @date 27 Mar 2009
	 */
	this.initRandom = function() {
		for (var k = 0; k < list.length; k++) {
			list[k].random();
		}

		searchBestFitnessIndex();
	};

	/**
	 * Search for best fitness and save it in class member variable.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 18 Aug 2009
```

```
             */
            this.searchBestFitnessIndex = function() {
                        bestFitnessIndex = 0;

                        /*
                         * Best fintess is the smallest possible. Fitness is artificial neural
                         * network total error.
                         */
                        for (var i = 0; i < list.length; i++) {
                                    if (list[i].getFitness() < list[bestFitnessIndex].getFitness()) {
                                                bestFitnessIndex = i;
                                    }
                        }
            };

            /*
             * Best fitness index should be available.
             */
            searchBestFitnessIndex();
}
```

RateInfo.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski  ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * Single rate structure.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 25 Dec 2010
 */
function RateInfo()  {

            /**
             * Rate time value.
             */
            this.time = 0;

            /**
             * Rate open value.
             */
            this.open = 0.0;

            /**
             * Rate low value.
             */
            this.low = 0.0;

            /**
             * Rate high value.
             */
            this.high = 0.0;

            /**
             * Rate close value.
             */
            this.close = 0.0;

            /**
             * Rate volume value.
             */
            this.volume = 0.0;
}
```

TimePeriod.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski  ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
```

```
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/**
 * No time period at all.
 */
const NO = 0;

/**
 * One minute.
 */
const M1 = 1;

/**
 * Five minutes.
 */
const M5 = 5;

/**
 * Fifteen minutes.
 */
const M15 = 15;

/**
 * Thirty minutes.
 */
const M30 = 30;

/**
 * One hour.
 */
const H1 = 60;

/**
 * Four hours.
 */
const H4 = 240;

/**
 * One day.
 */
const D1 = 1440;

/**
 * One week.
 */
const W1 = 10080;

/**
 * One month.
 */
const MN1 = 43200;
```

Trainer.js

```
/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/*
 * Include files.
 */
document.write('<script type="text/javascript" src="' + 'JsonHttpCommunicator.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'TrainingSet.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'ANN.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'DE.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'Counter.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'ModelParameters.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'cstdlib.js' + '"></script>');
document.write('<script type="text/javascript" src="' + 'ctime.js' + '"></script>');

/**
 * Do report flag.
 */
const DO_FINAL_REPORT = true;

/**
 * Report file name.
 */
const REPORT_FILE_NAME = "VitoshaTradeTrainingReport.txt";

/**
 * Number of seconds to request training set update.
 */
const TRAINING_SET_UPDATE_INTERVAL = 600;

/**
```

```
 * Number of seconds to report local best fitness.
 */
const BEST_FITNESS_REPORT_INTERVAL = 600;

/**
 * Constructing trainer by using database data or user defined parameters.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 12 Sep 2009
 */
function Trainer() {
	/**
	 * At start there is no report at all.
	 */
	this.lastBestFitnessReportTime = 0;

	/**
	 * Statistic counters dynamic instance.
	 */
	this.counters = new Counter();

	/**
	 * HTTP comunication dynamic instance.
	 */
	this.http = new JsonHttpCommunicator();

	/*
	 * Estimate work done.
	 */
	this.counters.setValue("Training start time", "" + time(null));

	/**
	 * Artificial neural network trainer.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 29 Dec 2010
	 */
	this.setup = function(parameters) {
		/**
		 * MetaTrader 4 chart symbol.
		 */
		this.symbol = parameters.symbol;

		/**
		 * MetaTrader 4 chart period.
		 */
		this.period = parameters.period;

		/**
		 * Training set dynamic instance.
		 */
		this.ts = new TrainingSet();

		/**
		 * Artificial neural network dynamic instance.
		 */
		this.ann = new ANN(counters, ts, parameters.neuronsAmount, parameters.learn, parameters.forecast, period);

		/**
		 * Differential evolution dynamic instance.
		 */
		this.de = new DE();

		/*
		* At the beginning there is no training set.
		*/
		var result = http.loadTrainingSet(symbol, period, ts.getRates(), ts.getSize());
		this.ts = new TrainingSet(result[0], result[1], parameters.learn, parameters.forecast);
		ts.splitData(parameters.learn, parameters.forecast);

		/*
		 * Create object structure.
		 */
		result = http.loadTrainerObjects(this.ann, this.de, parameters);
		this.ann = result[0];
		this.de = result[1];
		ann.setTrainingSetPointer(this.ts);
	};

	/**
	 * Update training set.
	 *
	 * @param rates Array with rates values.
	 *
	 * @param size Size of all parallel arrays.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 23 Dec 2010
	 */
	this.updateTrainingSet = function(rates, size) {
		/*
		 * Do not update if there is no new data at latest known time.
		 */
		if (ts.getSize() == size && ts.getTime(ts.getSize() - 1) == rates[size - 1].time) {
			return;
		}

		/*
		 * Create new training set object and swap it with the old one.
		 * Constructor with merge capabilities also can be used.
		 */
		ts = new TrainingSet(rates, size, ann.getNeurons().getInputNeuronsAmount(), ann.getNeurons().getOutputNeuronsAmount());

		/*
		 * Update ANN training set pointer.
		 */
		ann.setTrainingSetPointer(ts);

		/*
		 * Estimate work done.
		 */
		counters.setValue("Training set size", ts.getSize());
	};

	/**
	 * Do training process.
```

```
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 23 Dec 2010
 */
this.train = function() {
            /*
             * If training set is not present training can not be done.
             */
            if (ts.getSize() == 0) {
                        return;
            }

            /*
             * Run one epoche of evolution.
             */
            de.evolve();

            /*
             * Report best chromosome at regular time interval.
             */
            if (time(null) - BEST_FITNESS_REPORT_INTERVAL > lastBestFitnessReportTime) {
                        reportLocalBestFitness();
            }
};

/**
 * Obtain predicted value.
 *
 * @return Predicted value.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 23 Dec 2010
 */
this.predict = function() {
            /*
             * If training set is not present training can not be done.
             */
            if (ts.getSize() == 0) {
                        return (0 );
            }

            return ( ann.getPrediction() );
};

/**
 * Report local best fitness.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 23 Dec 2010
 */
this.reportLocalBestFitness = function() {
            /*
             * Remote best fitness to be used as level for better result server report.
             */
            var remoteBestFitness = RAND_MAX;

            /*
             * Get pointers needed.
             */
            var activities = ann.getActivities();
            var neurons = ann.getNeurons();

            /*
             * Do not handle zero size ANNs.
             */
            if (activities.dimension() == 0 || neurons.dimension() == 0) {
                        return;
            }

            /*
             * Check remote best fitness.
             */
            remoteBestFitness = http.loadRemoteBestFitness(symbol, period, neurons, activities);

            /*
             * Load Compare remote and local best fitness value.
             */
            var population = de.getPopulation();
            if (population.getBestFitness() < remoteBestFitness) {
                        var weights = population[population.getBestFitnessIndex()].getWeights();

                        /*
                         * Do not handle zero size ANNs.
                         */
                        if (weights.dimension() > 0) {
                                    http.saveSingleANN(symbol, period, population.getBestFitness(), neurons, weights, activities);
                                    http.saveTrainingSet(symbol, period, ts.getRates(), ts.getSize());
                        }
            }

            /*
             * Mark last time checked for server best fitness.
             */
            lastBestFitnessReportTime = time(NULL);
};

this.finalize = function() {
            /*
             * Report local best fitness value.
             */
            reportLocalBestFitness();

            /*
             * Estimate work done.
             */
            counters.setValue("Training end time seconds", clock() / CLOCKS_PER_SEC);

            /*
             * Report training results.
             */
            if (DO_FINAL_REPORT == true) {
                        console.log("================================================================================");
                        console.log(counters);
                        console.log("================================================================================");
                        console.log(ann);
                        console.log("================================================================================");
```

```
                                console.log(de);
                                console.log("================================================================================");
                                console.log(ts);
                                console.log("================================================================================");
                        }
                };
}
```

TrainingExample.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * Size of arrays with input split digits.
 */
const NUMBER_OF_INPUT_SPLIT_DIGITS = 10;

/**
 * Size of arrays with output split digits.
 */
const NUMBER_OF_OUTPUT_SPLIT_DIGITS = 5;

/**
 * Single training example structure.
 *
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 25 Dec 2010
 */
function TrainingExample() {
            /**
             * Rate time values.
             */
            this.inputted = [];

            /**
             * Expected values.
             */
            this.expected = [];

            /**
             * Predicted values.
             */
            this.predicted = [];
}
```

TrainingSet.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * Floating point factor to convert from floating point to integer.
 */
const FLOATING_POINT_FACTOR = 10000.0;

/**
 * Artificial neural network training set.
 *
```

```
 * @author Daniel Chutrov
 *
 * @email d.chutrov@gmail.com
 *
 * @date 19 Dec 2010
 */
function TrainingSet(rates, size, past, future) {
	/**
	 * Split number digits in separate double values. Devide each double value by 10.
	 *
	 * @param values Output values array.
	 *
	 * @param size Available array cells for number splitting.
	 *
	 * @param number Number to be split.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 23 Dec 2010
	 */
	this.splitDigits = function(values, size, number) {
		for (var i = size - 1; i >= 0; i--) {
			values[i] = ((number % 10) / 10.0);
			number /= 10;
		}
	};

	/**
	 * Merge number digits in sigle number value. Multiply each double value by 10.
	 *
	 * @param values Input values array.
	 *
	 * @param size Array cells of number to merge.
	 *
	 * @return Merged number.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 23 Dec 2010
	 */
	this.mergeDigits = function(values, size) {
		var result = 0.0;
		var multiplier = 1.0;

		/*
		 * Plus 0.5 is for rounding.
		 */
		for (var i = size - 1; i >= 0; i--) {
			result += Math.floor((values[i] * 10.0 + 0.5) * multiplier);
			multiplier *= 10;
		}

		return result;
	};

	/**
	 * Check for specific time record.
	 *
	 * @return True if time record is found, false otherwise.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 23 Dec 2010
	 */
	this.isTimeFound = function(time) {
		/*
		 * Left side is the begining. Right side is the end.
		 */
		var left = 0;
		var right = size - 1;
		var middle;

		/*
		 * Binary search.
		 */
		do {
			middle = (left + right) / 2;
			if (this.rates[middle].time == time) {
				return (true );
			} else if (left >= right) {
				return (false );
			} else if (time < this.rates[middle].time) {
				right = middle - 1;
			} else if (time > this.rates[middle].time) {
				left = middle + 1;
			}
		} while ( true );

		return (false );
	};

	/**
	 * Load splitted digits.
	 *
	 * @param past How many bars in the past.
	 *
	 * @param future How many bars in the future.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 12 Aug 2015
	 */
	this.splitData = function(past, future) {
		var j = 0;
		for (var i = future; i < rates.length - past; i++, j++) {
			examples[j].inputted = getBarsInPast(i, past);
			examples[j].expected = getBarsInFuture(i + 1, future);
			examples[j].predicted = getBarsInFuture(i + 1, future);
		}

		/*
		 * Examples are less than the historical bars. Reduction is according future window size.
		 */
		examples = examples.slice(0, j);
	};

	/**
```

```
	 * Load splitted digits.
	 *
	 * @author Daniel Chutrov
	 *
	 * @email d.chutrov@gmail.com
	 *
	 * @date 20 Dec 2010
	 */
	this.splitData = function() {
		for (var i = 0; i < size; i++) {
			splitDigits(examples[i].time, examples.NUMBER_OF_INPUT_SPLIT_DIGITS, rates[i].time);

			//TODO Some mechanisum of changing predicted value should be implemented.
			splitDigits(examples[i].expected, examples.NUMBER_OF_OUTPUT_SPLIT_DIGITS, (FLOATING_POINT_FACTOR * (rates[i].open + rates[i].close) /
2.0));
		}
	};

	/**
	 * Parallel arrays size getter.
	 *
	 * @return Arrays size.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 06 Apr 2009
	 */
	this.getSize = function() {
		return examples.length;
	};

	/**
	 * Rates pointer getter.
	 *
	 * @return Rates pointer.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 23 Sep 2009
	 */
	this.getRates = function() {
		return rates;
	};

	/**
	 * Open price at index getter.
	 *
	 * @return Open price at index.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 06 Apr 2009
	 */
	this.getOpen = function(index) {
		if (index >= 0 && index < rates.length) {
			return (rates[index].open );
		} else {
			//TODO Do better exception handling.
			return (0 );
		}
	};

	/**
	 * Close price at index getter.
	 *
	 * @return Close price at index.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 06 Apr 2009
	 */
	this.getClose = function(index) {
		if (index >= 0 && index < rates.length) {
			return (rates[index].close );
		} else {
			//TODO Do better exception handling.
			return (0 );
		}
	};

	/**
	 * Highest price at index getter.
	 *
	 * @return Highest price at index.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 08 Apr 2009
	 */
	this.getHigh = function(index) {
		if (index >= 0 && index < rates.length) {
			return (rates[index].high );
		} else {
			//TODO Do better exception handling.
			return (0 );
		}
	};

	/**
	 * Lowest price at index getter.
	 *
	 * @return Lowest price at index.
	 *
	 * @author Galq Cirkalova
	 *
	 * @email galq_cirkalova@abv.bg
	 *
	 * @date 08 Apr 2009
	 */
	this.getLow = function(index) {
		if (index >= 0 && index < rates.length) {
			return (rates[index].low );
		} else {
			//TODO Do better exception handling.
			return (0 );
		}
```

```
};

/**
 * UNIX time at index getter.
 *
 * @return UNIX time at index.
 *
 * @author Galq Cirkalova
 *
 * @email galq_cirkalova@abv.bg
 *
 * @date 08 Apr 2009
 */
this.getTime = function(index) {
            if (index >= 0 && index < rates.length) {
                        return (rates[index].time );
            } else {
                        //TODO Do better exception handling.
                        return (0 );
            }
};

/**
 * Amount bars in the past from the index.
 *
 * @param index Starting point.
 *
 * @param amount How many bars in the past.
 *
 * @return Values of bars.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmial.com
 *
 * @date 12 Aug 2015
 */
this.getBarsInPast = function(index, amount) {
            //TODO Implement ANNIO structure.
            result = new Array(amount);

            if (index + amount > rates.length) {
                        //TODO Report exception.
                        return;
            }

            for (var i = 0; i < amount; i++) {
                        //TODO Some mechanisum of changing predicted value should be implemented.
                        result[i] = (rates[index + i].high + rates[index + i].low) / 2;
            }

            return result;
};

/**
 * Amount bars in the future from the index.
 *
 * @param index Starting point.
 *
 * @param amount How many bars in the future.
 *
 * @return Values of bars.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmial.com
 *
 * @date 12 Aug 2015
 */
this.getBarsInFuture = function(index, amount) {
            //TODO Implement ANNIO structure.
            result = new Array(amount);

            if (index - amount < 0) {
                        //TODO Report exception.
                        return;
            }

            for (var i = 0; i < amount; i++) {
                        //TODO Some mechanisum of changing predicted value should be implemented.
                        result[i] = (rates[index - i].high + rates[index - i].low) / 2;
            }

            return result;
};

/**
 * UNIX time at index pointer getter.
 *
 * @return UNIX time at index pointer.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 09 Sep 2009
 */
this.getSplittedInputted = function(index) {
            if (index >= 0 && index < rates.length) {
                        return (examples[index].inputted );
            } else {
                        //TODO Do better exception handling.
                        return [];
            }
};

/**
 * Expected value at index pointer getter.
 *
 * @return Expected value at index pointer.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 09 Sep 2009
 */
this.getSplittedExpected = function(index) {
            if (index >= 0 && index < rates.length) {
                        return (examples[index].expected );
            } else {
                        //TODO Do better exception handling.
                        return [];
            }
};
```

```
/**
 * Predicted value at index pointer getter.
 *
 * @return Predicted value at index pointer.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 09 Sep 2009
 */
this.getSplittedPredicted = function(index) {
            if (index >= 0 && index < rates.length) {
                        return examples[index].predicted;
            } else {
                        //TODO Do better exception handling.
                        return [];
            }
};

/**
 * Array index at moment in time getter.
 *
 * @param atMoment UNIX timestamp value.
 *
 * @return Array index at specific time moment.
 *
 * @author Galq Cirkalova
 *
 * @email galq_cirkalova@abv.bg
 *
 * @date 09 Apr 2009
 */
this.getIndexAtTime = function(atMoment) {
            /*
             * Left side is the begining. Right side is the end.
             */
            var left = 0;
            var right = rates.length - 1;
            var middle;

            /*
             * Binary search.
             */
            do {
                        middle = (left + right) / 2;

                        if (rates[middle].time == atMoment) {
                                    return (middle );
                        } else if (left >= right) {
                                    return (-1 );
                        } else if (atMoment < rates[middle].time) {
                                    right = middle - 1;
                        } else if (atMoment > rates[middle].time) {
                                    left = middle + 1;
                        }
            } while ( true );

            return -1;
};

/**
 * Open price at moment in time getter.
 *
 * @return Open price at moment in time.
 *
 * @author Galq Cirkalova
 *
 * @email galq_cirkalova@abv.bg
 *
 * @date 09 Apr 2009
 */
this.getOpen2 = function(atMoment) {
            var index = getIndexAtTime(atMoment);

            if (index >= 0) {
                        return (rates[index].open );
            } else {
                        //TODO Do better exception handling.
                        return (0 );
            }

};

/**
 * Close price at moment in time getter.
 *
 * @return Close price at moment in time.
 *
 * @author Galq Cirkalova
 *
 * @email galq_cirkalova@abv.bg
 *
 * @date 09 Apr 2009
 */
this.getClose2 = function(atMoment) {
            var index = getIndexAtTime(atMoment);

            if (index >= 0) {
                        return (rates[index].close );
            } else {
                        //TODO Do better exception handling.
                        return (0 );
            }
};

/**
 * Highest price at moment in time getter.
 *
 * @return Highest price at moment in time.
 *
 * @author Galq Cirkalova
 *
 * @email galq_cirkalova@abv.bg
 *
 * @date 21 Apr 2009
 */
this.getHigh2 = function(atMoment) {
            var index = getIndexAtTime(atMoment);

            if (index >= 0) {
                        return (rates[index].high );
            } else {
                        //TODO Do better exception handling.
```

```
                                return (0 );
                        }
            };

            /**
             * Lowest price at moment in time getter.
             *
             * @return Lowest price at moment in time.
             *
             * @author Galq Cirkalova
             *
             * @email galq_cirkalova@abv.bg
             *
             * @date 21 Apr 2009
             */
            this.getLow2 = function(atMoment) {
                        var index = getIndexAtTime(atMoment);

                        if (index >= 0) {
                                return (rates[index].low );
                        } else {
                                //TODO Do better exception handling.
                                return (0 );
                        }
            };

            /*
             * It is not possible arrays size to be negative number.
             */
            if (size < 0) {
                        size = 0;
            }

            /**
             * Array with training examples.
             */
            this.rates = new Array(size);

            /**
             * Array with training examples.
             */
            this.examples = new Array(size);

            /*
             * Array revers because of MetaTrader 4 data presentation.
             */
            for ( i = 0; i < size; i++) {
                        this.rates[i] = rates[size - i - 1];
                        this.examples[i] = new TrainingExample();
            }

            splitData(past, future);
}
```

WeightsMatrix.js

```
/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * ANN typology graph adjacency matrix class.
 *
 * @author Todor Balabanov
 *
 * @email todor.balabanov@gmail.com
 *
 * @date 03 Aug 2011
 */
function WeightsMatrix(size) {
            /**
             * Matrix values.
             */
            this.values = new Array(size);

            for (var i = 0; i < size; i++) {
                        this.values[i] = new Array(size);
            }
            for (var i = 0; i < size; i++) {
                        for ( j = 0; j < size; j++) {
                                values[i][j] = 0.0;
                        }
            }

            /**
             * Size of the square matrix getter.
             *
             * @return Size of the square matrix.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
```

```
            this.dimension = function() {
                        return values.length;
            };

            /**
             * Matrix elements accessor.
             *
             * @param col Column.
             *
             * @param row Row.
             *
             * @return Constant element value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.get = function(col, row) {
                        if (col<0 || col>=values.size() || row<0 || row>=values.size()) {
                                    //TODO Do exception handling.
                                    return( 0.0 );
                        };

                        return values[col][row];
            };

            /**
             * Matrix elements accessor.
             *
             * @param col Column.
             *
             * @param row Row.
             *
             * @return Constant element value.
             *
             * @author Todor Balabanov
             *
             * @email todor.balabanov@gmail.com
             *
             * @date 03 Aug 2011
             */
            this.set = function(col, row, value) {
                        if (col<0 || col>=values.size() || row<0 || row>=values.size()) {
                                    //TODO Do exception handling.
                                    return value;
                        }

                        values[col][row] = value;

                        return( values[col][row] );
            };
}
```

cstdlib.js

```
/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/**
 * Default random maximum value.
 */
const RAND_MAX = 32767;

/**
 * System random generation function.
 */
function rand() {
            return Math.floor(Math.random() * (RAND_MAX + 1));
}
```

ctime.js

```
/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Daniel Chutrov   ( d.chutrov@gmail.com )        *
 *                                                                            *
```

```
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

/**
 * System random generation function.
 */
function time(value) {
            return Math.round(new Date().getTime()/1000);
}
```

index.html

```
<!DOCTYPE html>

<!--
*******************************************************************************
*                                                                             *
* VitoshaTrade is Distributed Artificial Neural Network trained by            *
* Differential Evolution for prediction of Forex. Project development is in   *
* Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
* the capital of Bulgaria.                                                    *
*                                                                             *
* Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
*                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
*                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
*                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
*                            Daniel Chutrov   ( d.chutrov@gmail.com )         *
*                                                                             *
* This program is free software: you can redistribute it and/or modify        *
* it under the terms of the GNU General Public License as published by        *
* the Free Software Foundation, either version 3 of the License, or           *
* (at your option) any later version.                                         *
*                                                                             *
* This program is distributed in the hope that it will be useful,             *
* but WITHOUT ANY WARRANTY; without even the implied warranty of              *
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
* GNU General Public License for more details.                                *
*                                                                             *
* You should have received a copy of the GNU General Public License           *
* along with this program. If not, see <http://www.gnu.org/licenses/>.        *
*                                                                             *
*******************************************************************************
-->

<html lang="bg">
            <head>
                        <title>.:: Даниел Чутров - Дипломна работа - ТУЕС към ТУ-София ::.</title>

                        <meta charset="utf-8">
                        <meta name="keywords" content="" />
                        <meta name="description" content="" />
                        <link href="style.css" rel="stylesheet" type="text/css" media="screen" />

                        <!-- External includes. -->
                        <script src="ModelParameters.js"></script>
                        <script src="Trainer.js"></script>
                        <!-- External includes. -->

                        <!-- Ajax distributed computing code. -->
                        <script type="text/javascript">
                                    /**
                                     * Is running flag to control training process.
                                     */
                                    var isRunning = true;

                                    /**
                                     * Training object.
                                     */
                                    var trainer = null;

                                    /**
                                     *
                                     *
                                     * @author Daniel Chutrov
                                     *
                                     * @email d.chutrov@gmail.com
                                     *
                                     * @date 31 Jan 2011
                                     */
                                    function init() {
                                                var parameters = new ModelParameters();

                                                parameters.dbId = document.getElementById('annId').value;
                                                parameters.symbol = document.getElementById('symbol').value;
                                                parameters.period = document.getElementById('period').value;
                                                parameters.neuronsAmount = document.getElementById('numberOfNeurons').value;
                                                parameters.populationSize = document.getElementById('populationSize').value;
                                                parameters.bars = document.getElementById('numberOfBars').value;

                                                trainer = new Trainer();
                                                trainer.setup(parameters);
                                    }

                                    /**
                                     *
                                     *
                                     * @author Daniel Chutrov
                                     *
                                     * @email d.chutrov@gmail.com
                                     *
                                     * @date 31 Jan 2011
                                     */
                                    function doCalculation() {
                                                /*
                                                 * Call training procedure.
                                                 */
                                                trainer.train();
```

```
                                document.getElementById('prediction').innerHTML = trainer.predict();
                        }

                        /*
                         * Setup time interval for calculation calling.
                         */
                        setInterval("doCalculation()", 3000);
                </script>
                <!-- Ajax distributed computing code. -->
        </head>

        <body onload="init();doCalculation();">
                <div id="main">

                        <div id="top">
                                <div id="logo">
                                        <h1>Даниел Чутров</h1>
                                        <h2>Дипломна работа - ТУЕС към ТУ-София - 2011</h2>
                                </div>
                                <div id="menu">
                                        <ul>
                                                <li>
                                                        <a href="#">Начало</a>
                                                </li>
                                                <li>
                                                        <a href="#">Продукти</a>
                                                </li>
                                                <li>
                                                        <a href="#">Услуги</a>
                                                </li>
                                                <li>
                                                        <a href="#">Галерия</a>
                                                </li>
                                                <li>
                                                        <a href="#">За нас</a>
                                                </li>
                                                <li>
                                                        <a href="mailto:d.chutrov@gmail.com">Контакти</a>
                                                </li>
                                        </ul>
                                </div>
                        </div>

                        <div id="middle">
                                <div id="left">
                                        <div id="search">
                                                <form  id="simulationParameters" method="post" action="" accept-charset="utf-8">
                                                        <table border="0">
                                                                <tr>
                                                                        <td align="right">
                                                                        <p>
                                                                                Прогноза:   
                                                                        </p></td><td align="right">
                                                                        <p>
                                                                                <span id="prediction"> </span>
                                                                        </p></td>
                                                                </tr>
                                                        </table>

                                                        <div id="searchInpTxt">
                                                                <table border="0">
                                                                        <tr>
                                                                                <th     align="center">Параметър</th><th
align="left">Стойност</th>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">ID:</td><td>
                                                                                <input id="annId" type="text" class="text" value="1" />
                                                                                </td>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">Symbol:</td><td>
                                                                                <input  id="symbol"  type="text"  class="text"
value="EURUSD" />
                                                                                </td>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">Period:</td><td>
                                                                                <input  id="period"  type="text"  class="text"
value="M1" />
                                                                                </td>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">Брой неврони:</td><td>
                                                                                <input  id="numberOfNeurons"  type="text"  class="text"
value="20" />
                                                                                </td>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">Популация:</td><td>
                                                                                <input  id="populationSize"  type="text"  class="text"
value="40" />
                                                                                </td>
                                                                        </tr>
                                                                        <tr>
                                                                                <td align="right">Брой барове:</td><td>
                                                                                <input  id="numberOfBars"  type="text"  class="text"
value="5" />
                                                                                </td>
                                                                        </tr>
                                                                </table>
                                                        </div>
                                                        <div id="searchInpSub">
                                                                <input id="start" type="submit" class="submit" value="" />
                                                        </div>
                                                </form>
                                        </div>
                                        <div id="left_mid">
                                                <h3>Новини</h3>
                                                <h4>10 Фев 2011</h4>
                                                <p>
                                                        Випуск  2010/2011,  на  ТУЕС  към  ТУ-София,  предава  разработените  си  дипломни  работи,  по
предварително зададен график. <a href="#">подробности</a>
                                                </p>
                                                <h4>16 Фев 2011</h4>
                                                <p>
                                                        Процесът  за  предаване  на  дипломни  работи  протече  в  рамките  на  една  седмица,  като  срокът  за
разработка на някои бе удължен, по молба утвърдена от дипломните ръководители. <a href="#">подробности</a>
                                                </p>
                                                <h4>11 Мар 2011</h4>
                                                <p>
                                                        Дипломните  работи  окончателно  бяха  внесени  в  деловодството  на  училището,  за  назначаване  на
рецензенти и определяне на дати за защита. <a href="#">подробности</a>
                                                </p>
                                        </div>
                                        <div id="left_bot">
                                                <h3>Полезни връзки</h3>
                                                <ul>
                                                        <li>
                                                                &raquo; <a href="http://vitoshatrade.tbsoft.eu/">VitoshaTrade</a>
```

```
                                                                        </li>
                                                                        <li>
                                                                                &raquo; <a href="http://www.tradingsolutions.com/">TradingSolutions</a>
                                                                        </li>
                                                                        <li>
                                                                                &raquo; <a href="http://www.moneybee.net/">MoneyBee</a>
                                                                        </li>
                                                                        <li>
                                                                                &raquo; <a href="http://www.gstock.com/">GStock</a>
                                                                        </li>
                                                                        <li>
                                                                                &raquo; <a href="http://www.elsys-bg.org/">ТУЕС към ТУ-София</a>
                                                                        </li>
                                                                        <li>
                                                                                &raquo; <a href="http://www.tu-sofia.bg/">ТУ София</a>
                                                                        </li>
                                                                </ul>
                                                        </div>
                                                </div>

                                                <div id="right">
                                                        <div id="right_top">
                                                                <h3 class="firsth3">Уеб базирани системи за разпределени изчисления:</h3>
                                                                <p>
                                                                        Уеб базираните системи за разпределени изчисления използват възможностите за изпълнение на програмен код от страна на потребителските компютри, в рамките на уеб браузъра. За създаването на уеб базирани системи най-често се използват технологиите Java Applets, Adobe Flash, AJAX и други.
                                                                </p>
                                                                <p>
                                                                        В настоящата дипломна работа е представена уеб базирана система за разпределени изчисления, базирана на AJAX. Целта на системата е да обучава изкуствени невронни мрежи за прогнозиране на валутни курсове. Обучението се осъществява с популационен евристичен алгоритъм – диференциална еволюция.
                                                                </p>
                                                                <h3>Информация</h3>
                                                                <p>
                                                                        Разработката е част от проекта <a href="http://vitoshatrade.tbsoft.eu/" target="_blank">VitoshaTrade</a>.
                                                                </p>
                                                        </div>
                                                        <div id="right_mid">
                                                                <div id="right_mid_1"><img src="images/photo1.jpg" width="162" height="102" alt="" />
                                                                        <h4><a href="http://www.compsystech.org/">CompSysTech 2011</a></h4>
                                                                        <p>
                                                                                CompSysTech е международна конференция, организирана от българската академична област.
                                                                        </p>
                                                                </div>
                                                                <div id="right_mid_2"><img src="images/photo2.jpg" width="162" height="102" alt="" />
                                                                        <h4><a href="http://www.sigevo.org/gecco-2011/">GECCO 2011</a></h4>
                                                                        <p>
                                                                                GECCO е най-голямата и престижна конференция за еволюционни алгоритми.
                                                                        </p>
                                                                </div>
                                                                <div id="right_mid_3"><img src="images/photo3.jpg" width="162" height="102" alt="" />
                                                                        <h4><a href="http://parallel.bas.bg/Conferences/SciCom11.html">LSSC 2011</a></h4>
                                                                        <p>
                                                                                LSSC е българска конференция с международно участие, посветена на агоритмични проблеми трудни за пресмятане.
                                                                        </p>
                                                                </div>
                                                                <div id="right_mid_4">
                                                                         
                                                                </div>
                                                        </div>
                                                        <div id="right_bot">
                                                                <h3 class="firsth3">Инсталиране</h3>
                                                                <p>
                                                                        От страна на крайния потребител, не е нужна инсталация на допълнителни модули. Достатъчно е крайният потребител да има AJAX съвместим уеб браузър. За да се вгради в уеб страница изчислителният модул, то трябва да се добавят фрагменти код и да се подсигури връзка с отдалечен уеб сървър.
                                                                </p>
                                                                <h3>Юридически аспекти</h3>
                                                                <p>
                                                                        Най-популярните системи за разпределени изчисления разчитат на дарени изчислителни ресурси от потребители в глобалната мрежа. Този акт на дарение е обвързан с някои юридически особености. На първо място, лицето стартиращо модули за разпределени изчисления трябва да има административното право за извършването на подобно действие. На второ място, нужно е изричното съгласие на крайните потребители, ако тяхната изчислителна система ще се използва за допълнителни цели (в случая разпределени изчисления на фонов режим).
                                                                </p>
                                                        </div>
                                                </div>
                                        </div>
                                        <div id="bottom">
                                                <p>
                                                        Copyright &copy; <a href="mailto:d.chutrov@gmail.com">Даниел Чутров</a> / Дизайн: <a href="http://www.alphastudio.pl" target="_blank">Alpha Studio - Darmowe Szablony</a>
                                                </p>
                                        </div>
                                </div>
                        </body>
</html>
```

style.css

```
* {
        margin: 0;
        padding: 0;
}

body {
        padding: 30px 0;
        font: 13px Arial, Helvetica, sans-serif;
        text-align: center;
        color: #7C6031;
        background: #EEF6E9;
}

h1 {
        position: absolute;
        left: 15px;
        top: 15px;
        font: 30px Tahoma, Arial, Helvetica, sans-serif;
        text-transform: uppercase;
        color: White;
}

h2 {
        position: absolute;
        left: 15px;
        top: 157px;
        font: 18px Tahoma, Arial, Helvetica, sans-serif;
        color: White;
}

h3 {
        padding: 30px 0 5px 0;
```

```
            font: bold 16px Arial, Helvetica, sans-serif;
            letter-spacing: 1px;
}

h3.firsth3 {
            padding-top: 0;
}

p {
            padding: 5px 0;
            font: 13px/18px Arial, Helvetica, sans-serif;
            text-align: justify;
            color: #7C6031;
}

a {
            text-decoration: none;
            color: #3076E3;
}

a:hover {
            text-decoration: underline;
            color: #3076E3;
}

img {
            border: none;
}

#main {
            margin: 0 auto;
            padding: 10px 0 0 0;
            width: 820px;
            text-align: left;
            background: #B4DA9F;
}

#top {
            position: relative;
            margin: 0 10px;
            width: 800px;
            height: 253px;
            background: #7DB357 url(images/top.jpg) no-repeat top;
}

#logo {
            height: 179px;
}

#menu {
            margin: 0 10px;
            height: 74px;
}

#menu ul {
            list-style: none;
}

#menu ul li {
            float: left;
            display: block;
            padding: 0 25px;
}

#menu ul li a {
            font: bold 12px/74px Arial, Helvetica, sans-serif;
            text-transform: uppercase;
            text-decoration: none;
            color: #EEF6E9;
}

#menu ul li a:hover {
            text-decoration: none;
            color: White;
}

#middle {
            margin: 0 10px;
            padding: 10px 0 0 0;
}

#left {
            float: left;
            width: 270px;
            background: #D2E7C5;
}

#left h3 {
            padding: 0;
            font: bold 14px Arial, Helvetica, sans-serif;
            text-align: center;
            text-transform: uppercase;
            letter-spacing: 1px;
            color: #3FA14B;
}

#left h4 {
            padding: 18px 0 0 0;
            font: bold 12px Arial, Helvetica, sans-serif;
            color: #3FA14B;
}

#left p {
            padding: 5px 0;
            font: 12px/17px Arial, Helvetica, sans-serif;
            color: #3FA14B;
}

#search {
            margin: 10px;
            height: 220px;
            text-align: center;
            background: #7DB357;
}

#search p {
            padding: 12px 0 0 0;
            font: bold 14px/20px Arial, Helvetica, sans-serif;
            text-align: center;
            color: #EEF6E9;
}

#formSearch {
            margin: 0 auto;
            width: 200px;
```

```
}

#searchInpTxt {
            float: left;
            width: 190px;
            height: 170px;
            background: #D2E7C5;
            border: 1px solid #63A13E;
}

#search input.text {
            display: block;
            margin: 4px 0 0 2px;
            width: 60px;
            font: 14px Arial, Helvetica, sans-serif;
            color: #4E7C31;
            background: none;
            border: 1px solid #63A13E;
}

#searchInpSub {
            float: left;
            width: 26px;
            height: 24px;
            background: url(images/go.gif);
            border: 1px solid #63A13E;
}

#search input.submit {
            display: block;
            float: right;
            width: 26px;
            height: 24px;
            background: none;
            border: none;
}

#left_mid {
            padding: 15px;
}

#left_bot {
            margin: 10px;
            padding: 15px;
            color: #D2E7C5;
            background: #7DB357;
}

#left_bot h3 {
            color: #EEF6E9;
}

#left ul {
            list-style: none;
            padding: 10px 0 0 0;
}

#left ul li {
            padding: 0 0 0 16px;
            text-align: left;
}

#left ul li a {
            font: 12px/20px Arial, Helvetica, sans-serif;
            text-decoration: none;
            color: #D2E7C5;
}

#left ul li a:hover {
            text-decoration: underline;
            color: #EEF6E9;
}

#right {
            float: right;
            width: 520px;
}

#right_top {
            margin: 0 0 10px 0;
            padding: 15px;
            background: #D2E7C5;
}

#right_mid {
            margin: 0 0 10px 0;
            padding: 0 8px;
            background: #63A13E;
}

#right_mid h4 {
            padding: 5px 9px 7px 9px;
            font-size: 12px;
            text-align: left;
}

#right_mid p {
            padding: 0 9px;
            font: 11px/15px Arial, Helvetica, sans-serif;
            text-align: left;
            color: #D2E7C5;
}

#right_mid a {
            text-transform: uppercase;
            text-decoration: none;
            color: #A0B7D9;
}

#right_mid a:hover {
            text-decoration: underline;
            color: #A0B7D9;
}

#right_mid_1, #right_mid_2 {
            float: left;
            width: 168px;
            padding: 10px 0;
            text-align: center;
}

#right_mid_3 {
            float: right;
            width: 168px;
            padding: 10px 0;
```

```
            text-align: center;
}

#right_mid_4 {
            clear: both;
            height: 5px;
            overflow: hidden;
}

#right_bot {
            padding: 15px;
            background: #D2E7C5;
}

#bottom {
            clear: both;
            padding: 0 0 0 25px;
}

#bottom p {
            padding: 0;
            font: 11px/40px Arial, Helvetica, sans-serif;
            color: #3FA14B;
}

#bottom a, #bottom a:hover {
            color: #3FA14B;
}
```

build.bat

```
@echo off

rem ############################################################################
rem #                                                                          #
rem # VitoshaTrade is Distributed Artificial Neural Network trained by         #
rem # Differential Evolution for prediction of Forex. Project development is in #
rem # Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, #
rem # the capital of Bulgaria.                                                 #
rem #                                                                          #
rem # Copyright (C) 2008-2010 by Todor Balabanov  ( tdb@tbsoft.eu )            #
rem #                            Iliyan Zankinski ( iliyan_mf@abv.bg )         #
rem #                            Galq Cirkalova   ( galq_cirkalova@abv.bg )    #
rem #                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) #
rem #                            Momchil Anachkov ( mZer0000@gmail.com )       #
rem #                                                                          #
rem # This program is free software: you can redistribute it and/or modify     #
rem # it under the terms of the GNU General Public License as published by     #
rem # the Free Software Foundation, either version 3 of the License, or        #
rem # (at your option) any later version.                                      #
rem #                                                                          #
rem # This program is distributed in the hope that it will be useful,          #
rem # but WITHOUT ANY WARRANTY; without even the implied warranty of           #
rem # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the            #
rem # GNU General Public License for more details.                             #
rem #                                                                          #
rem # You should have received a copy of the GNU General Public License        #
rem # along with this program. If not, see <http://www.gnu.org/licenses/>.     #
rem #                                                                          #
rem ############################################################################

rem ############################################################################
rem # Check input parameters.                                                  #
rem ############################################################################
if "%~1"=="" goto end

rem ############################################################################
rem # Create web back-end directory.                                           #
rem ############################################################################
md "%~dp1backend\"

rem ############################################################################
rem # Compilation.                                                             #
rem ############################################################################
javac.exe -d ./binaries ./source/*.java -classpath ./binaries

rem ############################################################################
rem # Archiving.                                                               #
rem ############################################################################
cd binaries
jar.exe cf VitoshaTrade.jar *.class
cd ..

rem ############################################################################
rem # Deployment.                                                              #
rem ############################################################################
rem copy .\binaries\*.class "%~dp1backend\"
copy .\binaries\*.jar "%~dp1backend\"
copy .\source\*.html "%~dp1backend\"

rem ############################################################################
rem # End of the srcipt if there are no input parameters.                      #
rem ############################################################################
:end

rem ############################################################################
rem # Stop screen.                                                             #
rem ############################################################################
pause

@echo on
```

index.html

```
<!--
 ******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
```

```
 * Copyright (C) 2008-2010 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )         *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************
-->

<html>
            <head>
                        <title>.:: VitoshaTrade - backend ::.</title>
                        <style>
                                    body {text-align: center}
                        </style>
            </head>

<body>
<h1> Vitosha Trade Applet </h1>
<applet width="800" height="600" codebase="." code="eu.veldsoft.backend.VitoshaTradeApplet.class" archive="VitoshaTrade.jar"></applet>
</body>
</html>
```

ArtificialNeuralNetwork.java

```
package eu.veldsoft.backend;

/******************************************************************************
 *                                                                            *
 * VitoshaTrade is Distributed Artificial Neural Network trained by           *
 * Differential Evolution for prediction of Forex. Project development is in  *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,  *
 * the capital of Bulgaria.                                                   *
 *                                                                            *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )              *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )           *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )      *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )         *
 *                            Ralitza Koleva   ( rallly@abv.bg )              *
 *                                                                            *
 * This program is free software: you can redistribute it and/or modify       *
 * it under the terms of the GNU General Public License as published by       *
 * the Free Software Foundation, either version 3 of the License, or          *
 * (at your option) any later version.                                        *
 *                                                                            *
 * This program is distributed in the hope that it will be useful,            *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of             *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the              *
 * GNU General Public License for more details.                               *
 *                                                                            *
 * You should have received a copy of the GNU General Public License          *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.       *
 *                                                                            *
 ******************************************************************************/

/**
 * Data structure containing ANN for visual representation.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 24 Dec 2010
 */
public class ArtificialNeuralNetwork {

            /**
             * Regular neuron flag.
             */
            static final int REGULAR_NEURON = 0x00;

            /**
             * Bias neuron flag.
             */
            static final int BIAS_NEURON = 0x01;

            /**
             * Input neuron flag.
             */
            static final int INPUT_NEURON = 0x02;

            /**
             * Output neuron flag.
             */
            static final int OUTPUT_NEURON = 0x04;

            /**
             * Representing the ID of the ANN.
             */
            int id;

            /**
             * Containing the currency pair.
             */
            String symbol;

            /**
             * Representing the time period for the bars.
             */
            int period;

            /**
             * Representing the number of neurons for the selected ANN.
             */
            int numberOfNeurons;
```

```
	/**
	 * Containing the flags for the selected neuron.
	 */
	int flags[];

	/**
	 * Representing weights fitness value.
	 */
	double fitness;

	/**
	 * Containing the weight variables for the links.
	 */
	double weights[][];

	/**
	 * Containing the activity variables for the links.
	 */
	double activities[][];

	/**
	 * Neurons out signals.
	 */
	double signals[];

	/**
	 * Neurons out errors.
	 */
	double errors[];

	/**
	 * Containing the coordinates of the neurons for the selected ANN.
	 */
	int coordinates[][];

	/**
	 * Size of input layer.
	 *
	 * @return The size of the input layer.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 31 Aug 2015
	 */
	int numberOfInputNeurons() {
		int size = 0;

		for (int k = 0; k < flags.length; k++) {
			if (flags[k] != INPUT_NEURON) {
				continue;
			}

			size++;
		}

		return size;
	}

	/**
	 * Size of hidden layer.
	 *
	 * @return The size of the hidden layer.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 31 Aug 2015
	 */
	int numberOfHiddenNeurons() {
		int size = 0;

		for (int k = 0; k < flags.length; k++) {
			if (flags[k] != REGULAR_NEURON) {
				continue;
			}

			size++;
		}

		return size;
	}

	/**
	 * Size of output layer.
	 *
	 * @return The size of the output layer.
	 *
	 * @author Todor Balabanov
	 *
	 * @email todor.balabanov@gmail.com
	 *
	 * @date 31 Aug 2015
	 */
	int numberOfOutputNeurons() {
		int size = 0;

		for (int k = 0; k < flags.length; k++) {
			if (flags[k] != OUTPUT_NEURON) {
				continue;
			}

			size++;
		}

		return size;
	}
}
```

ConnectionPane.java

```
package eu.veldsoft.backend;

/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
```

```
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                         Iliyan Zankinski ( iliyan_mf@abv.bg )               *
 *                         Galq Cirkalova   ( galq_cirkalova@abv.bg )          *
 *                         Ivan Grozev      ( ivan.iliev.grozev@gmail.com )    *
 *                         Momchil Anachkov ( mZer0000@gmail.com )             *
 *                         Ralitza Koleva   ( rallly@abv.bg )                  *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * Panel with GUI controls for Connection Management.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Nov 2010
 */
class ConnectionPane extends JPanel {
	/**
	 * Default serial version UID.
	 */
	private static final long serialVersionUID = 1L;

	/**
	 * Parent applet reference.
	 */
	private VitoshaTradeApplet parent = null;

	/**
	 * GUI control for obtaining the source neuron for a specific connection.
	 */
	private JTextField neuronSource = new JTextField();

	/**
	 * GUI control for obtaining the neuron-recipient for a specific connection.
	 */
	private JTextField neuronDestination = new JTextField();

	/**
	 * GUI control for obtaining and setting the activity for a specific
	 * connection.
	 */
	private JTextField activity = new JTextField();

	/**
	 * GUI control for obtaining the weight for a specific connection.
	 */
	private JTextField weight = new JTextField();

	/**
	 * Constructing connection pane.
	 *
	 * @param parent
	 *            The parent class.
	 *
	 * @author Momchil Anachkov
	 *
	 * @email mZer0000@gmail.com
	 *
	 * @date 18 Nov 2010
	 */
	public ConnectionPane(final VitoshaTradeApplet parent) {
		this.parent = parent;
		this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
				VitoshaTradeApplet.EAST_PANE_HEIGHT));

		setLayout(new GridLayout(25, 1));

		add(new JLabel(Texts.LABEL_DESTINATION));
		add(neuronSource);
		neuronSource.setEditable(false);

		add(new JLabel(Texts.LABEL_SOURCE));
		add(neuronDestination);
		neuronDestination.setEditable(false);

		add(new JLabel(Texts.LABEL_WEIGHT));
		add(weight);

		add(new JLabel(Texts.LABEL_ACTIVITY));
		add(activity);

		weight.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent ) {
				parent.ann.weights[Integer.parseInt(neuronSource.getText())][Integer
						.parseInt(neuronDestination.getText())] = Double
						.parseDouble(weight.getText());
				parent.workArea.repaint();
			}
		});

		activity.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent ) {
				parent.ann.activities[Integer.parseInt(neuronSource.getText())][Integer
						.parseInt(neuronDestination.getText())] = Double
						.parseDouble(activity.getText());
				parent.workArea.repaint();
			}
```

```
                });
        }

        /**
         * Loading connection properties.
         *
         * @param sourceIndex
         *            Source neuron of selected connection.
         *
         * @param destinationIndex
         *            Destination neuron of selected connection.
         *
         * @param connectionActivity
         *            Activity of selected connection.
         *
         * @param connectionWeight
         *            Weight of selected connection.
         *
         * @author Momchil Anachkov
         *
         * @email mZer0000@gmail.com
         *
         * @date 01 Feb 2010
         */
        void setValues(int sourceIndex, int destinationIndex,
                        double connectionActivity, double connectionWeight) {
                neuronSource.setText("" + sourceIndex);
                neuronDestination.setText("" + destinationIndex);
                activity.setText("" + connectionActivity);
                weight.setText("" + connectionWeight);
        }
}
```

DatabaseHelper.java

```
package eu.veldsoft.backend;

/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )          *
 *                            Ralitza Koleva   ( rallly@abv.bg )               *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

/**
 * Database helping class.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 19 Dec 2010
 */
class DatabaseHelper {

        /**
         * SQL query for selecting all ANN ids, symbols and periods.
         */
        private static final String SQL_QUERY_SYMBOL_PERIOD_ID = "SELECT currency_pairs.symbol, time_periods.period, ann.id FROM ann, ann_kind, time_periods, currency_pairs WHERE currency_pairs.id=ann_kind.currency_pairs_id AND time_periods.id=currency_pairs.period_id AND ann.ann_kind_id=ann_kind.id ORDER BY ann.id;";

        /**
         * SQL query for selecting all ANN ids, symbols and periods by symbol.
         */
        private static final String SQL_QUERY_SYMBOL_PERIOD_ID_BY_SYMBOL = "SELECT currency_pairs.symbol, time_periods.period, ann.id FROM ann, ann_kind, time_periods, currency_pairs WHERE ann.ann_kind_id=ann_kind.id AND ann_kind.currency_pairs_id=currency_pairs.id AND time_periods.id=currency_pairs.period_id AND currency_pairs.symbol='%s' ORDER BY ann.id;";

        /**
         * SQL query for selecting all ANN ids, symbols and periods by period.
         */
        private static final String SQL_QUERY_SYMBOL_PERIOD_ID_BY_PERIOD = "SELECT currency_pairs.symbol, time_periods.period, ann.id FROM ann, ann_kind, time_periods, currency_pairs WHERE ann_kind.currency_pairs_id=currency_pairs.id AND time_periods.id=currency_pairs.period_id AND time_periods.id=%d AND ann.ann_kind_id=ann_kind.id ORDER BY ann.id;";

        /**
         * SQL query for selecting all ANN ids, symbols and periods by symbol and
         * period.
         */
        private static final String SQL_QUERY_SYMBOL_PERIOD_ID_BY_SYMBOL_AND_PERIOD = "SELECT currency_pairs.symbol, time_periods.period, ann.id FROM ann, ann_kind, time_periods, currency_pairs WHERE currency_pairs.symbol='%s' AND period_id=%d AND ann_kind.currency_pairs_id=currency_pairs.id AND time_periods.id=currency_pairs.period_id AND ann.ann_kind_id=ann_kind.id ORDER BY ann.id;";

        /**
         * SQL query for selecting information about all ANN kinds ids symbol,
         * period.
         */
        private static final String SQL_QUERY_ANN_KINDS_ID = "SELECT ann_kind.id AS id, currency_pairs.symbol AS symbol, time_periods.id AS period FROM ann_kind, time_periods, currency_pairs WHERE ann_kind.currency_pairs_id=currency_pairs.id AND time_periods.id=currency_pairs.period_id ";

        /**
         * SQL query for selecting ANN values by ANN id.
         */
        private static final String SQL_QUERY_ANN_BY_ID = "SELECT ann.id AS id, currency_pairs.symbol AS symbol, time_periods.id AS period_id, ann_kind.number_of_neurons, ann_kind.flags, ann.fitness, ann.weights, ann_kind.activities, neurons.coordinates coordinates FROM ann, ann_kind, neurons coordinates, currency_pairs, time_periods WHERE ann.id=%d AND neurons.coordinates.ann_id=ann.id AND ann_kind.currency_pairs_id=currency_pairs.id AND currency_pairs.period_id=time_periods.id AND ann.ann_kind_id = ann_kind.id;";
```

```
	/**
	 * SQL query for selecting currency pairs ID by simbol and period.
	 */
	private static final String SQL_QUERY_CURRENCY_PAIR_BY_CURRENCY_PAIR_AND_PERIOD = "SELECT id FROM currency_pairs WHERE symbol='%s' AND period_id=%d";

	/**
	 * SQL query for checking if selected ANN has neuron coordinates.
	 */
	private static final String SQL_QUERY_HAS_COORDINATES = "SELECT * FROM neurons_coordinates WHERE neurons_coordinates.ann_id=%d;";

	/**
	 * SQL query for obtaining number of neurons.
	 */
	private static final String SQL_QUERY_NUMBER_OF_NEURONS = "SELECT ann_kind.number_of_neurons FROM ann, ann_kind WHERE ann.ann_kind_id = ann_kind.id AND ann.id =
%d;";

	/**
	 * SQL query for insertion of default initial coordinates.
	 */
	private static final String SQL_QUERY_INSERT_DEFAULT_COORDINATES = "INSERT INTO neurons_coordinates (ann_id, coordinates) VALUES (%d, '%s');";

	/**
	 * SQL query updating neurons flags and weights activities.
	 */
	private static final String SQL_QUERY_UPDATE_FLAGS_AND_ACTIVITIES = "UPDATE ann_kind SET flags = '%s', activities = '%s' WHERE ann_kind.id = (SELECT ann_kind_id
FROM ann WHERE ann.id = %d);";

	/**
	 * SQL query updating weights.
	 */
	private static final String SQL_QUERY_UPDATE_WEIGHTS = "UPDATE ann SET weights = '%s' WHERE ann.id = %d;";

	/**
	 * SQL query for updating neurons coordinates.
	 */
	private static final String SQL_QUERY_UPDATE_COORDINATES = "UPDATE neurons_coordinates SET coordinates = '%s' WHERE neurons_coordinates.ann_id = '%d';";

	/**
	 * SQL query for inserting information about a new ANN kind.
	 */
	private static final String SQL_QUERY_INSERT_ANN_KIND = "INSERT INTO ann_kind(currency_pairs_id, number_of_neurons, flags, activities) VALUES ('%s', '%d', '%s',
'%s')";

	/**
	 * SQL query for inserting information about a new ANN.
	 */
	private static final String SQL_QUERY_INSERT_ANN = "INSERT INTO ann(ann_kind_id, fitness, weights) VALUES('%d', '%d', '%s')";

	/**
	 * SQL query for deleting information about an ANN kind.
	 */
	private static final String SQL_QUERY_DELETE_ANN_KIND = "DELETE FROM ann_kind WHERE id IN (SELECT ann_kind_id FROM ann WHERE id='%d')";

	/**
	 * SQL query for deleting information about an ANN.
	 */
	private static final String SQL_QUERY_DELETE_ANN = "DELETE FROM ann WHERE id='%d'";

	/**
	 * SQL query for inserting information about a new ANN.
	 */
	private static final String SQL_QUERY_INSERT_ANN_COORDINATES = "INSERT INTO neurons_coordinates(ann_id, coordinates) VALUES('%d', '%s')";

	/**
	 * SQL query for selecting currency pairs.
	 */
	private static final String SQL_QUERY_SYMBOL = "SELECT DISTINCT symbol FROM currency_pairs";

	/**
	 * SQL query for selecting periods.
	 */
	private static final String SQL_QUERY_PERIOD = "SELECT id, period FROM time_periods";

	/**
	 * Containing the MYSQL driver for the connection.
	 */
	private static final String MYSQL_DRIVER = "com.mysql.jdbc.Driver";

	/**
	 * Default minimum x coordinate to be written into database.
	 */
	private static final int DEFAULT_MIN_X_COORDINATE = 100;

	/**
	 * Default minimum y coordinate to be written into database.
	 */
	private static final int DEFAULT_MIN_Y_COORDINATE = 100;

	/**
	 * Default maximum x coordinate to be written into database.
	 */
	private static final int DEFAULT_MAX_X_COORDINATE = 500;

	/**
	 * Default maximum y coordinate to be written into database.
	 */
	private static final int DEFAULT_MAX_Y_COORDINATE = 500;

	/**
	 * Containing the address where the database is hosted.
	 */
	private String host;

	/**
	 * Representing the port number for the host.
	 */
	private int port;

	/**
	 * Containing the name of the database.
	 */
	private String databaseName;

	/**
	 * Containing the username for logging in.
	 */
	private String username;

	/**
	 * Containing the password for logging in.
	 */
	private String password;

	/**
	 * Link to the database server.
	 */
```

```
private Connection connection;

/**
 * Constructing the database helping class with passed parameters.
 *
 * @param host
 *            The address where the database is hosted.
 *
 * @param port
 *            The port number for the host.
 *
 * @param databaseName
 *            The name of the database.
 *
 * @param username
 *            The username for logging into the server.
 *
 * @param password
 *            The password for logging into the server.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public DatabaseHelper(String host, int port, String databaseName,
                      String username, String password) {
            super();

            this.host = host;
            this.port = port;
            this.databaseName = databaseName;
            this.username = username;
            this.password = password;
}

/**
 * Constructing the database helping class without passed parameters.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public DatabaseHelper() {
            this("", 0, "", "", "");
}

/**
 * Database host URL getter.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public String getHost() {
            return host;
}

/**
 * Database host URL setter.
 *
 * @param host
 *            The host to be set.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void setHost(String host) {
            this.host = host;
}

/**
 * Databes server port getter.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public int getPort() {
            return port;
}

/**
 * Databes server port setter.
 *
 * @param port
 *            The port to be set.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void setPort(int port) {
            this.port = port;
}

/**
 * Database name getter.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public String getDatabaseName() {
            return databaseName;
}

/**
 * Database name setter.
 *
 * @param databaseName
```

```
 *            The database name to be set.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void setDatabaseName(String databaseName) {
            this.databaseName = databaseName;
}

/**
 * Database username getter.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public String getUsername() {
            return username;
}

/**
 * Database username setter.
 *
 * @param username
 *            The username to be set.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void setUsername(String username) {
            this.username = username;
}

/**
 * Database password getter.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public String getPassword() {
            return password;
}

/**
 * Database password setter.
 *
 * @param password
 *            The password to be set.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void setPassword(String password) {
            this.password = password;
}

/**
 * Connecting to database.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void connect() throws Exception {
            Class.forName(MYSQL_DRIVER).newInstance();
            connection = DriverManager.getConnection("jdbc:mysql://" + host + ":"
                                    + port + "/" + databaseName, username, password);
}

/**
 * Disconnecting from database.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void disconnect() throws Exception {
            connection.close();
}

/**
 * Provide list with available networks.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 *
 * @return 2D array of strings with ANN information (id, symbol, period).
 */
public String[][] loadAnnList() {
            String[][] result = null;
            try {
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select
                                                .executeQuery(SQL_QUERY_SYMBOL_PERIOD_ID);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                    result = new String[size][3];
                                    int i = 0;
                                    while (resultSet.next() == true) {
                                                result[i][0] = resultSet.getString("symbol");
                                                result[i][1] = resultSet.getString("period");
```

```
                                        result[i][2] = Integer.toString(resultSet.getInt("id"));
                                        i++;
                                }
                        }
                } catch (Exception ex) {
                        result = null;
                        System.err.println(Texts.ERROR_LOAD_ANN_LIST);
                        ex.printStackTrace();
                        // TODO Inform user that there is no valid ANN list.
                }

                return (result);
        }

        /**
         * Provide list with available networks.
         *
         * @param symbol
         *            Currency pair symbol.
         *
         * @author Momchil Anachkov
         *
         * @email mZer0000@gmail.com
         *
         * @date 18 Dec 2010
         *
         * @return 2D array of strings with ANN information (id, symbol, period).
         */
        public String[][] loadAnnListBySymbol(String symbol) {
                String[][] result = null;

                try {
                        String sql = String.format(SQL_QUERY_SYMBOL_PERIOD_ID_BY_SYMBOL,
                                        symbol);
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select.executeQuery(sql);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                result = new String[size][3];
                                int i = 0;
                                while (resultSet.next() == true) {
                                        result[i][0] = resultSet.getString("symbol");
                                        result[i][1] = resultSet.getString("period");
                                        result[i][2] = Integer.toString(resultSet.getInt("id"));
                                        i++;
                                }
                        }
                } catch (Exception e) {
                        System.err.println(Texts.ERROR_LOAD_ANN_LIST);
                        result = null;
                        // TODO Inform user that there is no valid ANN list.
                }

                return (result);
        }

        /**
         * Provide list with available networks.
         *
         * @param period
         *            Currency pair period.
         *
         * @author Momchil Anachkov
         *
         * @email mZer0000@gmail.com
         *
         * @date 18 Dec 2010
         *
         * @return 2D array of strings with ANN information (id, symbol, period).
         */
        public String[][] loadAnnListByPeriod(int period) {
                String[][] result = null;

                try {
                        String sql = String.format(SQL_QUERY_SYMBOL_PERIOD_ID_BY_PERIOD,
                                        period);
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select.executeQuery(sql);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                result = new String[size][3];
                                int i = 0;
                                while (resultSet.next() == true) {
                                        result[i][0] = resultSet.getString("symbol");
                                        result[i][1] = resultSet.getString("period");
                                        result[i][2] = Integer.toString(resultSet.getInt("id"));
                                        i++;
                                }
                        }
                } catch (Exception ex) {
                        System.err.println(Texts.ERROR_LOAD_ANN_LIST);
                        result = null;
                        // TODO Inform user that there is no valid ANN list.
                }

                return (result);
        }

        /**
         * Provide list with available networks.
         *
         * @param symbol
         *            Currency pair symbol.
         *
         * @param period
         *            Currency pair period.
         *
         * @author Momchil Anachkov
         *
         * @email mZer0000@gmail.com
         *
         * @date 18 Dec 2010
         *
         * @return 2D array of strings with ANN information (id, symbol, period).
         */
        public String[][] loadAnnList(String symbol, int period) {
                String[][] result = null;

                try {
```

```
                        String sql = String.format(
                                                SQL_QUERY_SYMBOL_PERIOD_ID_BY_SYMBOL_AND_PERIOD, symbol,
                                                period);
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select.executeQuery(sql);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                    result = new String[size][3];
                                    int i = 0;
                                    while (resultSet.next() == true) {
                                                result[i][0] = resultSet.getString("symbol");
                                                result[i][1] = resultSet.getString("period");
                                                result[i][2] = Integer.toString(resultSet.getInt("id"));
                                                i++;
                                    }
                        }
            } catch (Exception e) {
                        System.err.println(Texts.ERROR_LOAD_ANN_LIST);
                        e.printStackTrace();
                        result = null;
                        // TODO Inform user that there is no valid ANN list.
            }

            return (result);
}

/**
 * Provides a list with available currency pairs.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 17 Oct 2011
 *
 * @return Array of strings with currency pairs.
 */
public String[][] loadCurrencyPairs() {
            String[][] result = null;
            try {
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select.executeQuery(SQL_QUERY_SYMBOL);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                    result = new String[size][2];
                                    int i = 0;
                                    while (resultSet.next() == true) {
                                                result[i][0] = resultSet.getString("symbol");
                                                i++;
                                    }
                        }
            } catch (Exception ex) {
                        result = null;
                        System.err.println(Texts.ERROR_LOAD_CURRENCY_PAIRS_LIST);
                        ex.printStackTrace();
                        // TODO Inform user that there is no valid currency pairs list.
            }
            return (result);
}

/**
 * Provides a list with available periods.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 17 Oct 2011
 *
 * @return 2D array of strings with periods.
 */
public String[][] loadPeriods() {
            String[][] result = null;
            try {
                        Statement select = connection.createStatement();
                        ResultSet resultSet = select.executeQuery(SQL_QUERY_PERIOD);

                        resultSet.last();
                        int size = resultSet.getRow();
                        resultSet.beforeFirst();
                        if (size > 0) {
                                    result = new String[size][2];
                                    int i = 0;
                                    while (resultSet.next() == true) {
                                                result[i][0] = resultSet.getString("id");
                                                result[i][1] = resultSet.getString("period");
                                                i++;
                                    }
                        }
            } catch (Exception ex) {
                        result = null;
                        System.err.println(Texts.ERROR_LOAD_PERIODS_LIST);
                        ex.printStackTrace();
                        // TODO Inform user that there is no valid periods list.
            }
            return (result);
}

/**
 * Checking if selected ANN has neuron coordinates.
 *
 * @param id
 *            Unique identifier for the ANN.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 03 Jan 2010
 */
void initIfNeeded(int id) {
            String sql = "";
            Statement statement = null;
            ResultSet resultSet = null;

            try {
                        statement = connection.createStatement();

                        sql = String.format(SQL_QUERY_HAS_COORDINATES, id);
```

```
                        resultSet = statement.executeQuery(sql);

                        resultSet.last();
                        if (resultSet.getRow() == 1) {
                                return;
                        }
                        sql = String.format(SQL_QUERY_NUMBER_OF_NEURONS, id);
                        resultSet = statement.executeQuery(sql);
                        resultSet.first();
                        int numberOfNeurons = resultSet.getInt("number_of_neurons");
                        String values = "";
                        for (int i = 0; i < numberOfNeurons; i++) {
                                values += DEFAULT_MIN_X_COORDINATE
                                                + (int) (Math.random() * (DEFAULT_MAX_X_COORDINATE
                                                                - DEFAULT_MIN_X_COORDINATE + 1));
                                values += " ";
                                values += DEFAULT_MIN_Y_COORDINATE
                                                + (int) (Math.random() * (DEFAULT_MAX_Y_COORDINATE
                                                                - DEFAULT_MIN_Y_COORDINATE + 1));
                                values += " ";
                        }
                        values = values.trim();

                        sql = String.format(SQL_QUERY_INSERT_DEFAULT_COORDINATES, id,
                                        values);

                        statement.executeUpdate(sql);
                } catch (Exception ex) {
                        // TODO Inform user that there is database problem.
                }
        }

        /**
         * Provide selected ANN.
         *
         * @param id
         *            Unique identifier for the ANN.
         *
         * @author Momchil Anachkov
         *
         * @email mZer0000@gmail.com
         *
         * @date 18 Dec 2010
         */
        public ArtificialNeuralNetwork loadAnn(int id) {
                String line = "";
                String values[] = null;
                ArtificialNeuralNetwork result = null;

                initIfNeeded(id);

                try {
                        Statement select = connection.createStatement();
                        String sql = String.format(SQL_QUERY_ANN_BY_ID, id);
                        ResultSet resultSet = select.executeQuery(sql);

                        resultSet.last();
                        if (resultSet.getRow() != 1) {
                                throw (new Exception(Texts.ERROR_NUMBER_DATABASE_RESULTS));
                        }

                        result = new ArtificialNeuralNetwork();
                        result.id = resultSet.getInt("id");
                        result.symbol = resultSet.getString("symbol");
                        result.period = resultSet.getInt("period_id");
                        result.numberOfNeurons = resultSet.getInt("number_of_neurons");
                        result.fitness = resultSet.getDouble("fitness");

                        /*
                         * Passing values from database to array of flags.
                         */
                        line = resultSet.getString("flags");
                        values = line.split("\\s+");
                        result.flags = new int[result.numberOfNeurons];
                        if (values.length != result.numberOfNeurons) {
                                throw (new Exception("Wrong number of neuron flags."));
                        }
                        for (int i = 0; i < result.flags.length; i++) {
                                result.flags[i] = Integer.parseInt(values[i]);
                        }

                        /*
                         * Passing values from database to array of weights.
                         */
                        line = resultSet.getString("weights");
                        values = line.split("\\s+");
                        result.weights = new double[result.numberOfNeurons][result.numberOfNeurons];
                        if (values.length != result.numberOfNeurons
                                        * result.numberOfNeurons) {
                                throw (new Exception("Wrong number of weights."));
                        }
                        for (int j = 0; j < result.numberOfNeurons; j++) {
                                for (int i = 0; i < result.numberOfNeurons; i++) {
                                        result.weights[i][j] = Double.parseDouble(values[i + j
                                                        * result.numberOfNeurons]);
                                }
                        }

                        /*
                         * Passing values from database to array of activities.
                         */
                        line = resultSet.getString("activities");
                        values = line.split("\\s+");
                        result.activities = new double[result.numberOfNeurons][result.numberOfNeurons];
                        if (values.length != result.numberOfNeurons
                                        * result.numberOfNeurons) {
                                throw (new Exception("Wrong number of activities."));
                        }
                        for (int j = 0; j < result.numberOfNeurons; j++) {
                                for (int i = 0; i < result.numberOfNeurons; i++) {
                                        result.activities[i][j] = Double.parseDouble(values[i + j
                                                        * result.numberOfNeurons]);
                                }
                        }

                        /*
                         * Passing values from database to array of coordinates.
                         */
                        line = resultSet.getString("coordinates");
                        values = line.split("\\s+");
                        result.coordinates = new int[result.numberOfNeurons][2];
                        if (values.length != result.numberOfNeurons * 2) {
                                throw (new Exception("Wrong number of coordinates."));
                        }
                        for (int i = 0; i < result.numberOfNeurons; i++) {
```

```
                                    for (int k = 0; k < 2; k++) {
                                                result.coordinates[i][k] = Integer.parseInt(values[i * 2
                                                                                    + k]);
                                    }
                        }

            } catch (Exception ex) {
                        result = null;
                        System.err.println(Texts.ERROR_LOAD_ANN_PART_1 + id
                                                + Texts.ERROR_LOAD_ANN_PART_2);
                        ex.printStackTrace();
                        // TODO Inform user that there is no valid ANN list.
            }

            return (result);
}

/**
 * Save ANN to remote database server.
 *
 * @param ann
 *            ANN container.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Dec 2010
 */
public void saveAnn(ArtificialNeuralNetwork ann) {
            String sql = "";

            String flags = "";
            for (int i = 0; i < ann.flags.length; i++) {
                        flags = flags + Integer.toString(ann.flags[i]) + " ";
            }
            flags = flags.trim();

            String activities = "";
            for (int j = 0; j < ann.numberOfNeurons; j++) {
                        for (int i = 0; i < ann.numberOfNeurons; i++) {
                                    activities = activities + Double.toString(ann.activities[i][j])
                                                            + " ";
                        }
                        activities = activities.trim();
                        activities += "\n";
            }
            activities = activities.trim();

            String weights = "";
            for (int j = 0; j < ann.numberOfNeurons; j++) {
                        for (int i = 0; i < ann.numberOfNeurons; i++) {
                                    weights = weights + Double.toString(ann.weights[i][j]) + " ";
                        }
                        weights = weights.trim();
                        weights += "\n";
            }
            weights = weights.trim();

            String coordinates = "";
            for (int i = 0; i < ann.numberOfNeurons; i++) {
                        for (int k = 0; k < 2; k++) {
                                    coordinates = coordinates
                                                            + Integer.toString(ann.coordinates[i][k]) + " ";
                        }
            }
            coordinates = coordinates.trim();

            try {
                        Statement update = connection.createStatement();
                        sql = String.format(SQL_QUERY_UPDATE_FLAGS_AND_ACTIVITIES, flags,
                                                activities, ann.id);
                        update.executeUpdate(sql);

                        sql = String.format(SQL_QUERY_UPDATE_WEIGHTS, weights, ann.id);
                        update.executeUpdate(sql);

                        sql = String.format(SQL_QUERY_UPDATE_COORDINATES, coordinates,
                                                ann.id);
                        update.executeUpdate(sql);
            } catch (Exception ex) {
                        System.err.println(Texts.ERROR_UPDATE_ANN);
                        // TODO Inform user that the update process failed.
            }
}

/**
 * Saves a new ANN to the remote database server.
 *
 * @param annSymbol
 *            A currency pair chosen for the new ANN.
 *
 * @param annPeriod
 *            A time period chosen for the new ANN.
 *
 * @param annNumberNeurons
 *            Number of neurons of the new ANN.
 *
 * @param annActivitiesWeights
 *            Weights of the new ANN.
 *
 * @param annFitness
 *            Fitness of the new ANN.
 *
 * @param annFlags
 *            Flags (neuron types) of the new ANN's neurons.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 20 Sep 2011
 */
public int saveNewAnn(String annSymbol, int annPeriod,
                        int annNumberNeurons, String annActivitiesWeights, int annFitness,
                        String annFlags) {
            String sql = "";

            /*
             * The ID of the currency pair of the new inserted ANN.
             */
            int currencyPairId = 0;

            /*
             * The ID of the new inserted ANN kind.
```

```
                 */
                int annKindId = 0;

                /*
                 * The ID of the new inserted ANN.
                 */
                int annId = 0;

                try {

                        /*
                         * Selects the currency pair ID by symbol and period.
                         */
                        Statement insert = connection.createStatement();
                        sql = String.format(
                                        SQL_QUERY_CURRENCY_PAIR_BY_CURRENCY_PAIR_AND_PERIOD,
                                        annSymbol, annPeriod);
                        insert.execute(sql);
                        ResultSet selectedCurrencyPairId = insert.getResultSet();
                        if (selectedCurrencyPairId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN kind.
                                 */
                                currencyPairId = selectedCurrencyPairId.getInt(1);
                        }

                        /*
                         * Inserts information about the new ANN kind.
                         */
                        sql = String.format(SQL_QUERY_INSERT_ANN_KIND, currencyPairId,
                                        annNumberNeurons, annFlags, annActivitiesWeights);
                        insert.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
                        ResultSet generatedAnnKindId = insert.getGeneratedKeys();
                        if (generatedAnnKindId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN kind.
                                 */
                                annKindId = generatedAnnKindId.getInt(1);
                        }

                        /*
                         * Inserts information about the new ANN.
                         */
                        sql = String.format(SQL_QUERY_INSERT_ANN, annKindId, annFitness,
                                        annActivitiesWeights);
                        insert.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
                        ResultSet generatedAnnId = insert.getGeneratedKeys();
                        if (generatedAnnId.next()) {

                                /*
                                 * Gets the automatically generated ID of the new ANN.
                                 */
                                annId = generatedAnnId.getInt(1);
                        }
                } catch (Exception ex) {
                        System.err.println(Texts.ERROR_ADD_ANN);
                        ex.printStackTrace();
                }
                return annId;
        }

        /**
         * Deletes an ANN from the remote database server.
         *
         * @param id
         *            Unique identifier of the ANN.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 27 Nov 2011
         */
        public void deleteAnn(int id) {

                /*
                 * SQL string.
                 */
                String sql = "";

                try {

                        /*
                         * Deletes information about the ANN kind.
                         */
                        Statement delete = connection.createStatement();
                        sql = String.format(SQL_QUERY_DELETE_ANN_KIND, id);
                        delete.executeUpdate(sql);

                        /*
                         * Deletes information about the ANN.
                         */
                        sql = String.format(SQL_QUERY_DELETE_ANN, id);
                        delete.executeUpdate(sql);
                } catch (Exception ex) {
                        System.err.println(Texts.ERROR_DELETE_ANN);
                        ex.printStackTrace();
                }
        }
}
```

DeleteAnnPane.java

```
package eu.veldsoft.backend;

/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                         Iliyan Zankinski  ( iliyan_mf@abv.bg )            *
 *                         Galq Cirkalova    ( galq_cirkalova@abv.bg )       *
 *                         Ivan Grozev       ( ivan.iliev.grozev@gmail.com ) *
 *                         Momchil Anachkov  ( mZer0000@gmail.com )          *
```

```
 *                          Ralitza Koleva   ( rallly@abv.bg )              *
 *                                                                          *
 * This program is free software: you can redistribute it and/or modify     *
 * it under the terms of the GNU General Public License as published by     *
 * the Free Software Foundation, either version 3 of the License, or        *
 * (at your option) any later version.                                      *
 *                                                                          *
 * This program is distributed in the hope that it will be useful,          *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of           *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the            *
 * GNU General Public License for more details.                             *
 *                                                                          *
 * You should have received a copy of the GNU General Public License        *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.     *
 *                                                                          *
 ****************************************************************************/

import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JComboBox;

/**
 * Panel with GUI controls for Neural Network Management.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 27 Nov 2010
 */
class DeleteAnnPane extends JPanel {

            /**
             * Default serial version UID.
             */
            private static final long serialVersionUID = 1L;

            /**
             * Parent applet reference.
             */
            private VitoshaTradeApplet parent = null;

            /**
             * GUI control for obtaining and selecting network ID.
             */
            private JComboBox networkId = new JComboBox();

            /**
             * GUI control for deleting a selected network.
             */
            private JButton delete = new JButton();

            /**
             * GUI control for updating ANN server information.
             */
            private JButton refresh = new JButton();

            /**
             * Constructing delete ANN pane.
             *
             * @param parent
             *            The parent class.
             *
             * @author Ralitza Koleva
             *
             * @email rallly@abv.bg
             *
             * @date 27 Nov 2011
             */
            public DeleteAnnPane(final VitoshaTradeApplet parent) {
                        this.parent = parent;
                        this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
                                                VitoshaTradeApplet.EAST_PANE_HEIGHT));

                        setLayout(new GridLayout(25, 1));

                        add(new JLabel(Texts.LABEL_ANN_ID));
                        add(networkId);
                        networkId.setEditable(false);

                        /*
                         * Acts as separator.
                         */
                        add(new JLabel());

                        add(delete);
                        delete.setText(Texts.LABEL_BUTTON_DELETE);
                        delete.addActionListener(new ActionListener() {

                                    /**
                                     * Deletes all ANN information.
                                     *
                                     * @author Ralitza Koleva
                                     *
                                     * @email rallly@abv.bg
                                     *
                                     * @date 27 Nov 2011
                                     */
                                    public void actionPerformed(ActionEvent event) {
                                                try {
                                                            int ann_id = Integer.parseInt((String) networkId
                                                                                    .getSelectedItem());
                                                            parent.dbHelp.deleteAnn(ann_id);
                                                            loadAllAnnIds();
                                                            parent.workArea.repaint();
                                                } catch (Exception ex) {
                                                            showInformationMessage();
                                                }
                                    }
                        });

                        /*
                         * Acts as separator.
                         */
                        add(new JLabel());

                        add(refresh);
                        refresh.setText(Texts.LABEL_BUTTON_REFRESH);
```

```
                refresh.addActionListener(new ActionListener() {

                        /**
                         * Refreshes ANN IDs in the ANN ID JComboBox.
                         *
                         * @author Ralitza Koleva
                         *
                         * @email rallly@abv.bg
                         *
                         * @date 27 Nov 2011
                         */
                        public void actionPerformed(ActionEvent event) {
                                loadAllAnnIds();
                        }
                });
        }

        /**
         * Loads all ANN IDs in the ANN ID JComboBox.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 29 Oct 2011
         */
        public void loadAllAnnIds() {
                String listIds[][] = parent.dbHelp.loadAnnList();
                networkId.removeAllItems();
                networkId.addItem("---");
                for (int i = 0; i < listIds.length; i++) {
                        networkId.addItem(listIds[i][2]);
                }
        }

        /**
         * Shows a message if no ANN ID is selected to delete.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 05 Dec 2011
         */
        private void showInformationMessage() {
                InformationMessages error = new InformationMessages(
                                Texts.INFORMATION_SELECT_ANN_ID,
                                Texts.INFORMATION_SELECT_ANN_ID_TITLE,
                                JOptionPane.INFORMATION_MESSAGE);
                error.showMessage();
        }
}
```

DrawingPane.java

```
package eu.veldsoft.backend;

/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )        *
 *                            Ralitza Koleva   ( rallly@abv.bg )             *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.FileInputStream;
import java.util.Properties;
import java.util.Random;

import javax.swing.JPanel;

/**
 * Panel with ANN drawoing functionality.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 10 Jan 2011
 */
class DrawingPane extends JPanel {
        /**
         * Default serial version UID.
         */
        private static final long serialVersionUID = 1L;

        /**
         * Selection alpha value.
         */
        private static int SELECTION_ALPHA = 127;
```

```
/**
 * Regular neuron color.
 */
private Color regularNeuronColor;

/**
 * Bias neuron color.
 */
private Color biasNeuronColor;

/**
 * Input neuron color.
 */
private Color inputNeuronColor;

/**
 * Output neuron color.
 */
private Color outputNeuronColor;

/**
 * Input-Output neuron color.
 */
private Color inputOutputNeuronColor;

/**
 * Input-Bias neuron color.
 */
private Color inputBiasNeuronColor;

/**
 * Output-Bias neuron color.
 */
private Color outputBiasNeuronColor;

/**
 * Input-Output-Bias neuron color.
 */
private Color inputOutputBiasNeuronColor;

/**
 * Connections color.
 */
private Color connectionColor = new Color(0, 0, 0);

/**
 * Helps making the connections colors, different colors depending on
 * activities/weights/both.
 */
double connectionColorTemp = 0;

/**
 * No index selected constant.
 */
private static final int INDEX_NOT_SELECTED = -1;

/**
 * Neuron visual radius.
 */
private int neuronRadius = 0;

/**
 * Parent applet reference.
 */
private VitoshaTradeApplet parent = null;

/**
 * Keep selection of first neuron index during visualization.
 */
private int firstNeuronIndex = INDEX_NOT_SELECTED;

/**
 * Keep selection of second neuron index during visualization.
 */
private int secondNeuronIndex = INDEX_NOT_SELECTED;

/**
 * Drawing pane constructor.
 *
 * @param parent
 *            The parent class.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 10 Jan 2011
 */
public DrawingPane(final VitoshaTradeApplet parent) {
            this.parent = parent;
            this.setPreferredSize(new Dimension(
                                    VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                    VitoshaTradeApplet.CENTRAL_PANE_HEIGHT));

            // TODO Do it in size changed listener.
            neuronRadius = calculateNeuronRadius();

            /*
             * Mouse listener for click events to select components (neurons or
             * connections).
             */
            this.addMouseListener(new MouseListener() {

                        public void mouseClicked(MouseEvent event) {
                        }

                        public void mouseEntered(MouseEvent event) {
                        }

                        public void mouseExited(MouseEvent event) {
                        }

                        public void mousePressed(MouseEvent event) {
                                    /*
                                     * Do nothing if there is no coordinates.
                                     */
                                    if (parent.ann == null) {
                                                return;
                                    }

                                    int x = event.getX();
                                    int y = event.getY();
                                    boolean controlKeyDown = event.isControlDown();

                                    /*
```

```
                         * Find selected neuron index.
                         */
                        int xi;
                        int yi;
                        int squareDistance;
                        int coordinates[][] = parent.ann.coordinates;
                        int selectedNeuronIndex = INDEX_NOT_SELECTED;
                        for (int i = 0; i < coordinates.length; i++) {
                                xi = coordinates[i][0];
                                yi = coordinates[i][1];

                                squareDistance = (x - xi) * (x - xi) + (y - yi) * (y - yi);

                                if (neuronRadius * neuronRadius >= squareDistance) {
                                        selectedNeuronIndex = i;
                                        break;
                                }
                        }

                        /*
                         * Display connection properties if connection is selected.
                         */
                        if (selectedNeuronIndex != INDEX_NOT_SELECTED
                                        && firstNeuronIndex != INDEX_NOT_SELECTED
                                        && secondNeuronIndex == INDEX_NOT_SELECTED
                                        && controlKeyDown == true) {
                                secondNeuronIndex = selectedNeuronIndex;

                                repaint();
                                parent.showConnectionPane();
                                parent.connectionPane
                                                .setValues(
                                                                firstNeuronIndex,
                                                                selectedNeuronIndex,
                                                                parent.ann.activities[firstNeuronIndex][secondNeuronIndex],
                                                                parent.ann.weights[firstNeuronIndex][secondNeuronIndex]);
                                return;
                        }

                        /*
                         * Start selection of connection.
                         */
                        if (selectedNeuronIndex != INDEX_NOT_SELECTED
                                        && firstNeuronIndex == INDEX_NOT_SELECTED
                                        && secondNeuronIndex == INDEX_NOT_SELECTED
                                        && controlKeyDown == true) {
                                firstNeuronIndex = selectedNeuronIndex;
                                secondNeuronIndex = INDEX_NOT_SELECTED;

                                return;
                        }

                        /*
                         * Select single neuron.
                         */
                        if (selectedNeuronIndex != INDEX_NOT_SELECTED
                                        && firstNeuronIndex == INDEX_NOT_SELECTED
                                        && secondNeuronIndex == INDEX_NOT_SELECTED
                                        && controlKeyDown == false) {
                                firstNeuronIndex = selectedNeuronIndex;
                                secondNeuronIndex = INDEX_NOT_SELECTED;

                                parent.showNeuronPane();
                                parent.neuronPane
                                                .setValues(
                                                                firstNeuronIndex,
                                                                parent.ann.flags[firstNeuronIndex],
                                                                parent.ann.coordinates[firstNeuronIndex][0],
                                                                parent.ann.coordinates[firstNeuronIndex][1],
                                                                parent.ann.activities[firstNeuronIndex][firstNeuronIndex],
                                                                parent.ann.weights[firstNeuronIndex][firstNeuronIndex]);

                                return;
                        }

                        /*
                         * If nothing is selected show network pane.
                         */
                        if (selectedNeuronIndex == INDEX_NOT_SELECTED) {
                                firstNeuronIndex = INDEX_NOT_SELECTED;
                                secondNeuronIndex = INDEX_NOT_SELECTED;
                                parent.showNetworkPane();
                                return;
                        }
                }

                public void mouseReleased(MouseEvent event) {
                        /*
                         * Selection of connections needs two mouse clicks.
                         */
                        if (event.isControlDown() == true) {
                                return;
                        }

                        /*
                         * After mouse button release it is better neuron not to be
                         * selected.
                         */
                        firstNeuronIndex = INDEX_NOT_SELECTED;
                        secondNeuronIndex = INDEX_NOT_SELECTED;
                }
        });

        /*
         * Drag neuron source code (changing neuron position).
         */
        addMouseMotionListener(new MouseMotionListener() {
                public void mouseDragged(MouseEvent event) {
                        if (firstNeuronIndex != INDEX_NOT_SELECTED
                                        && secondNeuronIndex == INDEX_NOT_SELECTED) {
                                parent.ann.coordinates[firstNeuronIndex][0] = event.getX();
                                parent.ann.coordinates[firstNeuronIndex][1] = event.getY();

                                parent.neuronPane.axisX.setText(""
                                                + parent.ann.coordinates[firstNeuronIndex][0]);
                                parent.neuronPane.axisY.setText(""
                                                + parent.ann.coordinates[firstNeuronIndex][1]);

                                repaint();
                        }
                }

                public void mouseMoved(MouseEvent event) {
                }
        });
```

```
}

/**
 * Panel paint.
 *
 * @param g
 *            Graphic context.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 10 Jan 2011
 */
public void paint(Graphics g) {
            ArtificialNeuralNetwork ann = parent.ann;

            if (ann == null) {
                        return;
            }

            /*
             * Calls a function to load the work area background color.
             */
            selectBackgroundColor(g);

            /*
             * Clear background.
             */
            g.fillRect(0, 0, getWidth(), getHeight());

            /*
             * Draw neurons connections between i and j neurons.
             */
            g.setColor(connectionColor);
            for (int j = 0; j < ann.numberOfNeurons; j++) {
                        for (int i = j + 1; i < ann.numberOfNeurons; i++) {

                                    /*
                                     * Do not draw selected connection.
                                     */
                                    if ((i == firstNeuronIndex && j == secondNeuronIndex)
                                                            || (j == firstNeuronIndex && i == secondNeuronIndex)) {
                                                continue;
                                    }

                                    /*
                                     * Draw line between coordinates of two neurons.
                                     */
                                    selectConnectionsColorByChosenMesh(i, j, g);
                        }
            }

            /*
             * Selected connection should be with different color.
             */
            g.setColor(new Color(connectionColor.getRed(), connectionColor
                                    .getGreen(), connectionColor.getBlue(), SELECTION_ALPHA));
            for (int j = 0; j < ann.numberOfNeurons; j++) {
                        for (int i = 0; i < ann.numberOfNeurons; i++) {
                                    if ((i == firstNeuronIndex && j == secondNeuronIndex)
                                                            || (j == firstNeuronIndex && i == secondNeuronIndex)) {
                                                /*
                                                 * Draw line between coordinates of two neurons of selected
                                                 * connection.
                                                 */
                                                g.drawLine(ann.coordinates[i][0], ann.coordinates[i][1],
                                                                        ann.coordinates[j][0], ann.coordinates[j][1]);
                                    }
                        }
            }

            /*
             * Set neuron types colors.
             */
            regularNeuronColor = readNeuronColor(Util.REGULAR_NEURON_COLOR_PROPERTY_KEY);
            inputNeuronColor = readNeuronColor(Util.INPUT_NEURON_COLOR_PROPERTY_KEY);
            outputNeuronColor = readNeuronColor(Util.OUTPUT_NEURON_COLOR_PROPERTY_KEY);
            biasNeuronColor = readNeuronColor(Util.BIAS_NEURON_COLOR_PROPERTY_KEY);
            inputOutputNeuronColor = readNeuronColor(Util.INPUT_OUTPUT_NEURON_COLOR_PROPERTY_KEY);
            inputBiasNeuronColor = readNeuronColor(Util.INPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);
            outputBiasNeuronColor = readNeuronColor(Util.OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);
            inputOutputBiasNeuronColor = readNeuronColor(Util.INPUT_OUTPUT_BIAS_NEURON_COLOR_PROPERTY_KEY);

            /*
             * Draw neurons.
             */
            Color neuronColor = null;
            for (int i = 0; i < ann.numberOfNeurons; i++) {
                        /*
                         * Change color for different neurons type. It is possible neuron to
                         * have more than one flag up (example neuron which is input and
                         * output).
                         */
                        switch (ann.flags[i]) {
                        case (ArtificialNeuralNetwork.REGULAR_NEURON):
                                    neuronColor = regularNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.BIAS_NEURON):
                                    neuronColor = biasNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.INPUT_NEURON):
                                    neuronColor = inputNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.BIAS_NEURON | ArtificialNeuralNetwork.INPUT_NEURON):
                                    neuronColor = inputBiasNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.OUTPUT_NEURON):
                                    neuronColor = outputNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.BIAS_NEURON | ArtificialNeuralNetwork.OUTPUT_NEURON):
                                    neuronColor = outputBiasNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.INPUT_NEURON | ArtificialNeuralNetwork.OUTPUT_NEURON):
                                    neuronColor = inputOutputNeuronColor;
                                    break;
                        case (ArtificialNeuralNetwork.BIAS_NEURON
                                                | ArtificialNeuralNetwork.INPUT_NEURON | ArtificialNeuralNetwork.OUTPUT_NEURON):
                                    neuronColor = inputOutputBiasNeuronColor;
                                    break;
                        }

                        /*
                         * Selected neuron should be with different color.
                         */
```

```
                        if (i == firstNeuronIndex
                                        && secondNeuronIndex == INDEX_NOT_SELECTED) {
                                g.setColor(new Color(neuronColor.getRed(), neuronColor
                                                .getGreen(), neuronColor.getBlue(), SELECTION_ALPHA));
                        } else {
                                g.setColor(neuronColor);
                        }

                        /*
                         * Gets the neuron radius size.
                         */
                        neuronRadius = calculateNeuronRadius();

                        /*
                         * Draw circle on coordinates of neurons.
                         */
                        g.fillOval(ann.coordinates[i][0] - neuronRadius,
                                        ann.coordinates[i][1] - neuronRadius, neuronRadius * 2,
                                        neuronRadius * 2);

                        /*
                         * Shows the ID of each neuron on the screen.
                         */
                        showNeuronNumbers(g, i, neuronColor, ann.coordinates[i][0] - 3,
                                        ann.coordinates[i][1] + 3);
                }
        }

        /**
         * Calculates neurons radius.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 06 Dec 2011
         */
        private int calculateNeuronRadius() {

                /**
                 * Default neuron radius.
                 */
                int neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 100;
                /**
                 * Default neuron radius size.
                 */
                String neuronRadiusSize = "Small";
                try {
                        Properties p = new Properties();
                        FileInputStream in = new FileInputStream(Util.PROPERTIES_FILE_NAME);
                        p.load(in);
                        neuronRadiusSize = p.getProperty("NeuronsSize");
                        if (neuronRadiusSize.equals("Small")) {
                                neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                                VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 100;
                        } else if (neuronRadiusSize.equals("Medium")) {
                                neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                                VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 80;
                        } else if (neuronRadiusSize.equals("Large")) {
                                neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                                VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 70;
                        }

                } catch (Exception ex) {
                        neuronRadius = Math.min(VitoshaTradeApplet.CENTRAL_PANE_WIDTH,
                                        VitoshaTradeApplet.CENTRAL_PANE_HEIGHT) / 100;
                }
                return (neuronRadius);
        }

        /**
         * Reads the current color of the neuron from a file, depending on its type
         * (flag).
         *
         * @param colorProperty
         *            The parameter name, containing the color.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 26 Oct 2011
         *
         * @return The current color of the neuron.
         */
        public Color readNeuronColor(String colorProperty) {
                Color neuronColor = new Color(0, 0, 0);
                try {
                        Properties properties = new Properties();
                        FileInputStream in = new FileInputStream(Util.PROPERTIES_FILE_NAME);
                        properties.load(in);
                        int color = Integer.parseInt(properties.getProperty(colorProperty));
                        neuronColor = new Color(color);
                } catch (Exception ex) {
                        neuronColor = new Color(Util.PRNG.nextInt(256),
                                        Util.PRNG.nextInt(256), Util.PRNG.nextInt(256));
                }
                return (neuronColor);
        }

        /**
         * Draws the connection in a specific color depending on the chosen radio
         * button in menu Tools->Mesh.
         *
         * @param firstNeuronIndex
         *            Index of the first neuron in the connection.
         *
         * @param secondNeuronIndex
         *            Index of the second neuron in the connection.
         *
         * @param g
         *            Graphic context.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 31 Oct 2011
         */
        public void selectConnectionsColorByChosenMesh(int firstNeuronIndex,
                        int secondNeuronIndex, Graphics g) {

                /*
                 * Takes minimum and maximum activity, weight and both activity and
```

```
             * weight.
             */
            double minActivity = parent.ann.activities[0][0];
            double maxActivity = parent.ann.activities[0][0];
            double minWeight = parent.ann.weights[0][0];
            double maxWeight = parent.ann.weights[0][0];
            double minActivityWeight = parent.ann.activities[0][0]
                                    * parent.ann.weights[0][0];
            double maxActivityWeight = parent.ann.activities[0][0]
                                    * parent.ann.weights[0][0];
            for (int j = 0; j < parent.ann.numberOfNeurons; j++) {
                        for (int i = 0; i < parent.ann.numberOfNeurons; i++) {
                                    if (parent.ann.activities[i][j] < minActivity)
                                                minActivity = parent.ann.activities[i][j];
                                    if (parent.ann.activities[i][j] > maxActivity)
                                                maxActivity = parent.ann.activities[i][j];
                                    if (parent.ann.weights[i][j] < minWeight)
                                                minWeight = parent.ann.weights[i][j];
                                    if (parent.ann.weights[i][j] > maxWeight)
                                                maxWeight = parent.ann.weights[i][j];
                                    if (parent.ann.activities[i][j] * parent.ann.weights[i][j] < minActivityWeight)
                                                minActivityWeight = parent.ann.activities[i][j]
                                                                        * parent.ann.weights[i][j];
                                    if (parent.ann.activities[i][j] * parent.ann.weights[i][j] > maxActivityWeight)
                                                maxActivityWeight = parent.ann.activities[i][j]
                                                                        * parent.ann.weights[i][j];
                        }
            }

            /*
             * If option "Solid" is chosen in "Mesh" menu - all connections are in
             * black color.
             */
            if (parent.meshSolidItem.isSelected() == true) {
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                                parent.ann.coordinates[firstNeuronIndex][1],
                                                parent.ann.coordinates[secondNeuronIndex][0],
                                                parent.ann.coordinates[secondNeuronIndex][1]);

                        /*
                         * If option "Activities" is chosen in "Mesh" menu - connections
                         * colors depend on their activities.
                         */
            } else if (parent.meshActivitiesItem.isSelected() == true) {
                        connectionColorTemp = parent.ann.activities[firstNeuronIndex][secondNeuronIndex]
                                                - minActivity;
                        connectionColorTemp = 255 - connectionColorTemp
                                                * (255 / (maxActivity - minActivity));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));

                        if (parent.ann.activities[firstNeuronIndex][secondNeuronIndex] != 0.0) {
                                    g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                                            parent.ann.coordinates[firstNeuronIndex][1],
                                                            parent.ann.coordinates[secondNeuronIndex][0],
                                                            parent.ann.coordinates[secondNeuronIndex][1]);
                        }

                        /*
                         * If option "Weights" is chosen in "Mesh" menu - connections colors
                         * depend on their weights.
                         */
            } else if (parent.meshWeightsItem.isSelected() == true) {
                        connectionColorTemp = parent.ann.weights[firstNeuronIndex][secondNeuronIndex]
                                                - minWeight;
                        connectionColorTemp = 255 - connectionColorTemp
                                                * (255 / (maxWeight - minWeight));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                                parent.ann.coordinates[firstNeuronIndex][1],
                                                parent.ann.coordinates[secondNeuronIndex][0],
                                                parent.ann.coordinates[secondNeuronIndex][1]);

                        /*
                         * If option "Both" is chosen in "Mesh" menu - connections colors
                         * depend on their activities and weights.
                         */
            } else if (parent.meshBothItem.isSelected() == true) {
                        connectionColorTemp = parent.ann.activities[firstNeuronIndex][secondNeuronIndex]
                                                * parent.ann.weights[firstNeuronIndex][secondNeuronIndex]
                                                - minActivityWeight;
                        connectionColorTemp = 255 - connectionColorTemp
                                                * (255 / (maxActivityWeight - minActivityWeight));
                        int color = (int) Math.round(connectionColorTemp);
                        g.setColor(new Color(color, color, color));
                        g.drawLine(parent.ann.coordinates[firstNeuronIndex][0],
                                                parent.ann.coordinates[firstNeuronIndex][1],
                                                parent.ann.coordinates[secondNeuronIndex][0],
                                                parent.ann.coordinates[secondNeuronIndex][1]);
            }
}

/**
 * Shows the IDs of the neurons on the screen.
 *
 * @param g
 *            Graphic context.
 *
 * @param neuronId
 *            Neuron ID shown on the screen.
 *
 * @param neuronColor
 *            The color of the neuron.
 *
 * @param neuronIdXCoordinate
 *            X-coordinate of the neuron ID.
 *
 * @param neuronIdYCoordinate
 *            Y-coordinate of the neuron ID.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 13 Nov 2011
 */
private void showNeuronNumbers(Graphics g, int neuronId, Color neuronColor,
                        int neuronIdXCoordinate, int neuronIdYCoordinate) {
            if (parent.numberingItem.isSelected()) {

                        /*
                         * Converts the neuron color to its opposite, so that the neuron ID
                         * stays visible inside the neuron.
                         */
```

```
                                int convertedNeuronRedPrimaryColor = (neuronColor.getRed() ^ 0x80) & 0xff;
                                int convertedNeuronGreenPrimaryColor = (neuronColor.getGreen() ^ 0x80) & 0xff;
                                int convertedNeuronBluePrimaryColor = (neuronColor.getBlue() ^ 0x80) & 0xff;

                                /*
                                 * Reads the numbers size from a file. If the process fails, a
                                 * default size is taken.
                                 */
                                int neuronIdFontSize = 9;
                                try {
                                        Properties properties = new Properties();
                                        FileInputStream in = new FileInputStream(
                                                        Util.PROPERTIES_FILE_NAME);
                                        properties.load(in);
                                        int currentNeuronsNumberSize = Integer.parseInt(properties
                                                        .getProperty("NeuronsNumbersSize"));
                                        neuronIdFontSize = currentNeuronsNumberSize;
                                } catch (Exception ex) {
                                }

                                /*
                                 * Creates a new font for the neuron IDs.
                                 */
                                String neuronIdFontFamily = "Verdana";
                                Font neuronIdFont = new Font(neuronIdFontFamily, Font.BOLD,
                                                neuronIdFontSize);
                                Color neuronIdColor = new Color(convertedNeuronRedPrimaryColor,
                                                convertedNeuronGreenPrimaryColor,
                                                convertedNeuronBluePrimaryColor);

                                /*
                                 * Sets font and color of the neuron ID and draws it inside the
                                 * neuron oval.
                                 */
                                g.setColor(neuronIdColor);
                                g.setFont(neuronIdFont);
                                g.drawString(Integer.toString(neuronId), neuronIdXCoordinate,
                                                neuronIdYCoordinate);
                        }
                }

                /**
                 * Reads the work area background color.
                 *
                 * @param g
                 *            Graphic context.
                 *
                 * @author Ralitza Koleva
                 *
                 * @email rallly@abv.bg
                 *
                 * @date 23 Nov 2011
                 */
                private void selectBackgroundColor(Graphics g) {
                        try {
                                Properties properties = new Properties();
                                FileInputStream in = new FileInputStream(Util.PROPERTIES_FILE_NAME);
                                properties.load(in);
                                int color = Integer.parseInt(properties
                                                .getProperty(Util.WORK_AREA_BACKGROUND_COLOR_PROPERTY_KEY));
                                g.setColor(new Color(color));
                        } catch (Exception ex) {
                                g.setColor(new Color(255, 255, 255));
                        }
                }
}
```

InformationMessages.java

```
package eu.veldsoft.backend;

/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )          *
 *                            Ralitza Koleva   ( rallly@abv.bg )               *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

import java.awt.Component;
import javax.swing.JOptionPane;

/**
 * Shows a pop-up message on the screen.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 05 Dec 2011
 */
class InformationMessages {

        /**
         * Frame for the connection error message.
         */
        protected static final Component frame = null;
```

```
            /**
             * Message text.
             */
            private String message;

            /**
             * Message title.
             */
            private String title;

            /**
             * Message type.
             */
            private int messageType;

            /**
             * Constructing pop-up message.
             *
             * @param message
             *            Message text.
             *
             * @param header
             *            Message header.
             *
             * @param messageType
             *            Message type (error/warning/information, etc.).
             *
             * @author Ralitza Koleva
             *
             * @email rallly@abv.bg
             *
             * @date 05 Dec 2011
             */
            public InformationMessages(String message, String header, int messageType) {
                        this.message = message;
                        this.title = header;
                        this.messageType = messageType;
            }

            /**
             * Shows a pop-up message on the screen.
             *
             * @author Ralitza Koleva
             *
             * @email rallly@abv.bg
             *
             * @date 05 Dec 2011
             */
            void showMessage() {
                        JOptionPane.showMessageDialog(frame, message, title, messageType);
                        return;
            }
}
```

LoginPane.java

```
package eu.veldsoft.backend;

/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )          *
 *                            Ralitza Koleva   ( rallly@abv.bg )               *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Properties;

import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JPasswordField;

/**
 * Panel with GUI controls for longin.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 16 Nov 2010
 */
class LoginPane extends JPanel {
            /**
             * Default serial version UID.
             */
            private static final long serialVersionUID = 1L;

            /**
```

```
 * Parent applet reference.
 */
private VitoshaTradeApplet parent = null;

/**
 * GUI control for obtaining username.
 */
private JTextField username = new JTextField();

/**
 * GUI control for obtaining password.
 */
private JPasswordField password = new JPasswordField();

/**
 * GUI control for obtaining password.
 */
private JButton login = new JButton(Texts.LABEL_BUTTON_LOGIN);

/**
 * GUI control for obtaining the database location.
 */
private JTextField dbHost = new JTextField();

/**
 * GUI control for obtaining the database port.
 */
private JTextField dbPort = new JTextField();

/**
 * GUI control for obtaining the database name.
 */
private JTextField dbName = new JTextField();

/**
 * Constructing login pane.
 *
 * @param parent
 *            The parent class.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 16 Nov 2010
 */
public LoginPane(final VitoshaTradeApplet parent) {
            this.parent = parent;
            this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
                                    VitoshaTradeApplet.EAST_PANE_HEIGHT));

            setLayout(new GridLayout(25, 1));

            add(new JLabel(Texts.LABEL_DATABASE_HOST));
            add(dbHost);

            add(new JLabel(Texts.LABEL_DATABASE_PORT));
            add(dbPort);

            add(new JLabel(Texts.LABEL_DATABASE_NAME));
            add(dbName);

            /*
             * Load login properties.
             */
            try {
                        Properties p = new Properties();
                        FileInputStream in = new FileInputStream(Util.PROPERTIES_FILE_NAME);
                        p.load(in);
                        in.close();
                        dbHost.setText(p.getProperty("DatabaseHost"));
                        dbPort.setText(p.getProperty("DatabasePort"));
                        dbName.setText(p.getProperty("DatabaseName"));
            } catch (Exception ex) {
            }

            add(new JLabel(Texts.LABEL_USERNAME));
            add(username);
            add(new JLabel(Texts.LABEL_PASSWORD));
            add(password);
            add(login);

            /*
             * Listener for the login button.
             */
            login.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                    parent.dbHelp.setDatabaseName(dbName.getText());
                                    parent.dbHelp.setHost(dbHost.getText());
                                    try {
                                                parent.dbHelp.setPort((new Integer(dbPort.getText()))
                                                                        .intValue());
                                    } catch (Exception ex) {
                                                System.err.println(Texts.ERROR_DATABASE_PORT);
                                                // TODO Inform user that port should be number.
                                    }
                                    parent.dbHelp.setUsername(username.getText());
                                    parent.dbHelp.setPassword(password.getText());
                                    try {
                                                parent.dbHelp.connect();

                                                /*
                                                 * Show network pane.
                                                 */
                                                parent.showNetworkPane();
                                    } catch (Exception ex) {
                                                System.err.println(Texts.ERROR_DATABASE_CONNECT);
                                                System.err.println(ex);

                                                /*
                                                 * Shows connection error.
                                                 */
                                                showConnectionError();
                                    }

                                    /*
                                     * Save login properties.
                                     */
                                    try {
                                                Properties p = new Properties();
                                                FileInputStream allProperties = new FileInputStream(
                                                                        Util.PROPERTIES_FILE_NAME);
                                                p.load(allProperties);
                                                p.setProperty("DatabaseHost", dbHost.getText());
                                                p.setProperty("DatabasePort", dbPort.getText());
```

```
                                                p.setProperty("DatabaseName", dbName.getText());

                                                FileOutputStream out = new FileOutputStream(
                                                                Util.PROPERTIES_FILE_NAME);
                                                p.store(out, "");
                                                allProperties.close();
                                                out.close();
                                        } catch (Exception ex) {
                                        }
                                }
                        });

                        password.setEchoChar('*');
                }

                /**
                 * Shows error message when login fails.
                 *
                 * @author Ralitza Koleva
                 *
                 * @email rallly@abv.bg
                 *
                 * @date 24 Nov 2011
                 */
                private void showConnectionError() {
                        InformationMessages error = new InformationMessages(
                                        Texts.ERROR_DATABASE_CONNECT,
                                        Texts.ERROR_DATABASE_CONNECT_TITLE, JOptionPane.ERROR_MESSAGE);
                        error.showMessage();
                        parent.showLoginPane();
                }
}
```

NetworkPane.java

```
package eu.veldsoft.backend;

/*****************************************************************************
 *                                                                           *
 * VitoshaTrade is Distributed Artificial Neural Network trained by          *
 * Differential Evolution for prediction of Forex. Project development is in *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia, *
 * the capital of Bulgaria.                                                  *
 *                                                                           *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )             *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )          *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )     *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )        *
 *                            Ralitza Koleva   ( rallly@abv.bg )             *
 *                                                                           *
 * This program is free software: you can redistribute it and/or modify      *
 * it under the terms of the GNU General Public License as published by      *
 * the Free Software Foundation, either version 3 of the License, or         *
 * (at your option) any later version.                                       *
 *                                                                           *
 * This program is distributed in the hope that it will be useful,           *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of            *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the             *
 * GNU General Public License for more details.                              *
 *                                                                           *
 * You should have received a copy of the GNU General Public License         *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.      *
 *                                                                           *
 *****************************************************************************/

import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JComboBox;

/**
 * Panel with GUI controls for Neural Network Management.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Nov 2010
 */
class NetworkPane extends JPanel {

        /**
         * Determines what filter should be applied when symbol and/or period are
         * changed.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 30 Oct 2011
         */
        private class SetFiltersBySimbolAndPeriod implements ItemListener {

                /**
                 * Calls functions to apply different filters depending on the chosen
                 * values in the drop down menus for currency pair and period.
                 *
                 * @author Ralitza Koleva
                 *
                 * @email rallly@abv.bg
                 *
                 * @date 30 Oct 2011
                 */
                public void itemStateChanged(ItemEvent event) {

                        /*
                         * If neither symbol, nor period is chosen, all ANN IDs are shown in
                         * the ANN ID JComboBox.
                         */
                        if ((networkSymbol.getItemCount() == 0 || "---"
```

```
                                        .equals(networkSymbol.getSelectedItem()) == true)
                                        && (networkPeriod.getItemCount() == 0 || "---"
                                                        .equals(networkPeriod.getSelectedItem()) == true)) {

                        /*
                         * Loads all ANN IDs.
                         */
                        setAllAnnIds();
                }

                /*
                 * If both symbol and period are chosen, only corresponding ANN IDs
                 * are shown in the ANN ID JComboBox.
                 */
                else if ((networkSymbol.getItemCount() > 0 && "---"
                                        .equals(networkSymbol.getSelectedItem()) == false)
                                        && (networkPeriod.getItemCount() > 0 && "---"
                                                        .equals(networkPeriod.getSelectedItem()) == false)) {

                        /*
                         * Loads ANN IDs depending on the chosen symbol and period.
                         */
                        try {
                                        setFilteredAnnIdsBySymbolAndPeriod();
                        } catch (Exception e) {
                                        // TODO: handle exception
                        }

                        /*
                         * If only symbol is chosen, only corresponding ANN IDs are
                         * shown in the ANN ID JComboBox.
                         */
                } else if ((networkSymbol.getItemCount() > 0 && "---"
                                        .equals(networkSymbol.getSelectedItem()) == false)
                                        && (networkPeriod.getItemCount() > 0 && "---"
                                                        .equals(networkPeriod.getSelectedItem()) == true)) {

                        /*
                         * Loads ANN IDs depending on the chosen symbol.
                         */
                        try {
                                        setFilteredAnnIdsBySymbol();
                        } catch (Exception e) {
                                        // TODO: handle exception
                        }

                        /*
                         * If only period is chosen, only corresponding ANN IDs are
                         * shown in the ANN ID JComboBox.
                         */
                } else if ((networkSymbol.getItemCount() > 0 && "---"
                                        .equals(networkSymbol.getSelectedItem()) == true)
                                        && (networkPeriod.getItemCount() > 0 && "---"
                                                        .equals(networkPeriod.getSelectedItem()) == false)) {

                        /*
                         * Load ANN IDs depending on the chosen period.
                         */
                        try {
                                        setFilteredAnnIdsByPeriod();
                        } catch (Exception e) {
                                        // TODO: handle exception
                        }
                }
}

/**
 * Shows all ANN IDs in the ANN ID JComboBox.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 29 Oct 2011
 */
public void setAllAnnIds() {
                String listIds[][] = parent.dbHelp.loadAnnList();
                networkId.removeAllItems();
                networkId.addItem("---");
                for (int i = 0; i < listIds.length; i++) {
                                networkId.addItem(listIds[i][2]);
                }
}

/**
 * Shows ANN IDs in the ANN ID JComboBox filtered by symbol and period.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 29 Oct 2011
 */
public void setFilteredAnnIdsBySymbolAndPeriod() {
                String selectedSymbol = networkSymbol.getSelectedItem().toString();
                int selectedPeriod = ((SymbolPeriodKeyValue) networkPeriod
                                        .getSelectedItem()).getKey();
                networkId.removeAllItems();
                networkId.addItem("---");
                String listFilteredId[][] = parent.dbHelp.loadAnnList(
                                        selectedSymbol, selectedPeriod);
                try {
                                for (int i = 0; i < listFilteredId.length; i++) {
                                        networkId.addItem(listFilteredId[i][2]);
                                }
                } catch (Exception e) {

                }
}

/**
 * Shows ANN IDs in the ANN ID JComboBox filtered by period.
 *
 * @author Ralitza Koleva
 *
 * @email rallly@abv.bg
 *
 * @date 29 Oct 2011
 */
public void setFilteredAnnIdsByPeriod() {
                int selectedPeriod = ((SymbolPeriodKeyValue) networkPeriod
                                        .getSelectedItem()).getKey();
                networkId.removeAllItems();
                networkId.addItem("---");
                String listFilteredId[][] = parent.dbHelp
```

```
                                        .loadAnnListByPeriod(selectedPeriod);
                        try {
                                for (int i = 0; i < listFilteredId.length; i++) {
                                        networkId.addItem(listFilteredId[i][2]);
                                }
                        } catch (Exception e) {

                        }
                }

                /**
                 * Shows ANN IDs in the ANN ID JComboBox filtered by symbol.
                 *
                 * @author Ralitza Koleva
                 *
                 * @email rallly@abv.bg
                 *
                 * @date 29 Oct 2011
                 */
                public void setFilteredAnnIdsBySymbol() {
                        String selectedSymbol = networkSymbol.getSelectedItem().toString();
                        networkId.removeAllItems();
                        networkId.addItem("---");
                        String listFilteredId[][] = parent.dbHelp
                                        .loadAnnListBySymbol(selectedSymbol);
                        try {
                                for (int i = 0; i < listFilteredId.length; i++) {
                                        networkId.addItem(listFilteredId[i][2]);
                                }
                        } catch (Exception e) {

                        }
                }
};

/**
 * Default serial version UID.
 */
private static final long serialVersionUID = 1L;

/**
 * Parent applet reference.
 */
private VitoshaTradeApplet parent = null;

/**
 * GUI control for obtaining and selecting network symbol.
 */
private JComboBox networkSymbol = new JComboBox();

/**
 * GUI control for obtaining and selecting network period.
 */
private JComboBox networkPeriod = new JComboBox();

/**
 * GUI control for obtaining and selecting network ID.
 */
private JComboBox networkId = new JComboBox();

/**
 * GUI control for loading a selected network.
 */
private JButton load = new JButton();

/**
 * GUI control for saving a selected network.
 */
private JButton save = new JButton();

/**
 * GUI control for rearrange neurons.
 */
private JButton rearrange = new JButton();

/**
 * GUI control for updating ANN server information.
 */
private JButton refresh = new JButton();

/**
 * Constructing network pane.
 *
 * @param parent
 *            The parent class.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Nov 2010
 */
public NetworkPane(final VitoshaTradeApplet parent) {
            this.parent = parent;
            this.setPreferredSize(new Dimension(VitoshaTradeApplet.EAST_PANE_WIDTH,
                                VitoshaTradeApplet.EAST_PANE_HEIGHT));

            setLayout(new GridLayout(25, 1));

            add(new JLabel(Texts.LABEL_SYMBOL));
            add(networkSymbol);
            networkSymbol.setEditable(false);
            networkSymbol.addItemListener(new SetFiltersBySimbolAndPeriod());

            add(new JLabel(Texts.LABEL_PERIOD));
            add(networkPeriod);
            networkPeriod.setEditable(false);
            networkPeriod.addItemListener(new SetFiltersBySimbolAndPeriod());

            add(new JLabel(Texts.LABEL_ANN_ID));
            add(networkId);
            networkId.setEditable(false);

            /*
             * Acts as separator.
             */
            add(new JLabel());

            add(load);
            load.setText(Texts.LABEL_BUTTON_LOAD);
            load.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try {
                                        int ann_id = Integer.parseInt((String) networkId
                                                        .getSelectedItem());
```

```
                                        parent.ann = parent.dbHelp.loadAnn(ann_id);
                                        parent.workArea.repaint();
                                } catch (Exception ex) {
                                        showInformationMessage();
                                }
                        }
                });

                add(save);
                save.setText(Texts.LABEL_BUTTON_SAVE);
                save.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try {
                                        parent.dbHelp.saveAnn(parent.ann);
                                } catch (Exception ex) {
                                        showInformationMessage();
                                }
                        }
                });

                /*
                 * Acts as separator.
                 */
                add(new JLabel());

                add(rearrange);
                rearrange.setText(Texts.LABEL_BUTTON_REARRANGE);
                rearrange.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                if (parent == null) {
                                        return;
                                }
                                if (parent.ann == null) {
                                        return;
                                }
                                if (parent.ann.flags == null) {
                                        return;
                                }

                                int width = parent.workArea.getWidth();
                                int height = parent.workArea.getHeight();

                                int idy = height / (parent.ann.numberOfInputNeurons() + 2);
                                int hdy = height / (parent.ann.numberOfHiddenNeurons() + 2);
                                int ody = height / (parent.ann.numberOfOutputNeurons() + 2);

                                for (int k = 0, i = idy, h = hdy, o = ody; k < parent.ann.flags.length; k++) {
                                        if (parent.ann.flags[k] == ArtificialNeuralNetwork.INPUT_NEURON) {
                                                parent.ann.coordinates[k][0] = 20;
                                                parent.ann.coordinates[k][1] = i;
                                                i += idy;
                                        }
                                        if (parent.ann.flags[k] == ArtificialNeuralNetwork.REGULAR_NEURON) {
                                                parent.ann.coordinates[k][0] = width / 2;
                                                parent.ann.coordinates[k][1] = h;
                                                h += hdy;
                                        }
                                        if (parent.ann.flags[k] == ArtificialNeuralNetwork.OUTPUT_NEURON) {
                                                parent.ann.coordinates[k][0] = width - 20;
                                                parent.ann.coordinates[k][1] = o;
                                                o += ody;
                                        }
                                }
                        }
                });

                /*
                 * Acts as separator.
                 */
                add(new JLabel());

                add(refresh);
                refresh.setText(Texts.LABEL_BUTTON_REFRESH);
                refresh.doClick(10000);
                refresh.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                String listSymbol[][] = parent.dbHelp.loadCurrencyPairs();
                                String listPeriod[][] = parent.dbHelp.loadPeriods();
                                String listIds[][] = parent.dbHelp.loadAnnList();
                                if (listSymbol == null || listPeriod == null || listIds == null) {
                                        return;
                                }

                                networkSymbol.removeAllItems();
                                networkPeriod.removeAllItems();
                                networkSymbol.addItem("---");
                                networkPeriod.addItem("---");

                                /*
                                 * Loads a drop down menu with symbols.
                                 */
                                for (int i = 0; i < listSymbol.length; i++) {
                                        networkSymbol.addItem(listSymbol[i][0]);
                                }

                                /*
                                 * Loads a drop down menu with periods.
                                 */
                                for (int i = 0; i < listPeriod.length; i++) {
                                        networkPeriod.addItem(new SymbolPeriodKeyValue(
                                                        listPeriod[i][0], listPeriod[i][1]));
                                }
                        }
                });
        }

        /**
         * Shows a message if no ANN ID is selected to load/save.
         *
         * @author Ralitza Koleva
         *
         * @email rallly@abv.bg
         *
         * @date 05 Dec 2011
         */
        private void showInformationMessage() {
                InformationMessages error = new InformationMessages(
                                Texts.INFORMATION_SELECT_ANN_ID,
                                Texts.INFORMATION_SELECT_ANN_ID_TITLE,
                                JOptionPane.INFORMATION_MESSAGE);
                error.showMessage();
        }
}
```

NeuronPane.java

```
package eu.veldsoft.backend;

/*******************************************************************************
 *                                                                             *
 * VitoshaTrade is Distributed Artificial Neural Network trained by            *
 * Differential Evolution for prediction of Forex. Project development is in   *
 * Sofia, Bulgaria. Vitosha is a mountain massif, on the outskirts of Sofia,   *
 * the capital of Bulgaria.                                                    *
 *                                                                             *
 * Copyright (C) 2008-2011 by Todor Balabanov  ( tdb@tbsoft.eu )               *
 *                            Iliyan Zankinski ( iliyan_mf@abv.bg )            *
 *                            Galq Cirkalova   ( galq_cirkalova@abv.bg )       *
 *                            Ivan Grozev      ( ivan.iliev.grozev@gmail.com ) *
 *                            Momchil Anachkov ( mZer0000@gmail.com )          *
 *                            Ralitza Koleva   ( rallly@abv.bg )               *
 *                                                                             *
 * This program is free software: you can redistribute it and/or modify        *
 * it under the terms of the GNU General Public License as published by        *
 * the Free Software Foundation, either version 3 of the License, or           *
 * (at your option) any later version.                                         *
 *                                                                             *
 * This program is distributed in the hope that it will be useful,             *
 * but WITHOUT ANY WARRANTY; without even the implied warranty of              *
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the               *
 * GNU General Public License for more details.                                *
 *                                                                             *
 * You should have received a copy of the GNU General Public License           *
 * along with this program. If not, see <http://www.gnu.org/licenses/>.        *
 *                                                                             *
 *******************************************************************************/

import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JCheckBox;
import javax.swing.JTextField;

/**
 * Panel with GUI controls for Neuron Management.
 *
 * @author Momchil Anachkov
 *
 * @email mZer0000@gmail.com
 *
 * @date 18 Nov 2010
 */
class NeuronPane extends JPanel {

	/**
	 * Default serial version UID.
	 */
	private static final long serialVersionUID = 1L;

	/**
	 * Parent applet reference.
	 */
	private VitoshaTradeApplet parent = null;

	/**
	 * GUI control for obtaining the ID of a specific neuron.
	 */
	private JTextField neuronNumber = new JTextField();

	/**
	 * GUI control for obtaining and setting the input flag for a specific
	 * neuron.
	 */
	private JCheckBox inputCheckbox;

	/**
	 * GUI control for obtaining and setting the output flag for a specific
	 * neuron.
	 */
	private JCheckBox outputCheckbox;

	/**
	 * GUI control for obtaining and setting the regular flag for a specific
	 * neuron.
	 */
	private JCheckBox regularCheckbox;

	/**
	 * GUI control for obtaining and setting the BIAS flag for a specific
	 * neuron.
	 */
	private JCheckBox biasCheckbox;

	/**
	 * GUI control for obtaining and setting X location of a specific neuron.
	 */
	JTextField axisX = new JTextField();

	/**
	 * GUI control for obtaining and setting the Y location of a specific
	 * neuron.
	 */
	JTextField axisY = new JTextField();

	/**
	 * GUI control for obtaining neuron signal.
	 */
	JTextField signal = new JTextField();

	/**
	 * GUI control for obtaining neuron error.
	 */
	JTextField error = new JTextField();

	/**
	 * Constructing neuron pane.
	 *
	 * @param parent
	 *            The parent class.
	 *
	 * @author Momchil Anachkov
```