Министерство образования Российской Федерации

Томский политехнический университет

Факультет Автоматики и вычислительной техники
Направление: Информатика и вычислительная техника
Кафедра Вычислительной техники

Разработка генетического алгоритма настройки искусственной нейронной сети

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе

Магистерская диссертация
(Обозначение документа)

Студент _____ _____ **Цой Ю.Р.**
(Подпись) (Дата) (Фамилия И.О.)

Руководитель _____ _____ **Спицын В.Г.**

Консультанты:

по _____ _____ _____ _____
по _____ _____ _____ _____
по _____ _____ _____ _____
по _____ _____ _____ _____

Допустить к защите:
Заведующий кафедрой
д. т. н., проф.

_____ _____ /Марков Н.Г./
(Подпись) (Дата) (Фамилия И.О.)

ТОМСК – 2004 г.

Министерство образования Российской Федерации

Томский политехнический университет

Кафедра Вычислительной техники
УТВЕРЖДАЮ:
Зав. кафедрой_____
(Подпись, дата)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Студенту Цою Юрию Робертовичу

1. Тема выпускной квалификационной работы_____
Разработка генетического алгоритма настройки
искусственной нейронной сети

утверждена приказом ректора (распоряжением декана) от_____№_____

2. Срок сдачи студентом готовой работы _____

3. Исходные данные к работе_____

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Глава 1. Теоретические сведения

Глава 2. Алгоритм NEvA

Глава 3. Результаты работы алгоритма

Глава 4. Исследование структур получаемых ИНС

ПРИЛОЖЕНИЕ 1. Возможности применения алгоритма NEvA

ПРИЛОЖЕНИЕ 2. Программа EasyNet

5. Перечень графического материала (с точным указанием обязательных чертежей)

5.1. Рисунки

_____

5.2. Таблицы

_____

5.3.

_____

5.4.

_____

5.5.

_____

5.6.

_____

6. Консультанты по разделам выпускной квалификационной работы

(с указанием разделов)

6.1.

_____

6.2.

_____

6.3.

_____

6.4.

_____

7. Дата выдачи задания на выполнение выпускной квалификационной работы

03.02.2003

_____

Руководитель_____

(подпись, дата)

Задание принял к исполнению (студент)

_____

(подпись, дата)

Аннотация

Искусственные нейронные сети (ИНС) являются мощным инструментов анализа данных, в то время как генетические алгоритмы (ГА) известны как робастные методы оптимизации, благодаря способностям к адаптации. Их совместное использование позволило бы объединить адаптивные способности ГА с аналитическими возможностями ИНС.

Настоящая работа посвящена разработанному алгоритму настройки весов и структуры искусственной нейронной сети с помощью генетического алгоритма. Отличительными особенностями разработанного подхода являются гибкий и эффективный эволюционный поиск, сбалансированность структур получаемых ИНС, а также возможность удаления малоинформативных входных признаков.

Пояснительная записка включает 4 главы и 2 приложения. Первая глава содержит краткую информацию о генетических алгоритмах (ГА), искусственных нейронных сетях (ИНС), а также обзор нейроэволюционных методов. Во второй главе приводится описание разработанного алгоритма. Третья глава содержит результаты работы алгоритма. Четвертая глава посвящена исследованию структур получаемых нейронных сетей в зависимости от параметров алгоритма. Возможности применения разработанного алгоритма кратко описаны в Приложении 1. Приложение 2 содержит описание демонстрационной программы EasyNet.

Пояснительная записка содержит 62 страницы, 9 таблиц, 15 рисунков, 7 формул, 47 источников и 2 приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6

Глава 1. Теоретические сведения 8

1.1. Генетические алгоритмы 8

1.2. Искусственные нейронные сети 10

1.3. Нейроэволюционные алгоритмы 13

1.3.1. Преимущества, недостатки и проблемы нейроэволюционного подхода 13

1.3.2. Способы кодирования информации 17

1.3.3. Краткий обзор существующих алгоритмов 20

Глава 2. Алгоритм NEvA 21

2.1. Общее описание 21

2.2. Кодирование информации 22

2.3. Генетические операторы 23

2.4. Выбор вида мутации 27

2.5. Алгоритм расчета выхода ИНС 29

Глава 3. Результаты работы алгоритма 32

3.1. Исключающее ИЛИ 32

3.2. Перевернутый маятник 35

3.3. Преследование жертвы 39

Глава 4. Исследование структур получаемых ИНС 46

4.1. Постановка задачи и условия экспериментов 46

4.2. Результаты 47

4.3. Выводы 50

Заключение 51

Список источников 52

ПРИЛОЖЕНИЕ 1. Возможности использования алгоритма NEvA 56

ПРИЛОЖЕНИЕ 2. Программа EasyNet 57

ВВЕДЕНИЕ

Искусственные нейронные сети (ИНС) и генетические алгоритмы (ГА) являются сравнительно молодыми направлениями исследований в области искусственного интеллекта и принятия решений. Обе концепции используют для функционирования аналоги природных принципов.

Использование генетических алгоритмов для одновременной настройки весов и структуры ИНС называется нейроэволюцией или нейрогенезисом (Д.Уитли, 1995). Вопрос о том, нужно ли «тонко» настраивать топологию сети довольно долго остается открытым, поскольку существующие решения удовлетворительно справляются со своими задачами. Есть мнение, что можно найти топологию сети, которая в дальнейшем быстрее обучается (Д.Уитли, 1990), однако процесс обучения проводится уже на этапе поиска архитектуры, поэтому аргумент отпадает сам собой. В более поздней работе другого исследователя (К.Стенли, 2002) приводится довод, что найденная топология сети позволит разработчикам в дальнейшем учитывать этот опыт и более эффективно использовать ИНС.

Мы попытаемся ответить на этот вопрос с несколько другой стороны. С точки зрения искусственного интеллекта представляют интерес системы, которые способны к самоорганизации и перестройке внутренней структуры только на основе внешних данных. При этом сама система изначально не имеет представления о характере и возможных взаимных связях входных и выходных параметров. Все необходимые зависимости должны появиться в результате адаптивной подстройки. Для ИНС в общем случае входные (выходные) данные между собой не различимы в силу однообразия частей, входящих в структуру нейронной сети. Применяя для обучения и настройки структуры сети генетический алгоритм, важной особенностью которого является способность к адаптации, можно получить самоорганизующуюся систему с достаточно большой степенью универсальности. Для работы этой системе будут необходимы только данные из внешнего окружения, причем их тип и отношение к решаемой задаче не играют большой роли. Таким образом, возможно решение поставленной

проблемы о самоорганизующейся системе, способной решать широкий спектр задач.

Первые работы по настройке ИНС с помощью генетических алгоритмов появились в конце 80-х годов 20 века и уже к 1994 году, согласно (Я.Аландер, 1994), по общему количеству публикаций, начиная с 1957 года, занимали первое место, значительно опередив остальные применения ГА.

Чаще всего нейроэволюционный подход применяется в задачах адаптивного управления, многоагентных системах и системах искусственной жизни.

Актуальность проблемы

Существует много задач, при решении которых с использованием нейросетевых технологий, встает вопрос о выборе структуры ИНС. Зачастую ограничиваются вариантом с регулярной топологией, который не является оптимальным с точки зрения используемых ресурсов компьютера. Кроме этого настройка топологии требует от пользователя достаточно глубокой компетенции в области нейронных сетей. Автоматический выбор структуры сети позволил бы существенно упростить эту проблему.

Целью работы является разработка нейроэволюционного алгоритма, позволяющего одновременно настраивать веса и топологию искусственной нейронной сети.

Практическая значимость

Разработанный алгоритм позволит получить следующие преимущества:

- возможность создания самообучающейся системы, претендующей на достаточно большую универсальность с точки зрения решаемых задач;
- более широкие возможности в адаптивном управлении, поведении, задачах классификации;
- более «тонкая» настройка топологии ИНС, позволяющая сэкономить вычислительные ресурсы;
- возможность создания программных продуктов и программно-аппаратных комплексов, использующих нейросетевые технологии и не требующие от конечного пользователя глубоких знаний о них.

Глава 1. Теоретические сведения

1.1. Генетические алгоритмы

Бурное развитие вычислительной техники в 20 веке привело к тому, что значительная часть вычислительных работ была возложена на ЭВМ. Благодаря большой скорости расчетов, их низкой стоимости и достаточной для многих прикладных задач точности, появилась возможность использовать “тяжелые” с точки зрения вычислений и временных затрат методы решения математических задач. В качестве примеров можно привести переборные методы, итерационные, методы, использующие большие объемы данных (статистические). Однако, наряду с созданными ранее способами решения и алгоритмами стали появляться новые, существование которых отдельно от ЭВМ трудно представить.

Данная работа рассматривает один из таких методов: генетические алгоритмы (ГА). В их основе лежит идея использовать аналоги эволюционных механизмов для поиска решения. Как известно, основными концепциями теории эволюции являются наследственность и естественный отбор. Эти же механизмы используются генетическим алгоритмом для нахождения решения некоторой проблемы [15].

Для работы ГА используют виртуальную популяцию, где гены каждой отдельной особи являются частным решением поставленной задачи. Число генов у особи зависит от числа параметров задачи. В результате оценивания популяции каждой особи ставится в соответствие некоторая величина, которая называется приспособленностью и показывает, насколько успешно данная особь решает данную задачу, т.е. насколько её гены соответствуют поставленным условиям. Более приспособленные особи скрещиваются, из их потомков формируется новая популяция, члены которой оцениваются, затем скрещиваются и т.д. В ходе скрещивания двух особей за счет применения генетических операторов происходит обмен генетической информацией, и получившиеся потомки обладают как свойствами первого родителя, так и свойствами второго.

Алгоритм прекращает работу в одном из следующих случаев:

- найдено решение;
- истекло установленное время работы либо число поколений;
- популяция длительное время не прогрессирует.

В результате работы ГА получается популяция, которая содержит особь, гены которой лучше генов других особей соответствуют требуемым условиям. Данная особь и будет являться найденным с помощью ГА решением. Следует отметить, что найденное решение может и не быть наилучшим, однако оно может быть близко к оптимальному. Отличительной особенностью ГА является то, что вычислительная сложность алгоритма мало зависит от сложности задачи. Значение имеют: вид целевой функции, количество параметров и, если имеется, область ограничений.

Основной областью применения ГА являются задачи оптимизации. К настоящему времени генетические алгоритмы были использованы для решения следующих проблем:

- экстремальные задачи;
- NP-полные проблемы (“Задача коммивояжера” и SAT-проблемы);
- составление расписаний;
- задачи о размещении;
- аппроксимация функций;
- построение минимальных диагностических тестов для задач распознавания;
- настройка и обучение искусственных нейронных сетей;
- игровые стратегии;
- моделирование искусственной жизни;
- развивающиеся агенты и машины.

Вместе с концепцией искусственных нейронных сетей генетические алгоритмы образуют новое направление в искусственном интеллекте, выгодно отличаясь высоким параллелизмом при поиске решения и эффективным сужением пространства поиска в областях оптимумов.

1.2. Искусственные нейронные сети

Основной структурной и функциональной частью нейронной сети является нейрон (рис.1.1). Он состоит из элементов 3 типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапс является функциональным узлом между двумя нейронами и характеризует силу (вес) связи между ними. Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов и внешних входных сигналов. Нелинейный преобразователь реализует функцию одного аргумента – выхода сумматора. Эта функция называется функцией активации или передаточной функцией нейрона [43].

Математическая модель нейрона:

$$S = \sum_{i=1}^{n} w_i x_i + b, \tag{1.1}$$

$$y = f(S), \tag{1.2}$$

где w_i – вес связи, b – значение нейронного смещения, s – результат суммирования, x_i – компонент входного вектора (входной сигнал), y – выходной сигнал нейрона, n – число входов нейрона, f – функция активации.

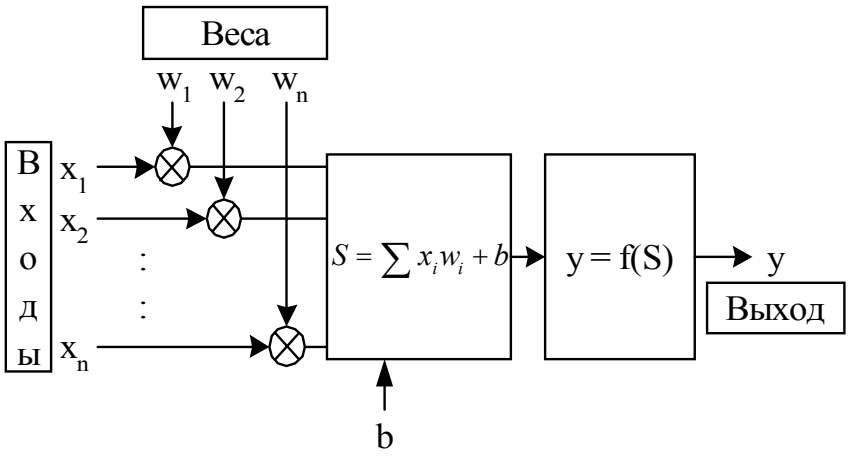


Рис.1.1. Искусственный нейрон

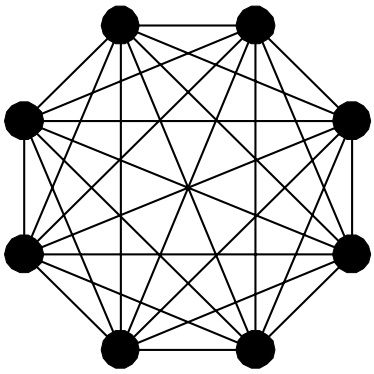
Одними из наиболее часто используемых являются сигмоидные функции активации. Их отличительными особенностями являются непрерывность,

одинарная и двойная дифференцируемость, а также не равенство нулю первой производной на всей области определения.

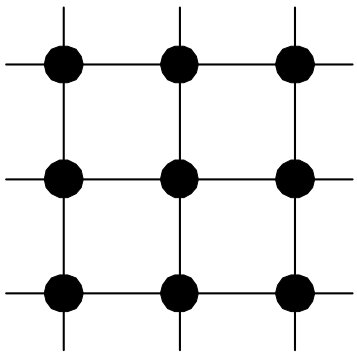
Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов. Чтобы подчеркнуть различие нейронов биологических и искусственных, вторые иногда называют нейроноподобными элементами или формальными нейронами.

В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

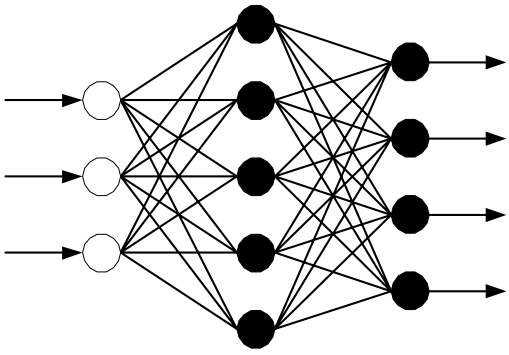
- входные нейроны, на которые подаются входные сигналы;
- выходные нейроны, выходные значения которых представляют выходы всей нейронной сети;
- промежуточные нейроны, не имеют связей с входными сигналами, и их выходные значения не являются выходами сети.



а)



б)



в)

Рис.1.2. Примеры топологий нейронных сетей

а – полносвязная; б – слабосвязная; в – многослойная

С точки зрения топологии можно выделить три основных типа нейронных сетей (рис.1.2):

- полносвязные;
- многослойные (слоистые);
- слабосвязные (с локальными связями).

В полносвязных нейронных сетях каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами сети могут быть сигналы всех или некоторых нейронов.

В многослойных сетях нейроны объединяются в слои. Слой содержит нейроны с одинаковыми входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. Внешние (входные) сигналы подаются на входы нейронов входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы нейронов последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети может быть один или несколько скрытых слоев. Выходные сигналы нейронов слоя (q) являются входными сигналами следующего слоя (q+1).

В слабосвязных нейронных сетях нейроны располагаются в узлах прямоугольной или гексагональной решетки. Каждый нейрон связан с четырьмя (окрестность фон Неймана), шестью (окрестность Голея) или восемью (окрестность Мура) своими ближайшими соседями.

Многослойные нейронные сети можно классифицировать по наличию в них обратных связей.

Сети без обратных связей являются классическим вариантом слоистых сетей и получили название сети прямого распространения, т.к. входные сигналы распространяются по сети последовательно от входного слоя через скрытые слои к выходному.

В сетях с обратными связями выходные сигналы слоя (q) передаются на входы нейронам предыдущих слоев (кроме входного), либо нейронам этого же слоя. Сети с обратными связями называются также рекуррентными [41].

1.3. Нейроэволюционные алгоритмы

1.3.1. Преимущества, недостатки и проблемы нейроэволюционного подхода

Как уже отмечалось выше, нейроэволюционные методы способны одновременно настраивать структуру и веса сети. Это позволяет получать готовые нейросетевые решения, имея только данные из обучающей выборки, и в большинстве случаев не требует от пользователя наличия глубоких знаний из теории ИНС. Тем не менее, в случае обучения с учителем, когда есть наборы входных и соответствующих им выходных данных, а для решения достаточно использовать многослойную сеть прямого распространения, существующие алгоритмы, такие как RProp (Resilient Propagation) или алгоритм сопряженных градиентов, позволяют обучать ИНС за приемлемое время. Использование здесь нейроэволюционного подхода мало оправдывает себя [35], в особенности учитывая значительно большие по сравнению с градиентными методами требования к объему используемой оперативной памяти, т.к. в большинстве случаев используется популяция нейросетей. Возможность получить более оптимальную, чем многослойная, структуру сети является в данном случае незначительным преимуществом, поскольку потребность в ней небольшая, за исключением, возможно, случаев, когда количество нейронов и связей в сети исчисляется сотнями и тысячами.

Ввиду вышеперечисленных недостатков и ограничений нейроэволюционные методы лучше применять в задачах, где градиентные методы неприменимы, либо их эффективность сравнительно невелика. Примером таких задач является обучение ИНС в изменяющейся динамичной среде, что часто встречается в задачах управления [35, 7], а также для поиска оптимальной стратегии поведения [5, 38]. При этом для имеющегося набора входных данных требуемые значения выходов ИНС, как правило, априори неизвестны, и работа сети оценивается по внешнему количественному критерию (целевая функция), отражающему общее качество ее функционирования. «Угадать» структуру ИНС в этом случае довольно

сложно, потому что неизвестно либо малоизвестно, как изменится управляемая система через определенное время, поэтому эволюционный поиск подходящего решения, благодаря высоким адаптивным качествам, выглядит достаточно перспективным вариантом.

Благодаря тому, что генетические алгоритмы представляют унифицированный подход к решению задач оптимизации, нейроэволюционные методы мало зависят от поставленной проблемы, т.е. один и тот же алгоритм может быть применен к совершенно разным задачам, для решения которых нужны ИНС с существенно различными топологиями и функциями активации нейронов. В качестве примера можно упомянуть, что одними из «стандартных» тестовых проблем для нейроэволюционных алгоритмов являются «Исключающее ИЛИ», где требуется настроить очень простую ИНС, и задача балансирования двух перевернутых маятников без информации о скорости их движения, что требует поиска нейронной сети с обратными связями [30].

Одной из основных проблем, возникающих при эволюционном поиске топологии нейронной сети является так называемая проблема одинаковых конкурирующих решений (competing conventions problem) [22]. Другое ее название – проблема пермутаций (permutation problem) [27]. Заключается она в том, что одна и та же ИНС может быть представлена в генотипе особи различным способом. Поскольку основной способ получения потомства в генетическом алгоритме – это комбинация (скрещивание) существующих особей, то скрещивание таких сетей практически бесполезно.

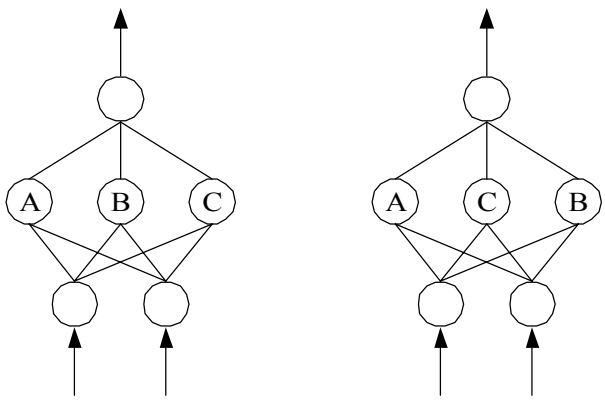


Рис.1.3. Проблема одинаковых конкурирующих решений

Например, если родительские особи представляют сети, изображенные на рис.1.3, причем веса связей у скрытых нейронов с одинаковыми метками равны, то получающиеся потомки будут, скорее всего, содержать в скрытом слое нейроны ABB и ACC с соответствующими им связями [32]. Было предложено довольно много вариантов разрешения этой проблемы, среди них стоит отметить:

- использование «умного» кроссовера, когда перед скрещиванием у родительских сетей ищутся скрытые нейроны с одинаковыми связями, и затем на основе полученной информации потомки создаются таким образом, чтобы они не содержали одинаковых скрытых нейронов [22, 28];
- использование исторических меток (historical markings), при этом все появляющиеся в ходе эволюции связи получают уникальный индекс, таким образом, если в двух сетях все межнейронные связи имеют одинаковые метки, то можно утверждать, что их топологии идентичны [32];

Корнинг в своей работе [17] предположил, что оценивание с использованием среднеквадратичной ошибки выхода сети является одной из причин проблемы одинаковых конкурирующих решений и предложил изменить критерий оценки и уничтожать всех особей, которые ему не удовлетворяют. Стоит отметить, что многие исследователи не считают проблему пермутаций большим препятствием эволюционному процессу [11, 14].

Один из довольно крупных недостатков нейроэволюционного подхода заключается в скудной теоретической базе, хотя попытки формализации метода в прошлом предпринимались [9, 3]. Как следствие, это не дает веских оснований «доверять» разработанному нейроэволюционному алгоритму, поскольку нет строгого доказательства или математического аппарата, позволяющего вывести это доказательство, что он успешно справляется, помимо уже рассмотренных, с какими-либо другими задачами. Большинство разработок и исследований в первую очередь ориентированы на практическое применение. Стоит отметить, что это общая проблема для генетических алгоритмов, однако здесь она стоит особенно остро.

Побочным эффектом этого недостатка является трудность в сравнении различных алгоритмов. Количество параметров, по которым они отличаются, может быть очень велико и их нельзя однозначно сопоставить. Например, существуют алгоритмы, ограничивающие число возможных скрытых нейронов и количество внутренних связей ИНС [23, 5] и, в то же время, разработаны намного более гибкие алгоритмы, позволяющие получать нейронные сети с практически любой топологией [9, 30]. Даже если признать, что последние алгоритмы потенциально более перспективны, то нельзя не отметить, что в них используются вычислительно более сложные механизмы, кроме этого алгоритмы с ограничениями на получаемые структуры ИНС успешно справляются с поставленными задачами [24, 7]. Часто для сравнения различных алгоритмов используется количество вычислений целевой функции для определенной задачи, т.е. сколько раз в сумме потребовалось оценить особей из популяции, чтобы получить решение. Преимуществом такой оценки является независимость от алгоритмов, однако, это же одновременно и недостаток, поскольку не учитывается их вычислительная сложность. Достаточно объективной оценкой могло бы быть время работы алгоритма в секундах, однако это требует, чтобы все разработчики программировали в одном стиле, использовали один и тот же компилятор и согласились проверять свои алгоритмы на компьютерах с абсолютно одинаковой конфигурацией. Хотя такая проверка может быть сделана одним человеком, но, учитывая количество существующих алгоритмов, это была бы очень сложная работа.

Обобщая, можно выделить следующие преимущества использования нейроэволюционных методов:

- большое разнообразие получаемых топологий – возможны решения с «нестандартными» структурами ИНС;
- адаптивность;
- универсальность;
- использование не требует наличия глубоких знаний об ИНС;

Недостатками и проблемами подхода являются:

- более высокие требования к объему оперативной памяти, чем у градиентных методов;
- недостаточно высокая эффективность при решении определенного круга задач;
- наличие проблемы одинаковых конкурирующих решений;
- скудная теоретическая база;
- требуются знания из области генетических алгоритмов;
- трудности в сравнении различных алгоритмов.

1.3.2. Способы кодирования информации

Для работы генетическому алгоритму необходимо представление информации о параметрах оптимизируемой системы, в данном случае нейронной сети, в виде строки (генотип). Разработано довольно много способов кодирования структуры и весов связей нейросети. Условно их можно разбить на два класса [2]:

- прямое (direct) кодирование – информация о нейронах и связях указывается в генотипе особи в явном виде;
- косвенное (indirect) кодирование. Здесь возможно два подкласса способов кодирования – с использованием грамматики представления топологических структур [21, 15, 9] и без, с использованием методов генетического программирования, кодирование отдельных блоков и другие [10].

Оба способа имеют свои преимущества и недостатки. Прямое кодирование позволяет более тонко настроить архитектуру сети, и получаемые решения, в общем случае, более компактные, чем при косвенном кодировании. С другой стороны при увеличении размеров сетей довольно существенно возрастает объем требуемой памяти. Для преодоления этого недостатка были разработаны способы косвенного кодирования, менее зависимые от размеров ИНС. В частности вместо отдельных нейронов выступали целые функциональные блоки, внутренняя структура которых неизменна, и настраиваются только связи между этими блоками. Это позволяет существенно снизить требования к объему памяти, при

этом теряется полнота в плане структур ИНС, которые можно представить подобным образом.

Однако и косвенные подходы изначально не были лишены недостатков, присущих прямому кодированию. В ранних работах (конец 80-х – начало 90-х годов прошлого столетия) архитектура сети описывалась в матричном виде [21, 15]. Элементы матрицы содержали символы разработанной грамматики. Также вводился набор правил, определяющий закон изменения размеров и содержимого матрицы. Недостатками матричного подхода является резкое возрастание необходимого объема оперативной памяти при увеличении размеров сети. К достоинствам следует отнести то, что такие системы достаточно легко поддаются формализации.

Рассмотрим более подробно варианты представления нейронных сетей в генотипе.

1) Кодирование связей. Возможно два варианта: использование матрицы смежности нейронов [19] (либо ее аналога с указанием весов на пересечении строк и столбцов) и перечисление связей [30].
2) Кодирование нейронов. При этом возможна запись данных о структуре ИНС в виде дерева [18]. Как правило, при кодировании информации о нейронах учитываются их положение в сети, входные связи и их вес, значение порога активационной функции и другая информация [16].
3) Кодирование слоев [39, 40, 44]. Используется для ИНС прямого распространения. В генотипе записываются данные о количестве слоев и числе нейронов в каждом из них. В случае, если два рядом находящихся слоя связаны не полностью, дополнительно кодируется информация об особенностях межслойных связей.
4) Кодирование маршрутов. Предложено в [13] для представления рекуррентных сетей. В генотипе записывается информация о возможных маршрутах от входных нейронов к выходным;
5) Клеточное (cellular) кодирование. Данный вариант предложен в [8]. Для записи информации о структуре ИНС используется специальная грамматика, описывающая образование нейронной сети как аналог процесса клеточного деления.

По иному пути пошли разработчики алгоритма SANE [23] и его модификаций [5, 26]. В генотипе особи кодируется информация об отдельном нейроне, причем суммарное количество входных и выходных связей у него фиксировано. Для конструирования нейронной сети случайно [23], либо по некоторому правилу [5, 24, 26] из популяции выбираются различные особи. Таким образом оцениваются разные комбинации особей, составляющие различные нейросети.

Начальная инициализация популяции может быть двух видов:

- случайная (неоднородная) – каждая особь представляет ИНС со случайной топологией, либо полностью или частично задана «вручную». Поиск необходимой сети ведется во всех «направлениях», как в сторону упрощения имеющихся структур, так и в сторону их усложнения;
- без скрытых нейронов – каждая особь представляет нейронную сеть, не имеющую скрытых нейронов, все входные нейроны соединены с выходными. При этом эволюционный поиск решения ведется преимущественно в направлении «роста» ИНС, т.е. увеличения числа скрытых нейронов и связей.

Очевидно, что для различных способов представления архитектуры ИНС требуются свои операторы скрещивания и мутации. Общая цель их разработки такова, чтобы обеспечить передачу информации от родительских особей потомкам. При этом, если в результате скрещивания создается одна особь, то она должна иметь топологические особенности обоих родителей, насколько это возможно. Если же потомков два, то каждый из них должен содержать признаки структуры хотя бы одной родительской особи. Т.е. не должна происходить потеря генетической информации, полученной в результате эволюции популяции.

1.3.3. Краткий обзор существующих алгоритмов

Из ранних работ заслуживает внимания клеточный алгоритм Фредерика Груо [8], использующий специальную грамматику для представления нейросетевых

структур. Одна особь представляла целую нейросеть, при этом каждый нейрон рассматривался как биологическая клетка, и рост сети определялся через механизмы последовательного и параллельного «деления» нейронов-клеток.

В алгоритме Hierarchical SANE (Symbiotic, Adaptive NeuroEvolution) [24] используется иной подход. Рассматривается развитие двух популяций, в одной из которых особи представляют собой отдельные нейроны, а в другой содержится информация о структурах ИНС. Количество скрытых нейронов и связей ограничено.

Алгоритм ESP является развитием алгоритма SANE [5]. Основным его отличием является то, что структура сети фиксирована, и задается априорно. Популяция нейронов разбивается на подпопуляции, в каждой из которых эволюция идет независимо. Благодаря распараллеливанию поиска решения, а также упрощению задачи за счет отказа от эволюции структуры ИНС, ESP быстрее, чем SANE, иногда на порядок.

Разработанный Кеннетом Стенли алгоритм NEAT (NeuroEvolution of Augmenting Topologies) позволяет настраивать структуру сети, причем без ограничений на ее сложность [32]. В алгоритме используется ряд приемов, такие, например, как исторические метки и специализация особей [4], позволяющих сделать процесс эволюции существенно более эффективным.

Глава 2. Алгоритм NEvA

2.1. Общее описание

В представленном алгоритме для поиска решения используется популяция нейронных сетей, т.е. каждая особь представляет отдельную ИНС. Во время инициализации популяции одна половина особей задается случайным образом. Гены второй половины популяции определяются как инверсия генов первой половины особей. Это позволяет равномерно распределить «1» и «0» биты в популяции, чтобы минимизировать вероятность ранней сходимости алгоритма.

После начальной инициализации в генах всех особей закодированы сети без скрытых нейронов, причем все входные нейроны соединены с каждым выходным нейроном. Т.е. изначально все представленные ИНС отличаются только весами межнейронных связей. В процессе оценивания, на основе генетической информации рассматриваемой особи сначала конструируется нейронная сеть, а затем проверяется ее работоспособность, которая определяет приспособленность данной особи. После оценивания все особи сортируются в порядке уменьшения приспособленности, и к скрещиванию допускается более успешная половина отсортированной популяции, причем лучшая особь сразу переходит в следующее поколение. В процессе репродукции каждая особь скрещивается со случайно выбранной особью из числа отобранных для скрещивания. Получившиеся два потомка добавляются в новое поколение. После того, как новое поколение сформировано начинает работать оператор мутации. Т.к. отбор усечением значительно ослабляет разнообразие внутри популяции [46] и ведет к ранней сходимости алгоритма, то вероятность мутации выбирается довольно большая, порядка 15–25%.

В случае, если лучшая особь в популяции не изменяется на протяжении более чем 7 поколений, происходит перезапуск алгоритма. Во время перезапуска вся популяция инициализируется заново, и процесс поиска решения начинается «с нуля». Таким образом осуществляется выход из локальных минимумов, обусловленных рельефом целевой функции, а также большой степенью сходства

особей в одном поколении.

В отличии от алгоритмов, представленных в [23, 5] количество скрытых нейронов теоретически неограниченно. Для регуляции размеров получаемых сетей используются два коэффициента, позволяющих на этапе мутации адаптивно выбирать, какой тип преобразования структуры больше подходит для данной сети.

2.2. Кодирование информации

Существует несколько способов кодирования информации о нейронной сети в генотипе особи [16], из которых наиболее распространенными являются следующие варианты:

- кодирование связей – в генотипе представлена информация о весах межнейронных связей;
- кодирование нейронов – используется информация о каждом нейроне сети, его положении в сети, параметрах и входящих связях;
- кодирование информации о слоях – особь содержит данные о числе слоев и количестве нейронов в каждом слое;
- косвенное кодирование (indirect encoding) – кодируются не отдельные элементы ИНС, а целые участки с фиксированной топологией. Структура сети представляется как комбинация таких участков. Также возможно использование специальной грамматики для представления ИНС в генотипе.

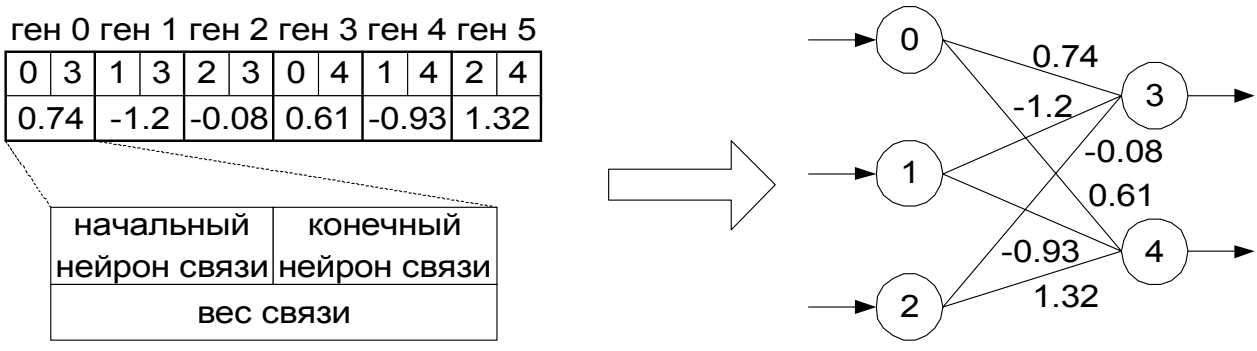


Рис.2.1. Кодирование информации о сети

Разработанный алгоритм использует первый вариант кодирования. При этом каждый ген содержит информацию об индексах начального и конечного нейрона связи, а также её вес. Пример представления ИНС в генотипе особи и соответствующая структура изображены на рис.2.1.

Все нейроны сети индексируются в соответствии с изложенными ниже правилами:

- поскольку число входов и выходов сети – величина фиксированная, индексы соответствующих нейронов постоянны и принимают значения [0; N_I-1] для входных нейронов, и [N_I; N_I+N_O-1] для выходных, где N_I и N_O – число входов и выходов сети соответственно. Удаление входных и выходных нейронов невозможно;
- новые нейроны, появляющиеся в результате мутаций получают минимальный возможный индекс. Например, если особь представляет сеть с двумя входами, двумя выходами и не содержит скрытых нейронов, то новому нейрону в этой сети будет присвоен индекс «4», следующему появившемуся – «5» и т.д.;
- индексы нейронов в сети, представленной особью, не могут содержать «пробелы», то есть не может быть ИНС с нейронами с индексами 0, 1, 2, 3, 4, 6, 7. Даже если такой случай возникает, например, после удаления из сети нейрона с индексом «5», индексы оставшихся нейронов корректируются следующим образом: «6» → «5», «7» → «6», при этом меняются данные в связях, относящихся к данным нейронам.

2.3. Генетические операторы

Очевидно, что выбранный метод кодирования требует специальных генетических операторов, реализующих скрещивание и мутацию.

При скрещивании используются две родительские особи, которые производят двух потомков. Общие нейроны и связи наследуются обоими

потомками, при этом значения связей в сетях потомков формируются с помощью двухточечного кроссовера, Различающиеся элементы ИНС «разыгрываются» между потомками. Пример скрещивания показан на рис.2.2. Сплошными линиями показаны общие нейроны и связи, пунктирными – различающиеся.

Важной особенностью является то, что нейроны с одинаковыми индексами считаются идентичными, несмотря на различное количество связей и положение в сети, а также то, что один из таких нейронов мог иметь другой индекс, который изменился в результате коррекции индексов после мутации.

Мутация может быть нескольких видов:

- добавление скрытого нейрона с присвоением индекса [N-1], где N – количество скрытых нейронов в сети после добавления. Новый нейрон добавляется вместе с входящей и выходящей связями. При этом выходная связь нейрона не может идти на входной нейрон;
- удаление случайно выбранного скрытого нейрона вместе со всеми входными и выходными связями. При этом если образуется «пробел» в индексах оставшихся нейронов, то производится коррекция индексов в соответствии с приведенным выше алгоритмом. Входные и выходные нейроны сети не могут быть удалены;
- добавление связи. Случайным образом определяются индексы начального и конечного нейронов в ИНС, представленной мутирующей особью. При этом связь не может заканчиваться входным нейроном. Вес связи определяется также случайно из диапазона [-0,5; 0,5]. Если в сети уже существует связь с такими же входными и выходными нейронами, то ее вес заменяется на случайный;
- удаление случайно выбранной связи. При этом может возникнуть ситуация, когда удаляется последняя связь у скрытого нейрона. В этом случае нейрон также удаляется, и, если необходимо, производится коррекция индексов нейронов сети;
- изменение веса случайно выбранной связи на случайную величину из диапазона [-0,5; 0,5].

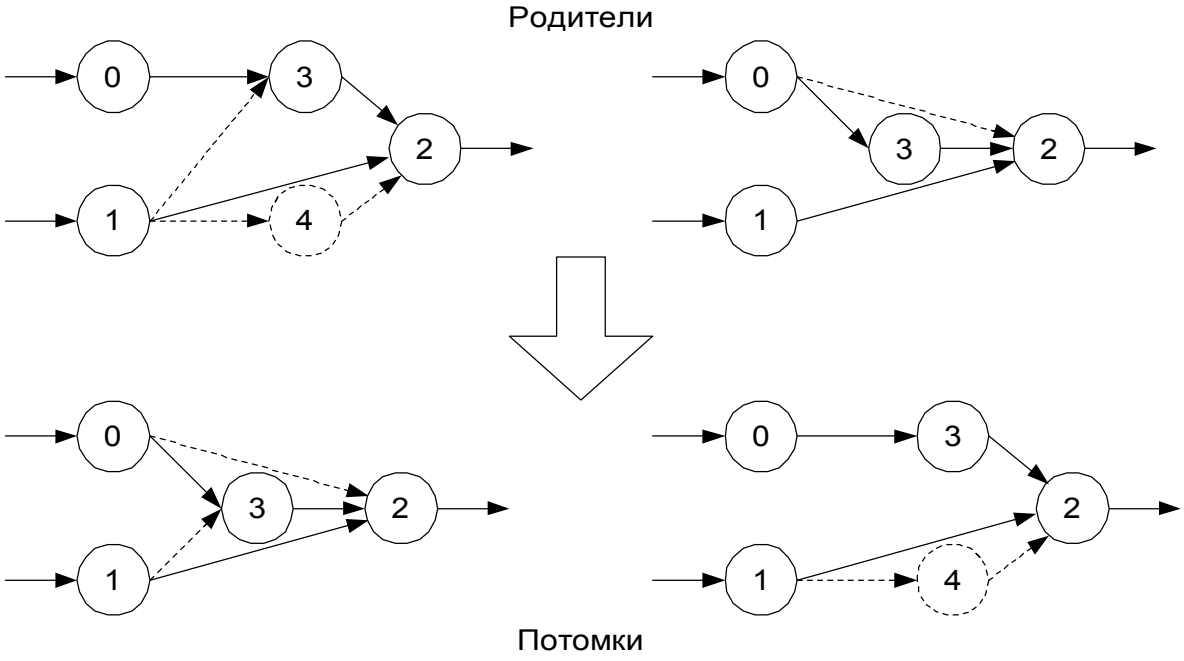


Рис.2.2. Скрещивание

Таким образом, с помощью мутации можно «точечно» изменять параметры структуры ИНС.

Беспорядочное добавление (удаление) нейронов и связей может привести к ситуациям, когда, например, в сети много нейронов и мало связей. Более логичным будет применять различные виды мутаций в зависимости от особенностей архитектуры сети, представленной мутирующей особью. Для этого были введены два коэффициента, регулирующие размер и «направление развития» сети.

Первый из них характеризует степень «связанности» нейронов сети и вычисляется по формуле:

$$f_C = \frac{N_C}{2^{FB-1}\left[N_N\left(N_N-1\right)-N_I\left(N_I-1\right)-\left(1-FB\right)N_O\left(N_O-1\right)\right]}, \qquad (2.1)$$

здесь и далее N_C – количество связей в сети, N_I, N_O, N_N – соответственно количество входных, выходных нейронов и общее число нейронов в сети, FB – флаг, обозначающий, разрешено появление обратных связей (FB=1) или нет (FB=0). Стоит отметить, что связи от скрытых нейронов к выходным могут появляться в любом случае. Таким образом, чем меньше f_C, тем больше с большей вероятностью в результате мутации будет добавлена новая связь. Возможно 4

варианта использования f_C:

- вычисленное по формуле (2.1);
- возведенное в квадрат;
- умноженное на некоторый коэффициент;
- возведенное в квадрат и умноженное на коэффициент.

В дальнейшем будем обозначать f_C, преобразованное одним из обозначенных выше способов, как F_C.

Использование второго коэффициента основано на предположении, что чем больше элементов в сумме во входном и выходном векторах обучающей выборки (чем больше суммарное количество входных и выходных нейронов), тем, вероятно, более сложная сеть необходима для решения поставленной задачи. Вычисляется второй коэффициент по следующей формуле:

$$f_N = \frac{N_I + N_O}{N_N}, \tag{2.2}$$

обозначения такие же, как и для формулы (2.1). То есть, чем больше будет нейронов в сети, тем меньше будет f_N, и с тем меньшей вероятностью будет выбрана мутация, добавляющая новый скрытый нейрон. Так же, как и для f_C, возможно 4 варианта использования f_N (обозначается как F_N):

- вычисленное по формуле (2.2);
- возведенное в квадрат;
- умноженное на некоторый коэффициент;
- возведенное в квадрат и умноженное на коэффициент.

Для любого из описанных случаев в алгоритме используется связка F_N*F_C, т.к. для использования F_N необходимо учитывать степень связанности уже существующих нейронов.

Удаление связей в ИНС дает побочный эффект: могут появляться «висячие» нейроны, у которых нет входящих связей, а также «тупиковые» нейроны, т.е. без выходящих связей. В случаях, когда функция активации нейронов такова, что при нулевой взвешенной сумме входов ее значение не равно нулю (например, лог-сигмоидная функция), наличие «висячих» нейронов дает возможность

настраивать нейронные смещения. Стоит отметить, что, с другой стороны, удаление связей может способствовать удалению части неинформативных и малоинформативных входных признаков, что будет продемонстрировано в следующей главе настоящего отчета при решении задачи о перевернутом маятнике.

2.4. Выбор вида мутации

Сам оператор мутации был подробно описан в предыдущем разделе. Здесь будет рассмотрена зависимость вида мутации от величин f_C и f_N. Адаптивный механизм мутации является одной из ключевых особенностей предлагаемого алгоритма.

Выбор вида мутации определяется исходя из значений F_C и F_C*F_N. Такой подход с одной стороны не ограничивает «сверху» количество скрытых нейронов, с другой – препятствует безмерному увеличению сети, т.к. добавление каждого нового нейрона в сеть будет происходить с меньшей вероятностью. Мутация веса случайной существующей связи происходит для всех мутирующих особей с вероятностью 0,5.

Рассмотрим более подробно, каким образом выбирается вид мутации. На рис. 2.3 представлена блок-схема выбора типа мутации. Здесь СВ – случайная величина, N_H – количество скрытых нейронов в мутирующей сети. Для краткости выбор варианта вычисления F_C и F_N, а также мутация веса случайно выбранной связи на схеме не указаны.

Условно весь алгоритм можно разделить на две «ветки» по первому условному переходу:

1) Ветка увеличения f_C, выполняется по выполнению условия перехода.
2) Ветка уменьшения f_C, выполняется при невыполнении условия перехода.

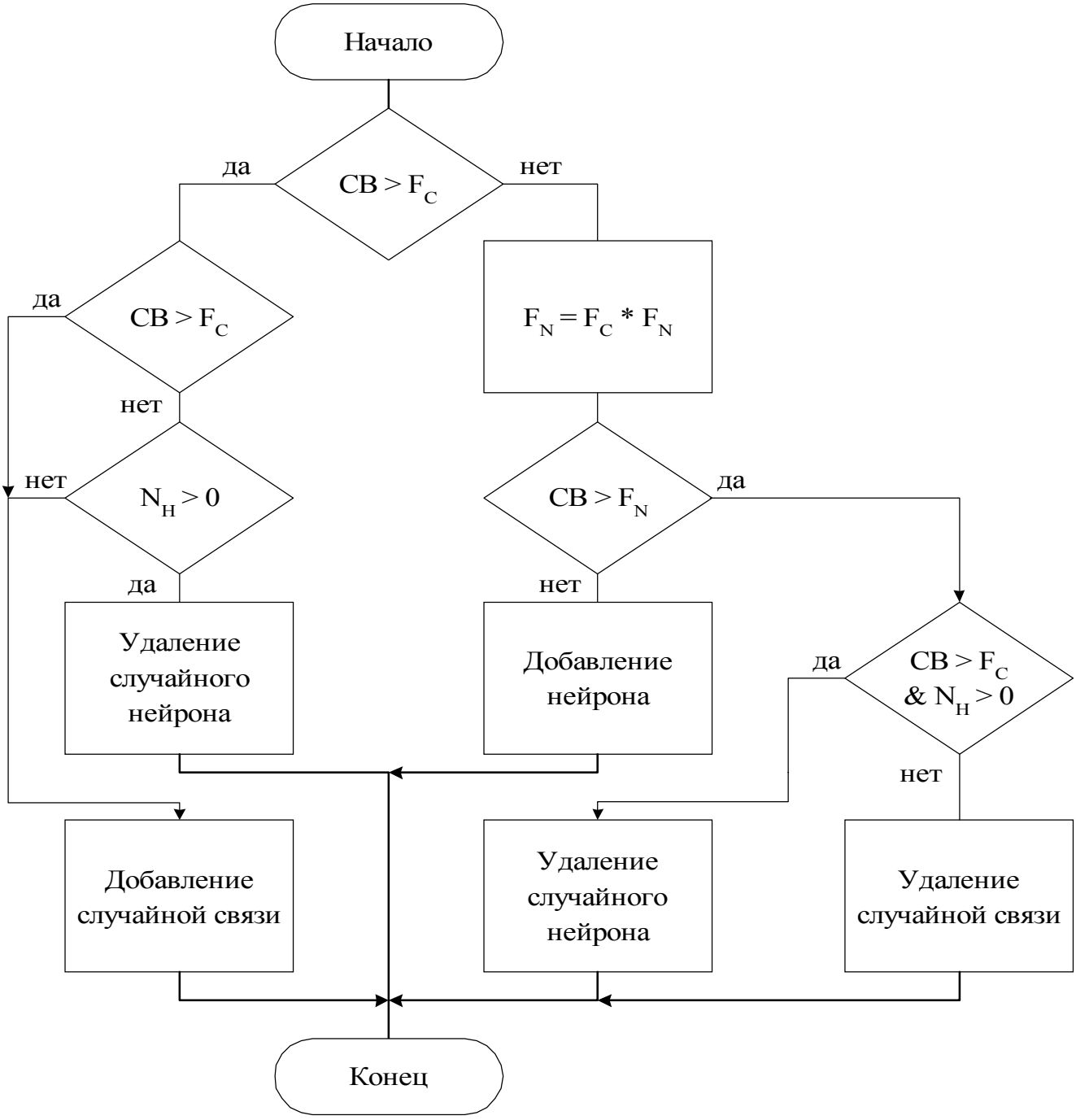


Рис. 2.3. Блок-схема выбора вида мутации

Поскольку удаление нейрона может привести как к уменьшению, так и к увеличению f_C, в зависимости от количества связей нейрона, то этот вариант присутствует в обоих ветвях. Таким образом, основным фактором для регуляции структуры получаемых ИНС является степень их «связанности». Другими словами, насколько полно реализуются возможные связи сети.

Умножение F_N на F_C необходимо для того, чтобы изменять количество

нейронов адекватно топологии сети, т.к. при добавлении (удалении) нейронов необходима информация о целесообразности изменений. Эта информация может быть косвенно получена, исходя из значения характеристики f_C.

2.5. Алгоритм расчета выхода ИНС

Одной из проблем реализации нейроэволюционного алгоритма является алгоритм расчета выхода ИНС с произвольной топологией.

Нейронная сеть может быть представлена, как направленный плоский граф. Исходя из того, что структура сети может быть любой, в графе допускаются петли и циклы, содержащие любые узлы, за исключением узлов соответствующих входным нейронам. Обозначим множество узлов графа через $V=\{v_i \mid i\in[0; N_V-1]\}$, а множество дуг через $E=\{e_j \mid j\in[0; N_E-1]\}$, где N_V и N_E– соответственно количество узлов и дуг в графе, причем $N_V\equiv N_N$, а $N_E\equiv N_C$. Дугу идущую от узла k к узлу l обозначим упорядоченной парой $e_{k,l}=(v_k, v_l)$, вес соответствующей связи будем обозначать через $w_{k,l}$.

Проиндексируем узлы графа также как и нейроны, т.е. узлы, соответствующие входным нейронам, назовем их входными. имеют индексы из диапазона $[0; N_I-1]$. По аналогии индексы выходных узлов принадлежат промежутку $[N_I; N_I+N_O-1]$, а индексы для скрытых узлов будут задаваться в интервале $[N_I+N_O; N_V-1]$.

Введем дополнительную характеристику для всех узлов графа, равную минимальной длине цепи до любого из входных узлов и обозначим ее l_i. Назовем l_i слоем, которому принадлежит i-й узел. Таким образом, все входные узлы принадлежат 0-му слою, все не входные узлы, имеющие входящие дуги от входных принадлежат 1-му слою, все остальные узлы со входящими дугами от узлов 1-го слоя будут принадлежать слою с индексом 2 и т.д. При этом возможны ситуации, когда узел не имеет входных дуг, будем называть его висячим узлом с номером слоя $l_i=-1$.

Для дуг также введем дополнительную характеристику $b_{k,l}$ для дуги $e_{k,l}$, необходимую для определения, соответствует дуга прямой связи или обратной. Будем вычислять ее следующим образом:

$$b_{k,l} = \begin{cases} 1, l_l - l_k > 0, \\ -1, l_l - l_k \le 0. \end{cases} \quad (2.3)$$

Т.е. если индекс слоя конечного узла дуги больше, чем индекс слоя начального узла, то будем считать такую дугу прямой, в противном случае, будем считать дугу обратной.

Т.к. каждый узел графа представляет нейрон, то обозначим через sum_i значение взвешенной суммы входов, а через o_i – значение выхода (величина активационной функции) i-го нейрона-узла. Тогда $o_i = f(sum_i)$, где f – активационная функция нейрона.

Разобьем весь процесс распространения сигналов от входных узлов на этапы, причем во время одного такого этапа сигналы «успевают» пройти только по одной дуге. Номер этапа обозначим через s. Для самого первого этапа s=1. Для простоты будем считать, что все дуги имеют одинаковую длину, а сигналы распространяются по ним мгновенно. Обозначим признак того, что выход узла i обновился на данном этапе через a_i, т.е. если $a_i \equiv 1$, то выход узла на этапе s вычислен, в противном случае, если $a_i \equiv 0$ – нет.

Введем еще одно обозначение $X=\{x_i \mid i \in [0; N_I-1]\}$ – вектор входных сигналов. Тогда алгоритм расчета выхода ИНС выглядит следующим образом:

1. $o_i=x_i$, $a_i=1$, для всех $i \in [0; N_I-1]$;
2. $o_i=0$, для всех $i \in [N_I; N_N-1]$;
3. $s=1$;
4. $sum_i=0$, $a_i=0$, для всех $i \in [N_I; N_N-1]$;
5. если $s \equiv 1$, то перейти на шаг 7;
6. учет всех обратных связей сети. Для всех входящих обратных дуг $e_{j,k}$ узла v_k, где $k \in [N_I; N_N-1]$: $sum_k = sum_k + o_j$, если $l_j < s$;
7. если $a_i \equiv 0$, то fn(i), для всех $i \in [N_I; N_N-1]$;
8. если не выполнен критерий останова, то $s = s+1$, переход на шаг 4.

Здесь fn(i) – рекурсивная функция, вычисляющая выход i-го узла с учетом всех прямых дуг. Работает по следующему алгоритму:

1. если $l_i<0$, то переход на 3;
2. для всех прямых входящих дуг $e_{k,i}$ узла v_i: если $a_k \equiv 1$, то $sum_i = sum_i + o_k$, иначе fn(k);
3. $o_i = f(sum_i)$;
4. выход.

Критерий останова алгоритма расчета выхода ИНС может быть одним из следующих:

- стабилизация значений на выходе ИНС;
- s превышает установленное значение.

Более надежным представляется расчет выхода до того момента, когда значения на выходе ИНС не изменяются, однако для случая, когда сеть содержит циклы и/или петли, ее выход может никогда не стать стабильным. Поэтому необходим дополнительный критерий останова, ограничивающий допустимое число этапов вычисления выхода сети. Для сетей без обратных связей (FB=0) во многих случаях достаточно разрешить $\max(l_i)+1$ этапов.

Глава 3. Результаты работы алгоритма

Для проверки работоспособности нейроэволюционных алгоритмов и оценивания в первом приближении их эффективности используют несколько «де факто» эталонных задач:

- «Исключающее ИЛИ».
- Перевернутый маятник (inverted pendulum) [34].
- Преследование жертвы (Prey capture) [20].

Оценивание каждой особи происходит в два этапа. Сначала на основе информации из генотипа конструируется соответствующая ИНС, затем проверяется, насколько хорошо она справляется с поставленной задачей, и численная характеристика работы нейросети отражает приспособленность рассматриваемой особи.

3.1. Исключающее ИЛИ

Данную задачу можно назвать классической. Применительно к разработанному алгоритму она заключается в том, чтобы сконструировать нейронную сеть, способную реализовать логическую операцию «Исключающее ИЛИ». Таблица истинности приведена в табл.2.

Табл.3.1. Таблица истинности для операции «Исключающее ИЛИ»

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Данная проблема является частным случаем задачи n-битной четности, когда для входной строки разрядностью n бит требуется установить на выходе

нейросети «1», если количество единичных разрядов в ней нечетно, и «0», если количество единичных разрядов четно. Встречаются варианты использования задачи 3- и 4-битной четности для тестирования нейроэволюционных систем [25, 37].

Для сравнения данная задача была решена с использованием следующих алгоритмов:

- алгоритм обратного распространения ошибки без инерционной составляющей;
- алгоритм обратного распространения ошибки с инерционной составляющей;
- канонический генетический алгоритм.

Первые два алгоритма являются градиентными, а последний использует эволюционный принцип для поиска решения. Перечисленные алгоритмы настраивали только веса сети, структура которой была фиксированной и включала 3 входных нейрона, 4 скрытых и 1 выходной. Иными словами использованные сети содержали 8 нейронов и 17 связей. Все алгоритмы запускались по 50 раз, и решение считалось найденным, если появлялась особь, представляющая сеть, ошибка выхода которой (значение целевой функции) была меньше 0,001. Если количество вычислений целевой функции превышало 150000, то работа алгоритма останавливалась, и результат считался неудачным.

Алгоритмы сравнивались по тому, какое количество вычислений ошибки выхода (N) требовалось для нахождения решения. Результаты работы представлены в табл.5.

Табл. 3.2. Результаты работы алгоритмов

N	NE	CGA	BP	BPM
0 – 499	0	12	0	24
500 – 999	0	10	0	20
1000 – 1999	0	3	0	2
2000 – 4999	21	4	46	2
5000 – 9999	26	4	1	2

10000 – 14999	3	1	1	0
15000 – 24999	0	4	1	0
25000 – 49999	0	5	0	0
50000 – 150000	0	5	1	0
>150000	0	2	0	0

В ячейках указано, сколько раз решение было найдено в соответствии с потребовавшимся количеством вычислений функции ошибки сети (N).

Средние значения количества вычислений целевой функции (ЦФ) по 50 запускам рассматриваемых алгоритмов приведены в табл. 3.3.

Табл. 3.3. Средние результаты

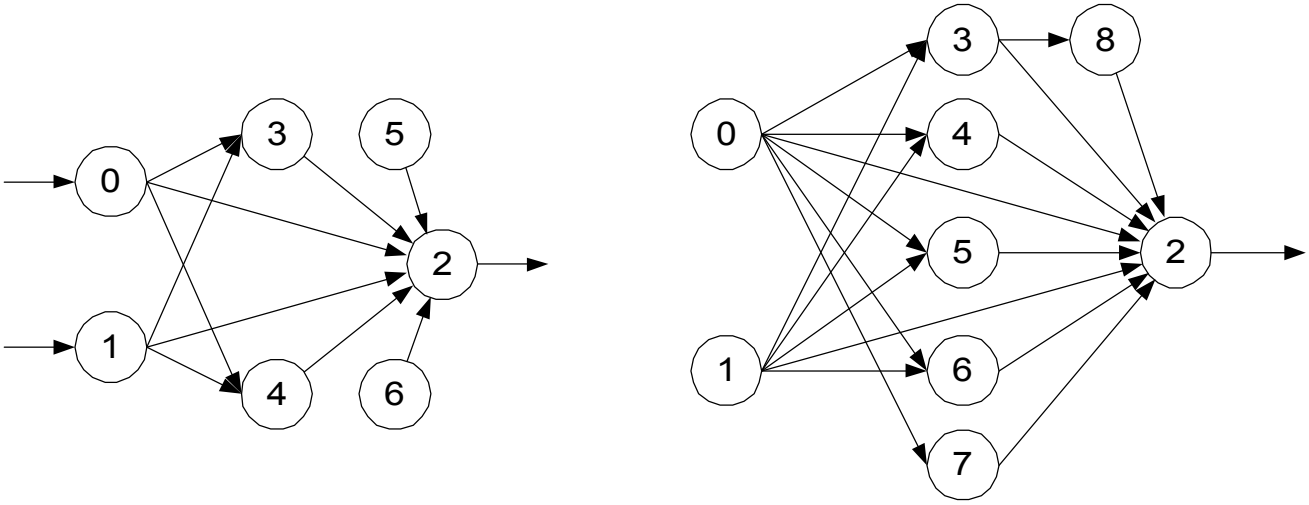
	NE	CGA	BP	BPM
Среднее количество вычислений ЦФ	6026,26	13165,63	5338,28	828,32

По полученным результатам можно сделать следующие выводы:

- среднее число скрытых нейронов в решениях, найденных разработанным алгоритмом, равно 4.8, связей – 16.22;
- 28 раз были найдены сети с более простой топологией, чем топология сети, использовавшейся для настройки алгоритмами CGA, BP и BPM, и 4 раза получены сети с эквивалентной сложностью;
- несмотря на проигрыш по скорости использованным градиентным алгоритмам, стоит отметить превосходство над каноническим ГА, а также стабильность работы, способствовавшую решению задачи с использованием меньше 15000 вычислений целевой функции, при изначально большей сложность задачи;

Примеры решений, полученных с помощью нейроэволюционного алгоритма, изображены на рис.3.1. Напомним, что изначально все особи представляли сети без скрытых нейронов (т.е. содержали только нейроны с индексами 0, 1 и 2). Также напомним, что остальные алгоритмы настраивали сеть с 8 нейронами и 17

связями. Обращает внимание наличие «висящих нейронов» у «удачного» решения. Использовались нейроны с лог-сигмоидной функцией активации, особенность которой заключается в том, что при нулевом аргументе значение функции равно 0,5. Таким образом, нейроны 5 и 6 играют роль нейронных смещений для выходного нейрона (с индексом 2).



7 нейронов, 10 связей 9 нейронов, 18 связей

Рис.3.1. Примеры найденных решений

Минимальная сеть, способная решать поставленную задачу, состоит из 4 нейронов и 5 связей [2].

3.2. Перевернутый маятник

Задача относится к области адаптивного управления. Существует несколько разновидностей этой проблемы. В самой простой имеется тележка, на которой установлен шест. В начальный момент времени он отклоняется от положения равновесия на случайную величину, и задача заключается в том, чтобы, перемещая тележку вперед и назад с определенным фиксированным усилием, удержать шест. В более сложном варианте воздействие на тележку может непрерывно изменяться в некоторых пределах (как правило [-10N; 10N]).

Поскольку описанные варианты являются довольно простыми проблемами, было предложено использовать два шеста разной длины [34], а также задачу в двухмерной среде, когда тележка может перемещаться по двум независимым направлениям [6]. Математическая модель системы приведена в [5].

Были использованы следующие варианты задачи:

1) один инвертированный маятник, сила воздействия на тележку фиксированная и равна –10Н или 10Н. Эксперименты проводились для сетей с одним выходом с лог-сигмоидной функцией активации.
2) два инвертированных маятника с полной информацией об их движении (не требует поиска сети с обратными связями).

Каждый алгоритм запускался по 50 раз. Результаты экспериментов в зависимости от параметров приведены в табл.3.4. Данные для алгоритмов GENITOR, SANE и ESP взяты из [5]. СКО – среднеквадратичное отклонение результатов работы алгоритмов.

Табл.3.4. Результаты решения проблемы с 1 маятником

Алгоритм	Количество попыток				Число неудач
	Среднее	Лучшее	Худшее	СКО	
GENITOR	1846	272	7052	1396	0
SANE	535	70	1910	329	0
ESP	285	11	1326	277	0
NEvA	451	37	3872	720	0

По полученным результатам разработанный алгоритм проигрывает только алгоритму ESP. Стоит отметить, что этот алгоритм не настраивает структуру ИНС. Информация о структуре использованных для решения сетей приведены в табл. 3.5. Данные для алгоритмов GENITOR, SANE, ESP приведены в [5, 47]. “+/-” для алгоритма SANE означает, что количество скрытых нейронов и связей фиксировано, но алгоритм настраивает их расположение.

Табл.3.5. Данные о сетях для проблемы с 1 маятником

Алгоритм	Среднее число скрытых нейронов	Среднее число связей	Настройка структуры сети
GENITOR	5	35	-
SANE	8	40	+/-
ESP	5	30	-
NEvA	0.58	5.22	+

Результаты решения задачи с двумя маятниками и информация об использованных сетях представлены соответственно в табл. 3.6 и 3.7 Результаты для эволюционного программирования взяты из работы [29], для алгоритмов SANE, ESP из [5] для алгоритма NEAT из публикации [31]. Скорость работы алгоритма NEvA сопоставима с результатами алгоритма NEAT. При этом полученные сети имеют меньшее количество скрытых нейронов: от 0 до 2 против от 0 до 4 у NEAT [31]. Количество связей для NEAT дано предположительно и посчитано на основе информации о числе скрытых нейронов.

Табл.3.6. Результаты решения проблемы с 2 маятниками

Алгоритм	Среднее количество попыток	Размер популяции	Число неудач
Эволюционное программирование	307200	2048	0
SANE	12600	200	0
ESP	3800	200	0
NEAT	3578	150	0
NEvA	3777	59	0

Табл.3.7. Данные о сетях для проблемы с 2 маятниками

Алгоритм	Среднее число скрытых нейронов	Среднее число связей	Число неудач
ESP	5	35	0
NEAT	0 – 4	6 – 15	0
NEvA	0 – 3 (1,4)	4 – 16 (7.24)	0

Одна из особенностей разработанного алгоритма заключается в том, что в результате возможности удаления связей мутацией можно выделять неинформативные входы. Например, для успешного балансирования маятников необязательно иметь информацию о скорости передвижения тележки или о ее координатах. В результате возможны нейросетевые решения с 1 изолированным входом из 2, на которые подается информация о тележке. Такие решения были найдены:

- в 2 случаях для балансирования одного маятника для сетей с одним выходом;
- в 8 случаях для задачи с двумя маятниками, при этом 1 раз была получена сеть с 2-мя изолированными входами с информацией о тележке.

3.3. Преследование жертвы

Данная проблема имеет отношение к адаптивному управлению и поведению. В некоторой среде находятся хищник и жертва. Задача заключается в том, чтобы, управляя хищником, поймать жертву, которая убегает в соответствии с алгоритмом, описанным в [20]. Предложено несколько вариантов решения, при этом хищник либо изначально пытается догнать убегающую жертву, либо постепенно обучается от простых задач (поймать неподвижную жертву) к более

сложным (поймать жертву, скорость которой увеличивается при каждой тренировке) [5]. Последний вариант получил название эволюция с усложнением (Incremental evolution) [33]. Математическая модель движения жертвы приведена в [20].

Для решения задачи был использован многоагентный подход. Ниже приводится описание разработанной системы.

Имеется популяция из N агентов $A = \{A_i \mid i \in [1;N]\}$. Каждый агент представляет ИНС, заданную перечислением всех её связей. Нейроны сети индексируются в зависимости от занимаемого ими места. Входные нейроны получают индексы от 0 до (N_I-1), где N_I – число входов сети, выходные – от N_I до (N_I+N_O-1), где N_O – число выходов, скрытые нейроны – от (N_I+N_O) до (N_N-1), где N_N – суммарное число нейронов в сети. Связь описывается следующим образом: $c = \{(b, e, w) \mid b \in [0;N_I-1] \cup [N_I+N_O;N_N-1], e \in [N_I;N_N-1], w \in [-65{,}536;65{,}535]\}$, здесь b и e – соответственно индексы начального и конечного нейронов связи, w – вес связи. Таким образом, каждый агент может быть определен как $A_i = \{c_{ij} \mid j \in [1;N_{C,i}]\}$, где $N_{C,i}$ – число связей в ИНС i-го агента.

Также есть система S, обладающая набором параметров. Общая цель МАС – управление системой S таким образом, чтобы она достаточно эффективно выполняла свои функции. Действие i-го агента на систему описывается в соответствии с формулой $S_{k'} = C(A_i, S_k, E_k)$, где S_k и $S_{k'}$ – соответственно состояния системы S до управления и после него, E_k – состояние среды до управления, C – функция, реализующая управление системой под руководством агента A_i. Сама система S находится в некоторой изменчивой среде E, также имеющей ряд характеристик, которые могут изменяться как самопроизвольно, так и под влиянием изменений в S. Опишем изменение среды $E_{k'} = V(E_k, S)$, где $E_{k'}$ и E_k – соответственно состояния среды до и после изменения, V – функция, определяющая закон взаимодействия системы S со средой, а также правила изменчивости самой среды.

При оценивании популяции каждый агент имеет возможность управлять системой, причем после завершения его работы система возвращается в

состояние, предшествовавшее управлению. В результате каждому агенту ставится в соответствие число, характеризующее его успешность (приспособленность), т.е. существует функция $f(C(A_i, S(t)), E(t)) = f_i$, оценивающая работу i-го агента, где S(t) и E(t) – соответственно состояния системы и среды в текущем поколении. Затем агенты сортируются в порядке ухудшения этой характеристики. Самый лучший агент изменяет параметры управляемой системы и переходит в новое поколение без изменений. Все остальные агенты замещаются потомками, получившимися в результате скрещивания «родителей» из текущего поколения. При этом к скрещиванию допускается только те агенты, чья характеристика лучше средней, т.е. если A' – агенты отобранные для скрещивания и f_{cp} – средняя величина приспособленности, то $A' = \{A_i \mid f_i > f_{cp}\}$. При скрещивании происходит рекомбинация топологий ИНС, представленных агентами, в результате чего есть возможность получить новые структурные решения. Это также достигается с помощью механизма мутации потомков, «точечно» изменяющего параметры ИНС.

Таким образом, $A(t+1) = A_0(t) \cup R(A'(t))$, где t – номер поколения, A(t+1) – популяция на (t+1) поколении, $A_0(t)$ – лучшая особь в поколении t, R – функция, генерирующая подпопуляцию потомков, полученных после применения к A'(t) операторов кроссовера (скрещивание) и мутации. Изменение системы: $S(t+1) = C(A_0(t), S(t))$, где S(t) и S(t+1) – соответственно состояния S в поколениях t и (t+1). Аналогично для среды $E(t+1) = V(E(t), S(t+1))$, т.е. среда реагирует на действия системы S, управляемой $A_0(t)$. Стоит отметить, что в явном виде обратная связь в поведении отсутствует, поскольку выбор следующего действия системы S не зависит от предыдущего, т.е. можно говорить о том, что рассматривается система без памяти.

Жертва действует на основании алгоритма, который является модификацией алгоритма из [20], описанного во второй главе. Опишем алгоритм выбора направления движения жертвы. Его отличие от классического варианта заключается в том, что после вычисления $P(A_i)$ выбираются две величины с наибольшим значением, которые принимаются за длины векторов направленных в соответствующие стороны. Их векторная сумма и определит направление

движения жертвы. Если только 1 вероятность события A_i больше, чем остальные (все остальные равны между собой, либо 2-е и 3-е места разделились), то именно это направление и станет определяющим.

Скорость жертвы постоянна и равна 0,475. Управление движением хищника осуществлялось с использованием информации о векторе «хищник→жертва». Управляющими параметрами являлись угол, на который должен повернуться хищник и расстояние, которое он должен пройти в заданном направлении в диапазоне [0; 1]. Каждый управляющий агент представляет ИНС с 2 входами и 2 выходами, при этом топология сети у каждого агента может быть различной. Заметим, что управление хищником осуществлялось на основе неполной информации, т.к. на входы нейронных сетей агентов не подавались данные о текущем направлении взгляда хищника.

Ставилось два различных вида экспериментов. В первом жертва неподвижна, а во втором жертва двигалась в соответствии с разработанным алгоритмом. В обоих случаях хищник должен подойти на расстояние не больше 0,5 единиц. В самом начале эксперимента хищник находится в начале координат, а жертва в случайной точке в пределах определенного радиуса r (были использованы значения 10 и 25). В каждом запуске алгоритма производилось 250 испытаний МАС с различными стартовыми позициями, причем после успешного завершения одного испытания популяция агентов, не изменяясь, подвергалась следующему испытанию. В качестве критерия оценки запуска для случая неподвижной жертвы было взято отношение

$$\eta_1 = \frac{Length_{SUM}}{Steps_{SUM}}, \tag{3.9}$$

где $Length_{SUM}$ – суммарное расстояние, которое *робот* прошел за 250 испытаний, $Steps_{SUM}$ – потраченное для этого количество шагов. Т.е. оценивается отношение фактического расстояния, пройденного хищником к максимальному теоретическому. Для случая подвижной жертвы добавилась ещё одна оценка

$$\eta_2 = \frac{\Delta Length_{SUM}}{Length_{0,SUM}}, \tag{3.10}$$

где $Length_{0,SUM}$ – суммарное начальное расстояние от хищника до жертвы, $\Delta Length_{SUM}$ – суммарное расстояние, которое жертва успела пройти от своей начальной позиции. Второй критерий оценивает, насколько далеко ушла жертва в зависимости от своего первоначального удаления от хищника. Для каждого случая осуществлялось 10 запусков, результаты усреднялись.

Значения критерия η_1 для неподвижной жертвы при различных значениях размера популяции (|A|) и радиуса удаления (r) приведены в табл.3.8. В последних двух столбцах указаны значения при предварительном обучении (pre-training) популяции перед каждым испытанием в течение 5 поколений.

Табл.3.8.

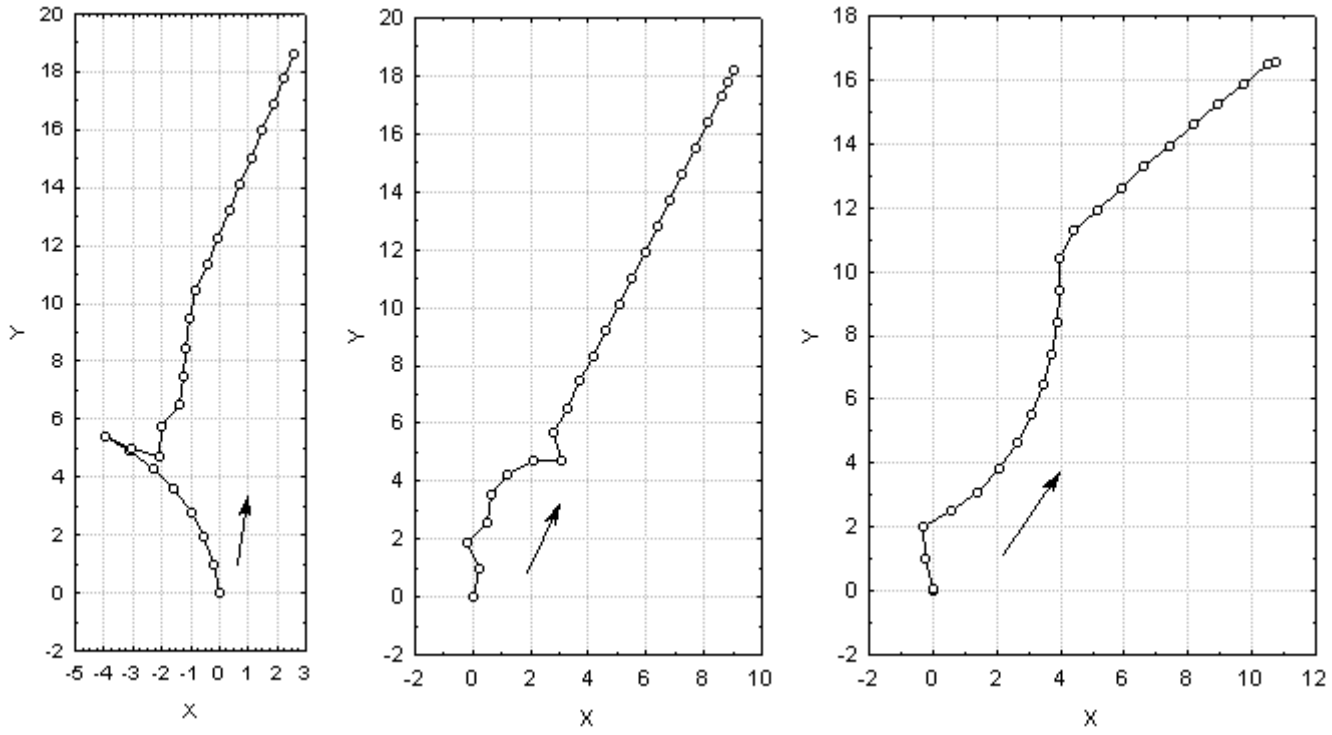
	r = 10	r = 25	r = 10 (pre)	r = 25 (pre)
|A| = 25	0,6902	0,7609	0,7115	0,7792
|A| = 50	0,8310	0,9031	0,8559	0,9032

Примеры траекторий движения хищника до неподвижной жертвы представлены на рис. 3.2. Стрелками отмечены направления движения. Результаты для случая подвижной жертвы представлены в табл.3.9 (r = 25).

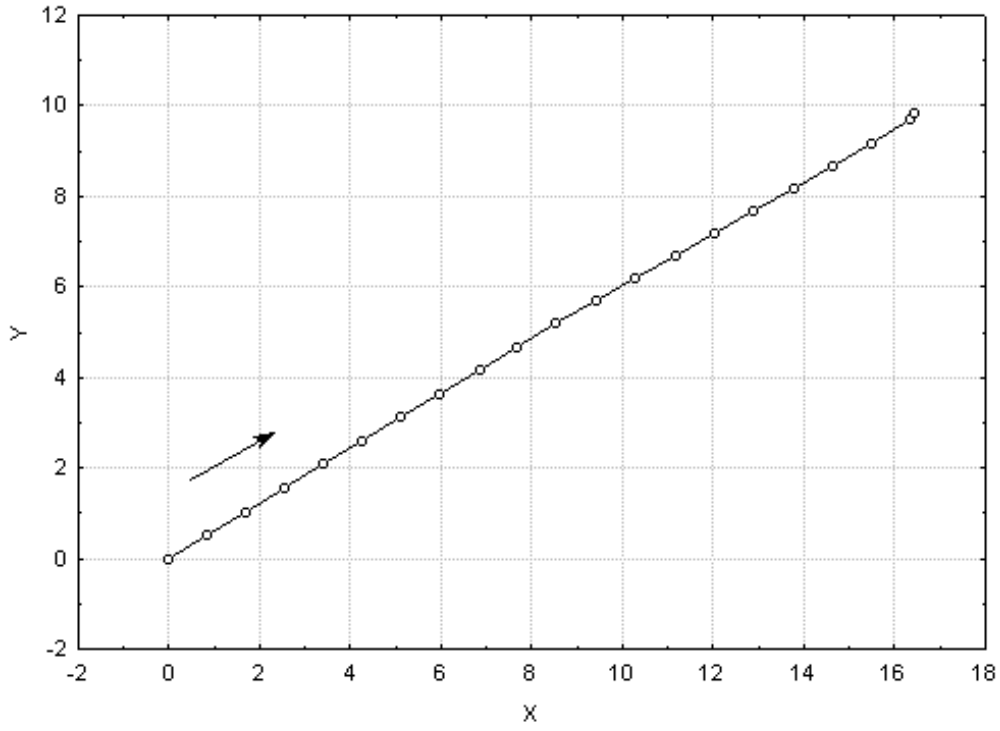
Табл.3.9.

	η_1	η_2
|A| = 25	0,5164751	7,291877
|A| = 50	0.5741533	3,699684

Из данных в табл.3.9 видно, что использование в два раза большей популяции агентов способствовало в два раза более успешной поимке жертвы. Примеры траекторий движения хищника (обозначено маленькими кругами) и жертвы (обозначено ромбами) представлены на рис. 3.3. Места поимки жертвы обозначены большими кругами.



а) б) в)



г)

Рис.3.2. Примеры траекторий движения хищника:

а) r=25, |A|=25; б) r=25, |A|=50;

в) r=25, |A|=25, pre-training; г) r=25, |A|=50, pre-training.

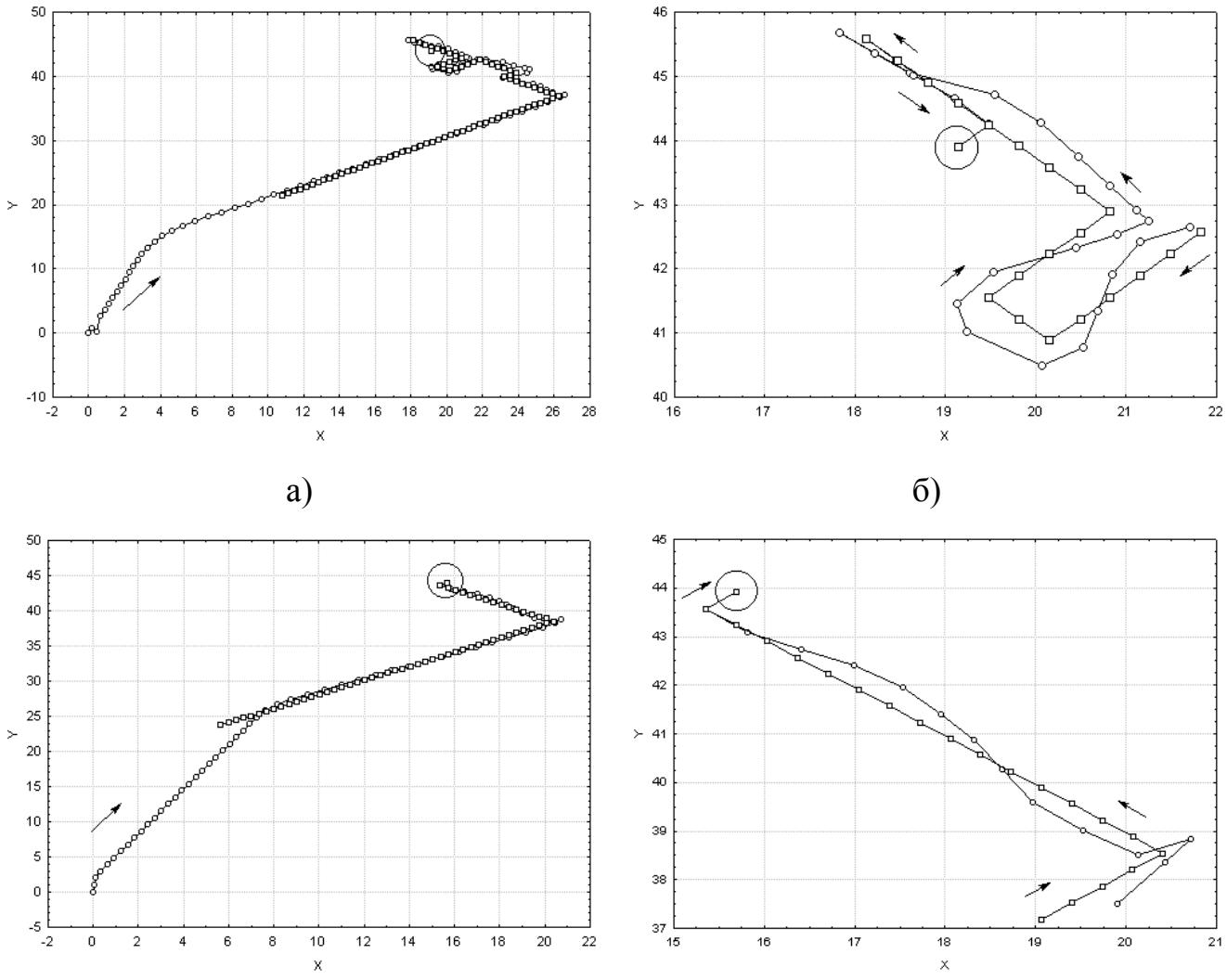


Рис.3.3. Примеры траекторий движения хищника и жертвы:
а) |A|=25; б) |A|=25 увеличено; в) |A|=50; г) |A|=50 увеличено.

При увеличении траекторий движения для случая с подвижной жертвой можно заметить одну интересную особенность, наличие памяти, о чем можно судить по инерционности в движении хищника. При проектировании предлагаемой МАС память не закладывалась, однако она реализовалась за счет наследственности, которая присутствует при формировании нового поколения агентов.

Число случаев, когда количество скрытых нейронов в полученных решениях превышало 1, составляет ~10%.

Предложенная задача является по большому счету «синтетической» и не

требует сложных алгоритмов для решения, однако она вполне подходит, чтобы показать в действии разработанную МАС. Результаты показали, что в работе МАС присутствуют как достоинства (адаптивность, сравнительно простые топологии ИНС), так и недостатки (бóльший, чем для других вариантов необходимый объем оперативной памяти).

Глава 4. Исследование структур получаемых ИНС

В главе 2 при описании предлагаемого алгоритма были введены две характеристики f_N и f_C, а также зависящие от них параметры F_N и F_C, регулирующие процесс поиска топологий ИНС. В данной главе будет рассмотрено влияние этих величин на структуры получаемых нейронных сетей.

4.1. Постановка задачи и условия экспериментов

Обе величины f_C и f_N уменьшают вероятность добавления каждого нового нейрона в сети в следующих случаях:

- количество скрытых нейронов существенно больше, чем количество входных и выходных нейронов;
- количество связей в сети мало по сравнению с максимально возможным.

Влияние количества связей объясняется тем, что при добавлении каждого следующего нейрона максимальное количество связей в сети увеличивается на все большую величину, уменьшая тем самым вероятность появления в результате мутации нового нейрона.

В силу описанных выше обстоятельств представляют интерес максимально сложные по структуре нейронные сети, которые могут быть получены при использовании различных вариантов F_C и F_N.

Будем использовать следующие комбинации:

1) Original – оба параметра равны f_C и f_N, вычисленные соответственно по формулам (3) и (4);
2) Square – оба параметра получены возведением f_C и f_N в квадрат;
3) Multiply – величины F_C и F_N равны f_C и f_N, умноженным на 0,5;
4) SquareMultiply – комбинация вариантов 2 и 3, когда используются f_C и f_N, возведенные в квадрат, а затем умноженные на 0,5.

Очевидно, что существуют и другие варианты, однако мы остановимся на выбранных, поскольку они позволяют акцентировать внимание на различных

способах вычисления F_C и F_N. Для комбинаций 3 и 4 множитель 0,5 был выбран с тем, чтобы изучить каким образом уменьшение в два раза вероятностей добавления нейронов и связей повлияет на сложность получаемых сетей.

Установим вероятность мутации равную 1, чтобы исключить зависимость от случайности в появлении данного оператора, в результате которого и возможно появление новых связей и нейронов.

Приспособленность каждой особи вычислялась по формуле:

$$f_i = N_H^2 + N_C^2, \tag{4.1}$$

где N_H – количество скрытых нейронов в сети особи i, N_C – количество связей в сети i-й особи.

При проведении экспериментов особи инициализировались сетями со следующей топологией:

- 1 входной, 1 выходной нейрон, 1 связь (1i-1o);
- 10 входных, 1 выходной нейрон, 10 связей (10i-1o);
- 100 входных, 1 выходной нейрон, 100 связей (100i-1o).

Эксперименты проводились для популяций размером 10, 100 и 1000 особей. Алгоритм запускался 1 раз для каждого варианта настройки параметров, поэтому анализу в первую очередь будут подвергаться возможные общие тенденции и закономерности, а уже потом конкретные числовые результаты. При повторном запуске, в силу присутствия элемента случайности в работе алгоритма, возможны иные значения анализируемых параметров.

4.2. Результаты

Результаты для различных сетей с популяциями разного размера представлены на рис. 4.1–4.4. Графики показывают зависимость среднего количества связей в сетях от количества прошедших поколений. Буквами “O”, “S”, “M” и “SM” обозначены соответственно случаи для комбинаций Original, Square, Multiply, SquareMultiply параметров F_C и F_N. N обозначает размер

популяции. Для популяций из 100 особей и сетей с 10 входами и 1 выходом (10i-1o) графики зависимостей подобны приведенным.

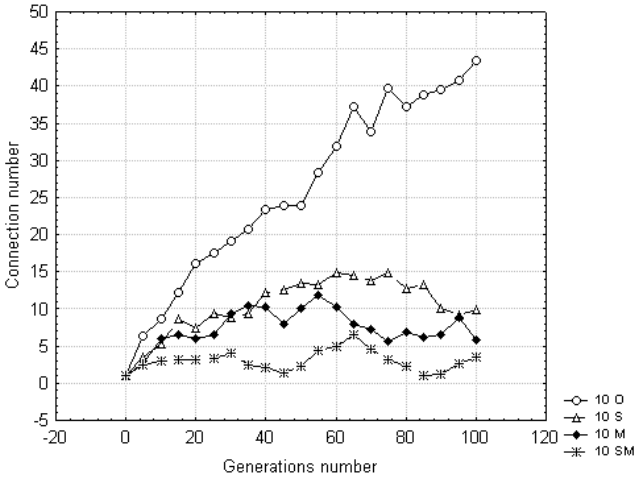


Рис. 4.1. Результаты для сети 1i-1o (N=10)

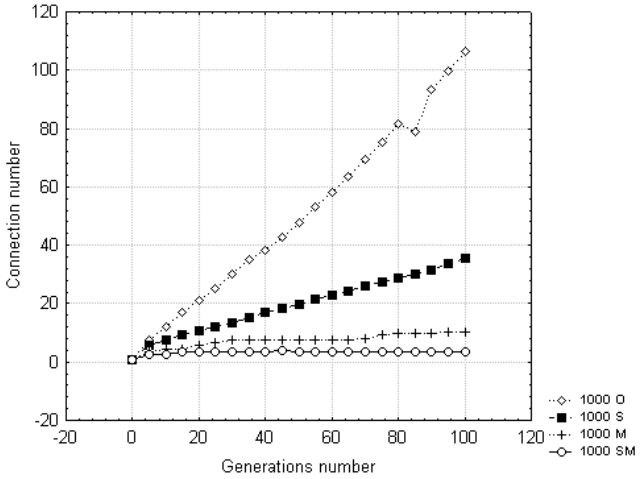


Рис. 4.2. Результаты для сети 1i-1o (N=1000)

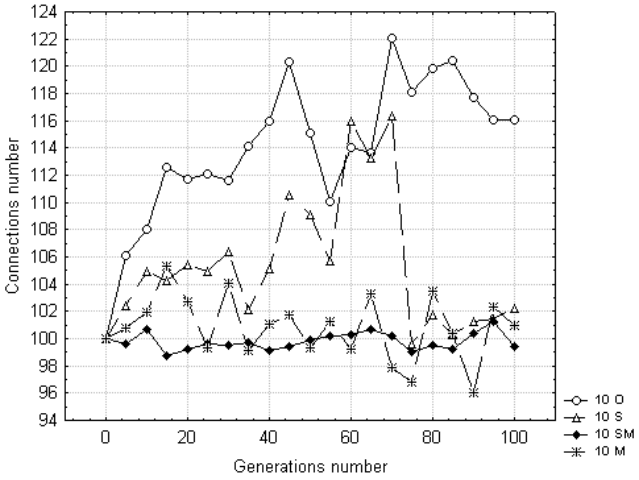


Рис. 4.3. Результаты для сети 100i-1o (N=10)

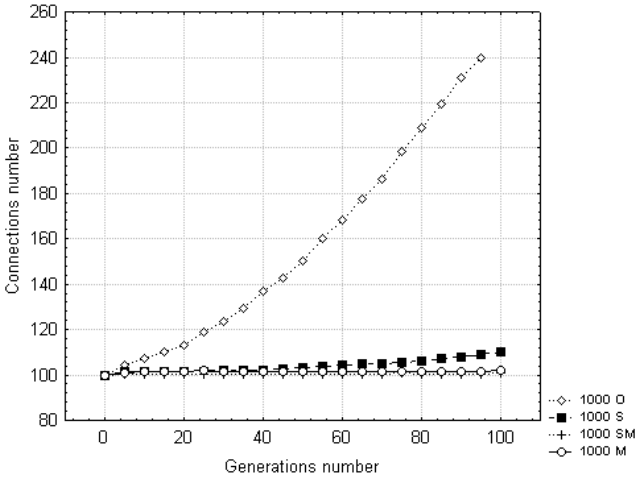


Рис. 4.4. Результаты для сети 100i-1o (N=1000)

Можно отметить следующие закономерности:

1) Увеличение размера популяции способствует более интенсивному «росту» сети.
2) При использовании комбинаций O и S наблюдается постепенное, близкое к линейному, увеличение размеров сети. Для комбинации O темпы увеличения числа связей больше.
3) Для комбинаций M и SM характерны медленный рост размеров сети с дальнейшей стабилизацией среднего количества скрытых связей около некоторого значения. При этом среднее число связей для комбинации M,

как правило, больше.

Графики зависимостей числа связей при фиксированной комбинации параметров F_C и F_N и переменном размере популяции приведены на рис. 4.5–4.8. Рассмотрены случаи для сетей со 100 входами и 1 выходом.

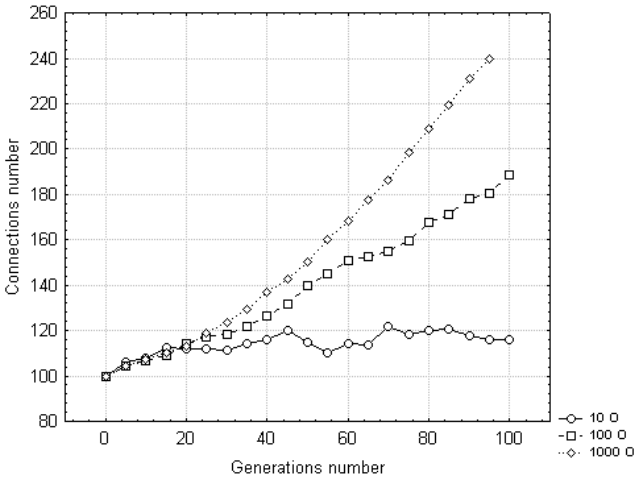


Рис. 4.5. Результаты для комбинации Original

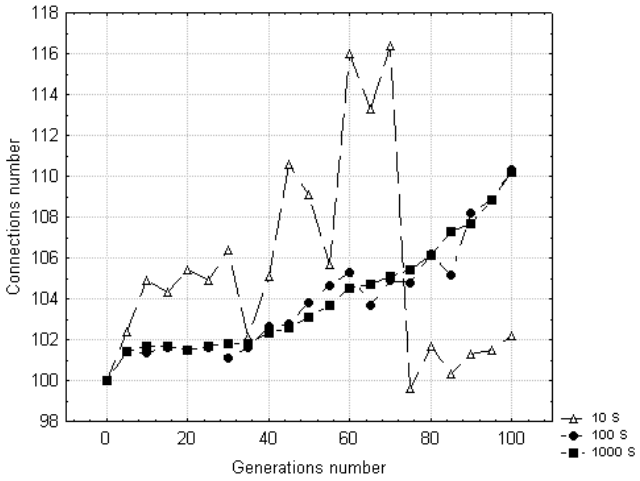


Рис. 4.6. Результаты для комбинации Square

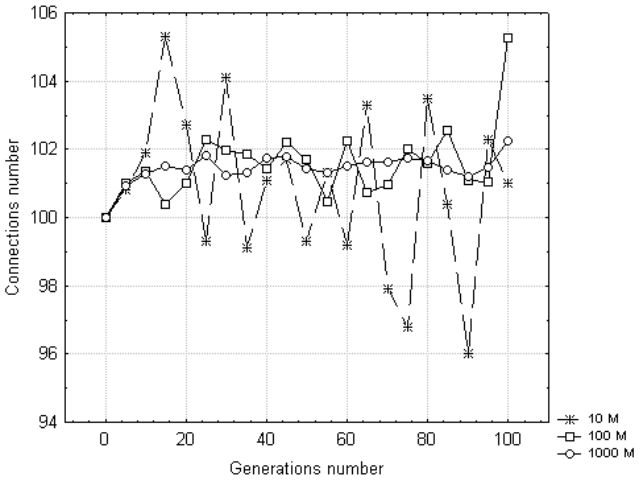


Рис. 4.7. Результаты для комбинации Multiply

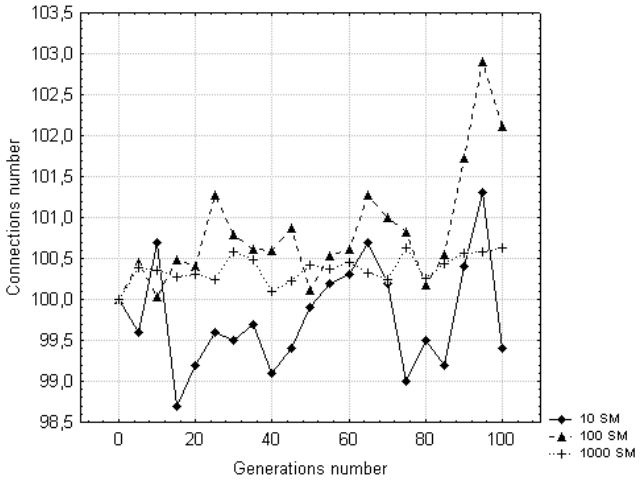


Рис. 4.8. Результаты для комбинации SquareMultiply

Из приведенных рисунков видно, что для малых популяций (N=10) характерна некоторая хаотичность изменения числа связей. Данная особенность наиболее сильно выражена для комбинаций S и M. Перепады, но уже не настолько сильные, также присутствуют и случаев с использованием комбинаций O и SM. Также можно более явно наблюдать стабилизацию числа связей для комбинаций M и SM.

Начиная с некоторого значения, при увеличении размера популяции скорость

роста среднего числа связей в сетях для комбинации O и S растет незначительно. Данное свойство можно наблюдать при сравнении среднего количества связей в популяциях из 100 и 1000 особей.

4.3. Выводы

По полученным результатам можно сделать следующие выводы:

1) В большинстве случаев предпочтительнее использовать комбинацию Square, когда характеристики f_C и f_N возводятся в квадрат. В этом случае наблюдается усложнение топологии сети, необходимое для решения задач классификации. С другой стороны темпы роста сложности сети не настолько велики, как для комбинации Original, что позволяет получать более простые и компактные решения. Рассмотренные в 3 главе задачи были решены с использованием комбинации Square.
2) Комбинация Original может быть использована для случаев, когда априори известно, что для решения необходима сложная сеть. Это могут быть, например, случаи классификации с разделяющими поверхностями высоких порядков.
3) Комбинации M и SM могут использоваться для решения задач, в которых известно, что топология сети может получиться простой. Кроме этого использование рассматриваемых комбинации могут способствовать более эффективному удалению неинформативных и малоинформативных параметров в задачах классификации.

В дальнейшем планируется исследовать зависимость скорости нахождения решения алгоритмом от различных комбинаций использования параметров F_C и F_N.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был произведен анализ современных методов искусственного интеллекта, включающих генетические алгоритмы и искусственные нейронные сети. Рассмотрены теоретические аспекты и созданы программные реализации обеих концепций.

Сделан обзор существующих нейроэволюционных алгоритмов. Их анализ позволил создать новый алгоритм, названный NEvA (NeuroEvolutionary Algorithm), способный одновременно настраивать веса и структуру искусственной нейронной сети. Ключевыми особенностями алгоритма являются:

- адаптивность мутационных изменений топологии ИНС, что позволяет получить сбалансированные с точки зрения структуры нейронные сети;
- удаление малоинформативных входных параметров задачи.

Проведен ряд экспериментов, показавших удовлетворительные и хорошие результаты работы созданного алгоритма для задач классификации, адаптивного управления и поведения.

В настоящее время проводится научно-исследовательская работа, посвященная более детальному исследованию свойств созданного алгоритма, решению ряда существующих проблем и выделения путей усовершенствования. В частности, наиболее значимыми и интересными для автора являются:

- увеличение скорости нахождения решений;
- возможности удаления неинформативных и малоинформативных признаков во время обучения ИНС;
- улучшение адаптивных свойств алгоритма, таких как изменяемый размер популяции.

Планируется создать программную реализацию с графическим интерфейсом пользователя для демонстрации ряда преимуществ нейроэволюционного подхода на базе разработанного алгоритма.

CONCLUSION

As a result of accomplished work an analysis of modern methods of artificial intelligence, namely artificial neural networks and genetic algorithms, was made. Some theoretical aspects were investigated and software implementations of both concepts were created.

Also a review was made concerning existing neuroevolutionary methods. Their analysis allowed to devise a new algorithm, called NEvA (NeuroEvolutionary Algorithm), that makes possible simultaneous design and training of neural networks. The key features of created algorithm are:

- adaptive mutation changes in network topology, that allows getting structure balanced neural networks;
- lesser- and non- informative task input parameters deletion.

A set of experiments showed satisfactory and good performance of created algorithm for classification, adaptive control and behavior tasks.

At the time additional researches take place for more thorough examination of NEvA properties and for solution of some existing problems and search of further improvements and development. Particularly, the most significant and interesting points are:

- performance increase;
- abilities of lesser- and non- informative inputs deletion during network design and training;
- improvements of adaptive abilities of algorithm, such as dynamical population sizing.

It is planned to make a program with graphical user interface on base of elaborated algorithm to show some advantages of neuroevolutionary approach.

СПИСОК ИСТОЧНИКОВ

1. Alander, J.T. An indexed bibliography of genetic algorithms: Years 1957–1993 (Report Series No. 94–1). Finland: University of Vaasa, 1994.
2. Balakrishan K., Honavar V. Evolutionary Design of Neural Architectures – A Preliminary Taxonomy and Guide to Literature. Iowa State University, CS TR #95-01, 1995.
3. Balakrishan K., Honavar V. Properties of Genetic Representation of Neural Architectures. Iowa State University, 1995.
4. Goldberg D.E., Richardson J. Genetic algorithms with sharing for multimodal function optimization. // Proceedings of the Second International Conference on Genetic Algorithms, Morgan Kaufmann. – San Francisco, California, 1987. – P. 148-154.
5. Gomez F., Miikkulainen R. Incremental evolution of complex general behavior. // Adaptive Behavior. –1997. – №5. – P. 317–342.
6. Gomez F., Miikkulainen R. 2-D Pole Balancing Benchmark with Recurrent Evolutionary Networks. // Proceedings of the International Conference on Artificial Neural Networks (ICANN–98, Skovde, Sweden). – New York, Elsevier, 1998.
7. Gomez F. J., Miikkulainen R. Active Guidance for a Finless Rocket using Neuroevolution. // Proceedings of the Genetic Evolutionary Computation Conference (GECCO). – 2003.
8. Gruau F. Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. // Combination of Genetic Algorithms and Neural Networks, IEEE Computer Society Press, 1992.
9. Gruau F. Neural Networks Synthesis using Cellular Encoding and the Genetic Algorithm. PhD thesis, Ecole Normale Superieure de Lyon, 1994.
10. Gruau F. Automatic Definition of Modular Neural Networks. // Adaptive Behavior. – 1995. – №3(2). – P. 151–183.
11. Hancock P.J.B. Genetic algorithms and permutation problems: a comparison of recombination operators and neural structure specification. // Combinations of Genetic Algorithms and Neural Networks. – IEEE Computer Society Press, 1992.

12. Holland J. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, 1975.
13. Jacob C., Rehder J. Evolution of neural networks architectures by a hierarchical grammar based genetic system. //International Joint Conference on Neural Networks and Genetic Algorithms. – Innsbruck, 1993. – P. 72–79.
14. Karunanithi N., Das R., Whitley D. Genetic cascade learning for neural networks. //International Workshop on combination of Genetic Algorithms and Neural Networks. – Baltimore, 1992. – P. 134–145.
15. Kitano H. Empirical studies on the speed of convergence of neural network training using genetic algorithms. // Proceedings AAAI. – 1990. – P. 789–795.
16. Koehn P. Genetic Encoding Strategies for Neural Networks. – University of Tennessee – Universität Erlangen-Nürnberg, Erlangen, 1996.
17. Korning P.G. Training of neural networks by means of genetic algorithm working on very long chromosomes. Technical Report. – Computer Science Department, Aarhus C, Denmark, 1994.
18. Koza J. R., Rice J. P. Genetic generation of both the weight and architecture for a neural network. // International Joint Conference on Neural Networks. – 1991. – Vol. 2, P. 397–404.
19. Law D., Miikkulainen R. Grounding Robotic Control with Genetic Neural Networks. Technical Report AI94–223. – The University of Texas at Austin, 1994.
20. Lin L.-J. Self-improving reactive agents based on reinforcement learning, planning, and teaching. // Machine Learning. – 1992 – №8 – P. 293–321.
21. Mjolness E., Sharp D.H., Alpert B.K. Scaling, machine learning, and genetic neural nets. // Advances in Applied Mathematics. – 1989 – №10 – P. 137–163.
22. Montana D.J., Davis L. Training feedforward neural networks using genetic algorithms. // Proceedings of the 11-th International Joint Conference on Artificial Intelligence. – Morgan Kaufmann, San Francisco, California, 1989. – P. 762–767.
23. Moriarty D. E., Miikkulainen R. Efficient reinforcement learning through symbiotic evolution. // Machine Learning. –1996. – № 22. – P. 11–32.
24. Moriarty D. E., Miikkulainen R. Forming Neural Networks through Efficient and Adaptive Coevolution. // Evolutionary Computation. – 1998. – № 5(4).

25. Pan Z., Kang. L., Nie S. Evolving Both the Topology and Weights of Neural Networks. // Parallel Algorithms and Applications. – 1996. – Vol.9, P. 299–307.
26. Polani D., Miikkulainen R. Eugenic Neuro-Evolution for Reinforcement Learning. // Proceedings of the Genetic and Evolutionary Computation Conference (GECCO–2000). –Las Vegas, NV, 2000.
27. Radcliffe N.J. Genetic Neural Networks on MIMD computers. Doctoral dissertation. – University of Edinburgh, Edinburgh, England, 1990.
28. Radcliffe N.J. Genetic set recombination and its application to neural network topology optimization. Technical Report EPCC–TR–91–21. – University of Edinburgh, Edinburgh, Scotland, 1991.
29. Saravanan N., Fogel D. B. Evolving neural control systems. // IEEE Expert. – 1995. – P. 23-27.
30. Stanley K. O., Miikkulainen R. Evolving Neural Network through Augmenting Topologies. Technical Report TR–AI–01–290. – The University of Texas at Austin, 2001.
31. Stanley K. O., Miikkulainen R. Efficient Reinforcement Learning through Evolving Neural Network Topologies. // Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002). – San Francisco, CA, Morgan Kaufmann, 2002.
32. Stanley K. O., Miikkulainen R. Evolving Neural Topologies through Augmenting Topologies. // Evolutionary Computation. –The MIT Press, 2002. – № 10(2). – P. 99-127.
33. Wieland A. Evolving controls for unstable systems. // Proceedings of the 1990 Connectionist Model Summer School. – San Mateo, CA: Morgan Kaufmann, 1990. – 91–102.
34. Wieland A. Evolving neural network controllers for unstable systems. // Proceedings of the International Joint Conference on Neural Networks. –IEEE Press, Piscataway, New Jersey, 1991. – P. 667–673.
35. Whitley D. Genetic Algorithms and Neural Networks. // Genetic Algorithms in Engineering and Computer Science. –John Wiley, 1995. – P. 203-216.
36. Whitley D. A Genetic Algorithm Tutorial. // Statistics and Computing. – 1994. –

№4. – P. 65-85.

37. Yang J., Honavar V. Feature subset selection using genetic algorithm. // Genetic Programming 97. – San Francisco, Morgan Kaufmann Publishers, 1997.
38. Chern H. Y., Miikkulainen R. Cooperative Coevolution of Multi-Agent Systems. Technical Report AI01–287. – The University of Texas at Austin, 2001.
39. Harp S.A., Samad T., Guha A. Towards the genetic synthesis of neural networks. // Third international conference on genetic algorithms. –San Mateo, CA: Morgan Kaufmann, 1989. – P. 360–369.
40. Божич В.И., Лебедев О.Б., Шницер Ю.Л. Разработка генетического алгоритма обучения нейронных сетей // Перспективные информационные технологии и интеллектуальные системы. – 2001. – №1. – С. 21-24.
41. Каллан Р. Основные концепции нейронных сетей.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 288 с.: ил. – Парал. тит. англ.
42. Короткий С. Нейронные сети: основные положения// http://www.neuropower.de/rus/books/.
43. Круглов В.В., Борисов В.В. Искусственные нейронные сети: Теория и практика. – 2-е изд. стереотип. – М.: Горячая линия-Телеком, 2002. – 382 с.: ил.
44. Паклин Н. Обучаем нейронную сеть генетическим алгоритмом. – 2003. http://paklin.newmail.ru/mater/gene_net.html
45. Спицын В.Г. Базы знаний и экспертные системы: Учеб. пособие. – Томск: Изд. ТПУ, 2001г. – 88 с.
46. Blickle T., Thiele L. A comparison of Selection Schemes Used in Genetic Algorithms. Technical Report No. 11. – Zurich, Switzerland, Swiss Federal Institute of Technology, 1995.
47. Moriarty D.E., Miikkulainen R. Efficient Reinforcement Learning through Symbiotic Evolution. // Machine Learning. – 1996. – № 22. – P. 11-33.

ПРИЛОЖЕНИЕ 1. Возможности применения алгоритма NEvA

Одним из основных направлений использования искусственных нейронных сетей являются задачи классификации. При этом, как правило, в качестве исходных данных используется некоторая выборка, содержащая совокупность уже известных данных и распределение по классам. Основная задача заключается в том, чтобы обучить ИНС классификации исходных данных с целью использования полученной сети с новой информацией. Более того, обученная нейронная сеть может быть использована в дальнейшем для анализа данных, в качестве модели некоторого процесса или явления.

При использовании предлагаемого алгоритма задача облегчается тем, что от пользователя не требуется определять структуру ИНС. Достаточно иметь обучающую выборку, с использованием которой будет произведен эволюционный поиск готовой к эксплуатации нейронной сети. При этом поиск решения осуществляется не только среди хорошо изученных многослойных топологий, но и для «нестандартных» сетей, использующих связи между несмежными слоями, а также между нейронами одного слоя. Это позволяет более полно использовать потенциал ИНС и путем меньших затрат решить ту или иную задачу классификации (А.И. Галушкин, 2000). Таким образом, возможно упрощение задачи внедрения и эффективного использования нейросетевых технологий.

Кроме этого возможно использование алгоритма в автоматизированных системах сбора и анализа данных, благодаря тому, что алгоритм способен самостоятельно находить решение.

Также представляет интерес использование нейронных сетей в многоагентных системах, которые являются одним из наиболее перспективных направлений исследования современного искусственного интеллекта. Одна из сложностей заключается в механизме обучения и взаимодействия агентов, принимающих решения на основе выхода ИНС. Эволюционный подход включает механизмы, позволяющий обойти эту проблему «естественным» путем. Т.е. за счет эволюционного отбора и скрещивания становится возможна эволюция агентов, и, следовательно, развитие МАС. В третьей главе настоящего отчета приведен пример

реализации такой многоагентной системы. При этом, также благодаря особенностям генетических алгоритмов, в созданной МАС наблюдается наличие памяти, а, значит, присутствует развитие с учетом предыдущих действий и событий.

ПРИЛОЖЕНИЕ 2. Программа EasyNet

Основной задачей при разработке данной программы является демонстрация возможностей предлагаемого алгоритма при решении задач классификации и построении нейросетевых моделей.

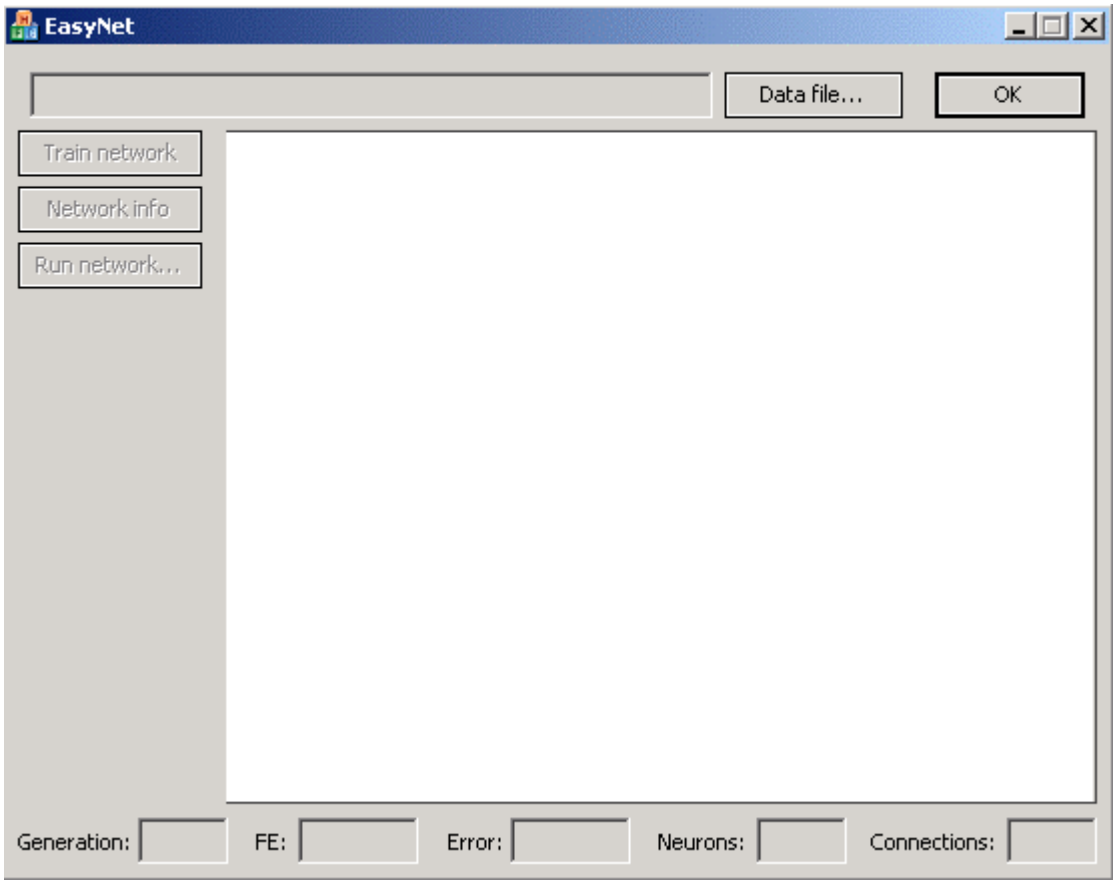


Рис. П2.1. Главное окно программы

На рис. П2.1 приведено изображение главного окна программы. Для запуска алгоритма обучения необходимо выбрать текстовый файл, содержащий обучающую выборку и нажать на кнопку «Train network». Обученная сеть отображается в центральной части окна (рис. П2.2), при этом входные нейроны расположены слева, а выходные – справа. В нижней части находятся результирующие данные о процессе обучения, включающие:

- величину ошибки найденного нейросетевого решения (Error);
- количество скрытых нейронов (Neurons);
- общее число межнейронных связей в сети (Connections);

- количество поколений генетического алгоритма (Generation);
- общее количество вычислений целевой функции (FE).

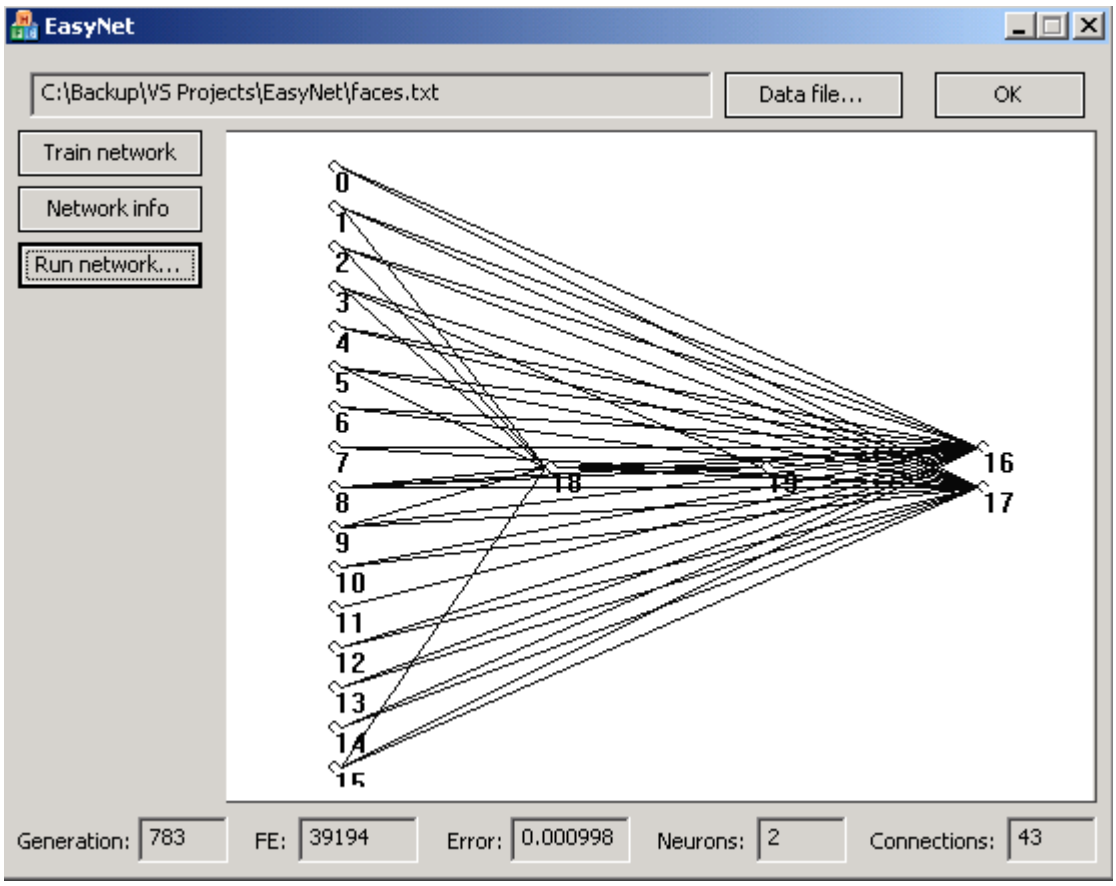


Рис. П2.2. Результат обучения

Более подробные данные о сети выводятся при нажатии на кнопку «Network info» (рис. П2.3). При этом отображается следующая информация:

- обучающая выборка;
- значения выходов сети при подаче различных наборов входных данных;
- список связей, где для каждой связи указываются индексы начального и конечного нейронов, а также вес связи.

Для использования сети с другими, отличными от обучающих, данными необходимо нажать кнопку «Run network...». В открывшемся окне (рис. П2.4) в левой части необходимо ввести входные данные. Подсчет выходных сигналов сети начинается после нажатия кнопки «Calculate network output». При этом значения выходов сети отображаются в правой части окна.

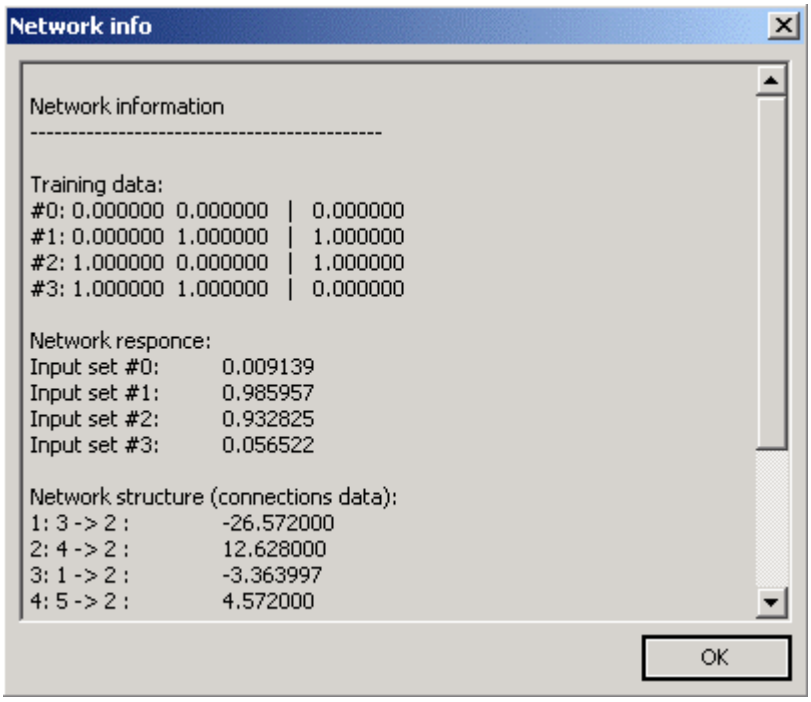


Рис. П2.3. Информация о сети

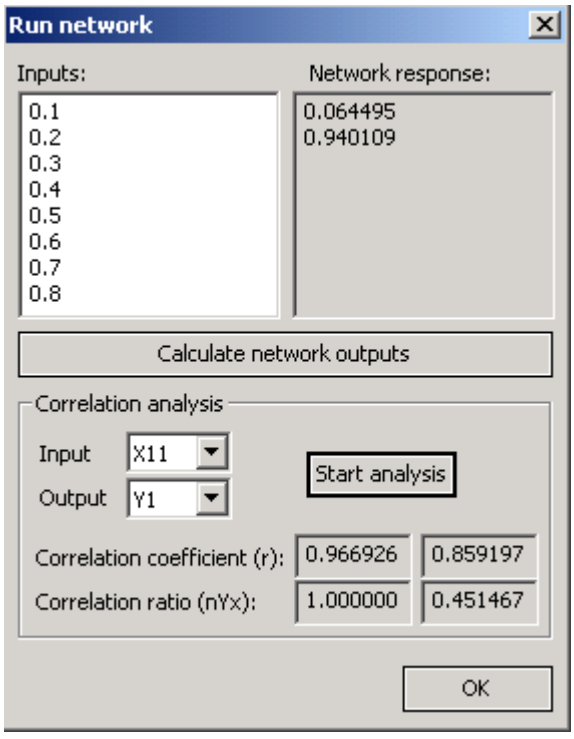


Рис. П2.4. Использование сети

Также в целях более глубокого анализа решения возможно определение некоторых статистических характеристик полученной нейросетевой модели, например, величину коэффициента корреляции и корреляционного отношения. В частности, в программе EasyNet реализована возможность вычисления этих

характеристик для случая линейной регрессии для пары, включающей один входной и один выходной параметр. Для сравнения полученных значений во втором столбце также приводятся величины этих же параметров, подсчитанные для данных из обучающей выборки.

Программа EasyNet написана на языке программирования C++ с использованием стандартной библиотеки шаблонов (STL) в среде Visual Studio .Net.