УДК 681.3

А.Б. Гаврилюк, В.А. Алексеев

МЕТОД ОПТИМАЛЬНОГО СТАТИЧЕСКОГО ПЛАНИРОВАНИЯ ЗАДАЧ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Рассматриваются методы погружения задач в распределенных системах и исследуются критичные характеристики методов для оптимизации скорости вычислений. Предлагается метод погружения, основанный на генетических алгоритмах, как более эффективный для данной задачи. Описываются экспериментальные результаты с применением данного метода.

Введение

В настоящее время для решения достаточно сложных задач не хватает мощности одиночных процессоров, несмотря на довольно большие величины, выражаемые в гигагерцах и сотнях мегафлопсов (мегафлопс — 1 миллион операций с плавающей запятой в секунду). Приведем несколько направлений, где необходима высокая мощность вычислительных систем:

- моделирование сложных химических или физических процессов,
- векторно-матричные вычисления,
- обработка 3D графики,
- хранилища информации...

Существуют два пути повышения производительности вычислительных комплексов:

- повышение частотных возможностей элементной базы,
- увеличение количества одновременно работающих вычислителей в системе.

Недостаток первого в том, что повышение частоты процессора ограничено размером кристалла и, судя по всему, производители уже подошли к границе использования данного метода. Так что для решения задач высокой сложности данный метод не может эффективно использоваться.

Второй путь приводит к необходимости разработки эффективного параллельного алгоритма задачи, а также задачи оптимального планирования и отображения параллельного алгоритма в структуру вычислительной системы. По аппаратным особенностям, для решения задачи параллельного вычисления алгоритма более удобны распределенные системы.

Обычно параллельные системы подразумевают использование одинаковых по параметрам процессоров, быстрых каналов передачи данных или даже полносвязных соединений, дорогое ПО. В противовес этому возможно на основе недорогих компьютеров бизнес-класса создать систему, которая сможет с такой же эффективностью параллельно обрабатывать данные. Удобством этих систем является возможность использования неоднородных по производительности процессоров и связей между ними, реконфигурирование системы по мере надобности. В данном случае возникают определенные сложности при решении задач оптимального планирования параллельного алгоритма.

Для эффективной работы параллельной программы на распределенной системе необходим адаптивный алгоритм, которому приходится решать NP-полную задачу планирования задач на ресурсы системы. В общем виде математическое решение для данных задач на современных компьютерах может занимать от нескольких десятков минут до нескольких месяцев. Поэтому для решения этих проблем используют эвристические подходы. Учитывая, что исходная задача задана в виде ориентированного ациклического графа в ярусно-параллельной форме,

ярусно-параллельной форме, их можно формализовать в три группы:

- списочные;
- кластерные;
- генетические.

Списочное планирование [1, 4, 5, 10] эффективно только для систем с общей шиной или однородных полносвязных систем. Работа алгоритма заключается в формировании списка очереди готовых вычислительных заданий с помощью одного из эвристических методов и затем оптимального, имея в виду пересылки данных, назначения очереди вычислительных работ на свободные процессоры. Две эти задачи решаются независимо одна от другой, что приводит к неэффективности вычислений.

Кластерное планирование [2, 7, 9] в чистом виде применяется только для систем, где нет ограничений на выделяемые ресурсы, или в масштабируемых вычислительных системах. Задания из исходного графа делятся на группы (кластеры). Этот процесс называется кластеризацией, и его решение в общем виде имеет экспоненциальную временную сложность. Задания одной группы (кластера) выполняются на одном процессоре.

Особенность генетического планирования [11—14] заключается в возможности решения задач планирования и назначения нераздельно одна от другой, это приводит к уменьшению сетевого трафика и оптимальной загрузки процессоров в системе. В результате эффективность использования процессоров выше, а время решения задачи приближается к критическому (идеальному без учета пересылок).

Исследование основных проблем планирования и способы их решения

При решении проблемы планирования задачу планирования алгоритма приходится решать в пространственно-временных координатах [6—9], причем временные координаты — это планирование времени выполнения каждой подзадачи и пространственные — оптимальное погружение подзадач на вычислитель (процессор) системы. Данная задача является NP-полной и решается эвристическими методами, качество которых может как угодно близко стремиться к идеальному, но никогда не достигнет его.

Исходными данными для алгоритмов планирования являются:

- параллельный алгоритм;
- параметры вычислительной системы.

Параллельный алгоритм представлен в виде ациклического графа (рис. 1), в котором вес вершины *Тз* под номером *Nз* обозначает вычислительную сложность подзадачи, а вес дуги *Тп* — объем данных для пересылки в адресное пространство другой задачи. Коэффициент квантования веса вершин и дуг является переменной величиной, соразмерной с зернистостью алгоритма задачи.

Параметрами вычислительной системы являются:

- характеристика узла системы (вычислителя);
- тип;
- топология.

В данной статье рассмотрим неоднородные вычислительные системы. Таким образом, тип вычислительной системы является распределенным, топология — конфигурируемой и робастной.

Характеристиками узла системы являются:

- производительность процессора (количество тактов в единицу времени);
- количество физических каналов пересылки данных (линков);
- скорость передачи данных по линкам (количество единиц информации в единицу времени) и возможность пересылки в разных направлениях (дуплексные и полудуплексные);
- возможность одновременного выполнения вычислений и пересылок.

Вычислительная система задается в виде графа (рис. 2), на котором *Nп* — номер процессора, *Sп* — скорость процессора в квантах вычислений. Параметры физических каналов пересылки задаются в виде *Sл* — скорости свя-

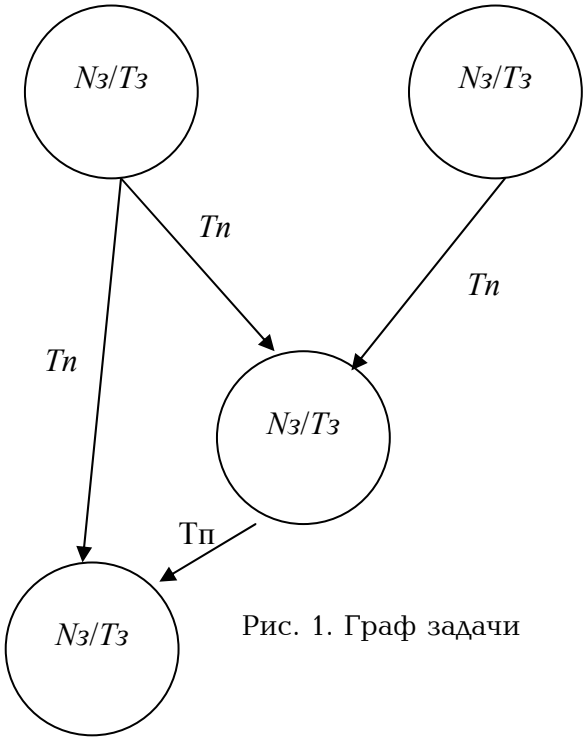


Рис. 1. Граф задачи

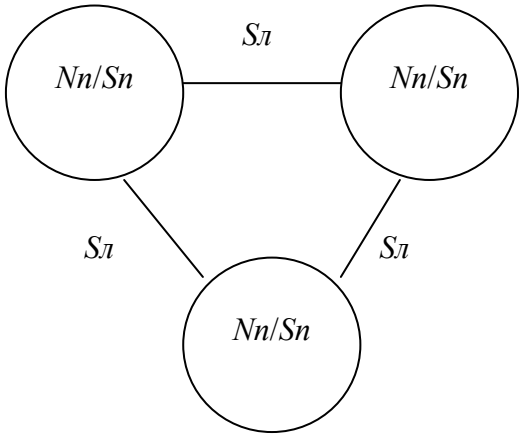


Рис. 2. Граф системы

зи в количестве квантов данных, которые могут передаться за 1 квант времени.

Таким образом, планировщик с учетом всех вышеперечисленных требований должен решать одновременно две задачи: выполнять оптимальную по скорости загрузку процессоров системы и в то же время минимизировать время ее простоя, затрачиваемое на пересылки готовых данных между подзадачами.

В этом и заключается основная проблема планирования, решением которой является сведение двух задач к одной. Идеально для этого подходят эволюционные или генетические алгоритмы, которые являются частным случаем вероятностных алгоритмов. Особенность их в том, что существует возможность поиска оптимума в многокритериальных системах. Генетические алгоритмы позволяют найти глобальный оптимум в многокритериальной системе, причем сложность решения будет линейной. Будет позволительно заметить, что данное семейство алгоритмов идеально подходит для решения поставленной задачи и оптимизации планирования.

Согласно выбранному критерию генетический алгоритм должен обеспечивать минимальное время выполнения указанных задач на вычислительной системе, заданной топологии.

Критическое время *Ткритич* — минимальное время выполнения параллельного алгоритма, которое не учитывает затрат при пересылках данных и таким образом является идеальным, но в большинстве случаев недостижимым из-за того, что в реальной ситуации не удается избежать пересылок данных. Исключениями могут быть те случаи, когда удается частично избежать пересылок посредством назначения связанных процессов, находящихся на критическом пути, на один и тот же процессор, а остальные пересылки выполнять без потери времени, зарезервированного за счет параллельной реализации процессов. С другой стороны, максимальное время выполнения задачи в любой системе — это время ее на одном процессоре *Тпослед*. В этом случае пересылки данных отсутствуют и время определяется как сумма вычислительных сложностей каждого процесса алгоритма. Поэтому величина реального минимального времени выполнения алгоритма *Треальн* всегда находится в следующем диапазоне: $Ткритич \le Треальн \le Тпослед$, для его достижения алгоритм планирования должен обеспечить минимальную суммарную задержку начала исполнения для всех процессов.

Основные этапы работы генетического алгоритм.

Генетические алгоритмы являются одними из эволюционных алго-

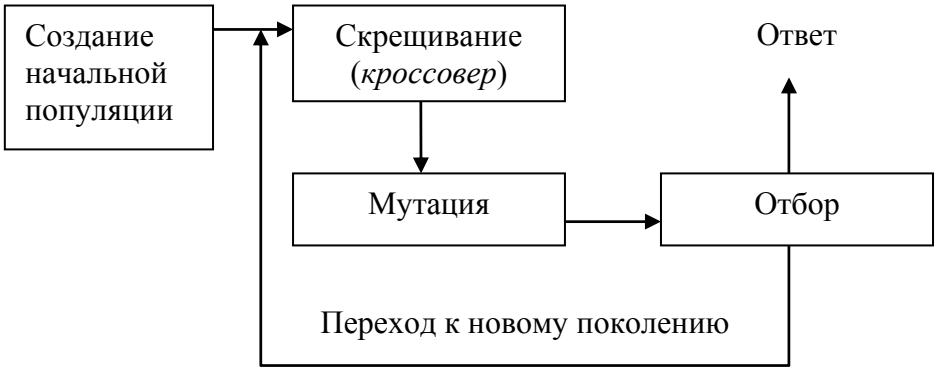


Рис 3. Схема работы генетического алгоритма

ритмов, применяемых для поиска глобального экстремума функции многих переменных. Принцип их работы основан на моделировании некоторых механизмов популяционной генетики (рис. 3): манипулирование хромосомным набором при формировании генотипа новой биологической особи путем наследования участков хромосомных наборов родителей (кроссовер) и случайное изменение генотипа, известное в природе как мутация. Другим важным механизмом, заимствованным у природы, является процедура естественного отбора, направленная на улучшение от поколения к поколению приспособленности членов популяции путем большей способности к "выживанию" особей, обладающих определенными признаками. Реализацию базового генетического алгоритма можно представить как итерационный процесс, включающий несколько этапов.

Генетический алгоритм случайным образом генерирует начальную популяцию структур $\{X1, \ldots, Xn\}$, каждая из которых должна однозначно отображать решение задачи. Каждая из структур или особей представляет собой одномерный или многомерный массив, на основании значений которого вычисляется значение целевой функции, например время работы параллельного алгоритма, которая и определяет качество каждого из решений.

Далее проводится кроссовер или скрещивание особей. Он может быть одноточечным, многоточечным или равномерным. При кроссовере случайным образом берутся биты из одной и другой структуры и заносятся в третью, которая и заменяет одну из них (рис. 4). Математический смысл данной операции — позиционирование решения внутри многокритериального пространства признаков.

После кроссовера происходят мутации структур или случайным образом и со случайной вероятностью замена некоторых битов, это уменьшает скорость сходимости алгоритма, но при этом и вероятность попадания решения в точку локального минимума. Некоторые из генетических алгоритмов не используют мутации.

При каждой итерации рассчитывается целевая функция по каждой из структур $F(Xi)$, $x = 1, \ldots, n$, которая определяет качество каждой из особей популяции. Несколько самых лучших особей отбираются и заносятся в элитный фонд. Метод элиты применяется для того, чтоб в процессе исследования не ухудшить качество решения. Далее итерационный процесс повторяется, пока не будет достигнут критерий остановки. После окончания работы алгоритма достигается максимум целевой функции, который и будет искомым.

Рис. 4. Кроссовер

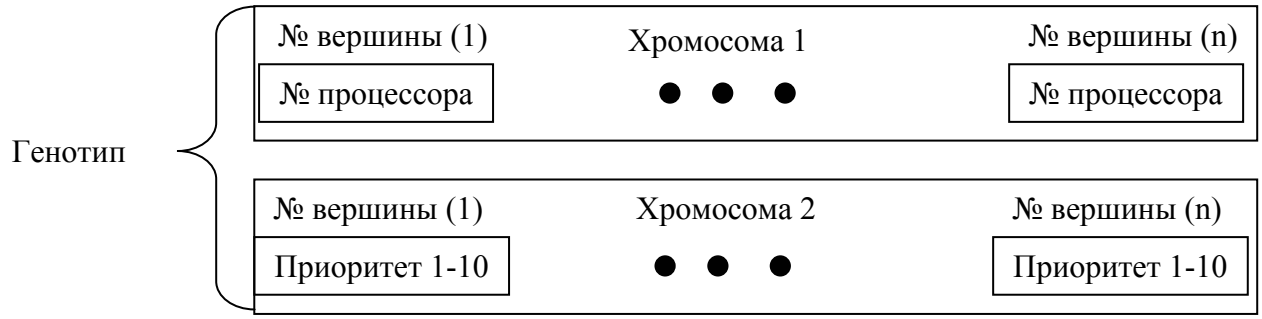


Рис. 5. Генотип системы

Хотя модель эволюционного развития, применяемая в генетических алгоритмах, сильно упрощена по сравнению природным аналогом, тем не менее она является достаточно мощным средством и может с успехом применяться для широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно, решить другими методами.

Функциональное описание генетического алгоритма планирования графа задачи на произвольную структуру вычислительной системы

Рассмотрим шаги генетического алгоритма погружения параллельной задачи в распределенную вычислительную систему.

1. Генерируется случайным образом N генотипов планирования граф (рис. 5). Как видно, индекс каждой ячейки массива — это номер вершины в графе задачи, а значение в ячейке — т номер процессора, на котором данная подзадача будет выполняться.

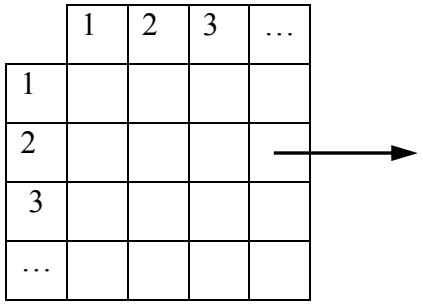
2. Строится матрица маршрутизации пакетов с данными внутри системы (рис. 6). Для определения кратчайшего пути используется метод Дейкстры, но возможно использовать любой другой. Каждая ячейка в матрице содержит два значения: время пересылки из вершины-источника (по горизонтали) к вершине-приемнику (по вертикали), номер следующей вершины на пути к заданной.

3. Осуществляется равномерный кроссовер всех хромосом со случайным образом выбранной из списка или в математическом выражении многоточечная аппроксимация между точками локальных оптимумов.

4. Осуществляется мутация всех хромосом с вероятностью Pm, вероятность является величиной переменной и зависит от необходимости скорости сходимости алгоритма. Выполняется случайное изменение значений для поиска глобального экстремума.

5. Вычисляется целевая функция для всех генотипов. На основании планировки, приоритетов пересылки и вычислений определяется время выполнения, которое и является целевой функцией алгоритма:

а) сортируются все вершины по их приоритетам. Высший приоритет получает задача, которая находится на



1. Время пересылки из вершины-источника к вершине-приемнику.
2. Номер следующей вершины на пути.

Рис. 6. Матрица маршрутизации пакетов

более высоком ярусе графа;

б) сортируются задачи по приоритетам пересылок данных из них. Приоритеты пересылок соответствуют приоритетам задач, для которых нужны вычисленные данные;

в) пока все вершины не выполнены, осуществляется:

– поиск не выполненной вершины, для которой получены все данные и процессор которой не был занят при текущем событийном цикле. Время выполнения вершины рассчитывается по формуле

$$Tp = max\ (TPнач, Tрнач)+Tз/Sn; \qquad (1)$$

где Tp — время, с которого процессор может начать выполнять следующую задачу, окончив эту;

$TPнач$ — начало выполнения задачи в квантах времени;

$Tрнач$ — время, с которого процессор может начать выполнять задачу;

$Tз$ — время выполнения задачи;

Sn — скорость процессора.

Маскируем процессор так, чтобы он снова не использовался для обработки новой задачи при текущем цикле исполнения;

– пересылка пакета/пакетов готовых данных по маршруту, который выбирается из матрицы маршрутизации. После пересылки время, когда по этой связи можно переслать другой пакет, вычисляется по формуле

$$Tл = max\ (TЛнач, Tпакета) + D*Sл; \qquad (2)$$

где $Tл$ — время, когда по этой связи можно переслать другой пакет;

$TЛнач$ — время, когда по этой связи можно начать пересылать пакет;

$Tпакета$ — время, когда можно начать пересылать пакет;

D — объем данных в пакете;

$Sл$ — скорость связи: чем больше значение, тем медленнее связь.

Если пересылки осуществляются не по дуплексной связи, то время, когда по ней возможно переслать другой пакет увеличивается в обе стороны, если связь дуплексная, то только в сторону пересылки пакета. Пока пакет данных не дошел до приемника, повторяем итерации;

— для подзадачи, которая получила пакет данных, время начала выполнения задачи рассчитывается по формуле

$$TPнач = max\ (TPнач, Tпакета); \qquad (3)$$

– осуществляется переход на два шага назад, пока все вершины не будут промоделированы;

г) время выполнения алгоритма выбирается максимальным из времени, завершения выполнения задач на процессорах (4):

$$Tвып = max\ (Tp1,\ldots,Tpn); \qquad (4)$$

где $Tвып$ — время выполнения параллельного алгоритма, искомая целевая функция;

$Tp1, \ldots, Tpn$ — время завершения задач на процессорах.

6. Если это первая итерация, то сравниваются целевые функций и $M < N$ лучших структур заносим в отдельный массив. Они используются для того, чтобы решение в процессе итераций не ухудшилось. Если это не первая итерация, то "**привилегированные"** структуры также участвуют в процессе сравнения.

7. Сортируются индексной сортировкой список структур и M последних худших структур заменяются на M структур, которые являются элитным решением.

8. С определенной периодичностью уничтожаются идентичные структуры, процесс выбора элитных структур приводит к созданию идентичных структур и попаданию решения в локальный экстремум.

9. Пока не закончилось число эпох (циклов алгоритма) или не достигнуто оптимальное для данной задачи решение, переходим на шаг 3.

Экспериментальные результаты

Разработана программа для экспериментального исследования результатов работы предлагаемого алгоритма, при помощи которой промоделировано решение задачи планирования.

Таблица 1. **Результаты статистического планирования графов задач на 9-процессорную полносвязную систему**

A	Количество задач в графе параллельного алгоритма					
	10	30	50	70	90	110
90/10	1.01	1.07	1.51	1.84	1.99	2.60
80/20	1.01	1.05	1.16	1.35	1.64	1.76
70/30	1.02	1.05	1.11	1.27	1.38	1.53
60/40	1.00	1.04	1.09	1.18	1.25	1.35
50/50	1.01	1.05	1.09	1.15	1.17	1.28

Исследование проводилось с целью выявления тех областей входных данных, при которых предлагаемый алгоритм дает наиболее оптимальный результат. При этом генерировались произвольные графы задач, которые варьировались и классифицировались по следующим параметрам:

- соотношение величины средней вычислительной сложности процессов в графе задачи (*Твып*) и усредненного значения времени пересылок (*Тперес*):

$$A = Твып/Тперес; \quad (5)$$

- размерность матриц графов задач.

В качестве критерия оценки выступал так называемый коэффициент эффективности (*Кэ*), который вычислялся как соотношение времени выполнения задания (*Твып*) и критического времени (*Ткрит*):

$$Кэ = Твып/Ткрит. \quad (6)$$

В табл. 1 и 2 даны результаты планирования.

Таким образом, можно заметить, что качество работы алгоритма не зависит от сложности задачи. Высокое значение Кэ при низкой связности задач определяется тем, что возможно не хватает процессоров для одновременного выполнения задач. При связности 50/50 качество планирования стремится к идеальному.

Выводы

Предложен новый алгоритм планирования на основе семейства генетических алгоритмов. Определено, что качество работы алгоритма близко к идеальному (полного перебора), причем сложность алгоритма возрастает

Таблица 2. **Результаты статистического планирования графов задач на 16-процессорную полносвязную систему**

A	Количество задач в графе параллельного алгоритма					
	20	40	60	80	100	120
90/10	1.01	1.12	1.38	1.29	1.85	1.62
80/20	1.04	1.03	1.12	1.23	1.40	1.28
70/30	1.04	1.09	1.12	1.17	1.28	1.19
60/40	1.04	1.07	1.10	1.15	1.25	1.22
50/50	1.04	1.08	1.10	1.14	1.11	1.18

линейно со сложностью задачи. Данный алгоритм является универсальным и ориентирован для работы с распределенными и конфигурируемыми вычислительными системами. При небольшой доработке данный алгоритм возможно использовать для динамического планирования.

1. *Русанова О.В., Нагорнюк В.В.* Двухпроходной эвристический алгоритм планирования и отображения для систем с распределенной памятью // Вестник НТУУ "КПИ" "Информатика, управление и вычислительная техника". — 1998. — №31. — С. 238—251.
2. *Князькова З.В., Симоненко В.П.* Метод направленного поиска при статистическом планировании задач в распределенных вычислительных системах // Пробл. программирования. — 2002. — №1—2. — С. 247—252.
3. *Grench V.* Массовые параллельные компьютеры // Computer World. — 1994. — **157**. — С. 53—58.
4. *Луцкий Г.М., Русанова О.В.* Проблемы отображения и планирования транспьютерных систем // 1-я Междунар. конф по параллельной обработке и прикладной математики PRAM'94. — Czestochowa, 1994. — P. 77—83.
5. *Berman F.* Experience with an automatic solution to the mapping problem // The Characteristics of Parallel Algorithms, MIT Press, Cambridge, MA, 1987. — P. 307—334.
6. *Bokhari S.H.* On the mapping problem // IEEE Trans. Comput. — 1981. — **C-30**. — P. 207—214.
7. *Kramer O., Muhlenbein H.* Mapping strategies in message based multiprocessor systems // Lecture Notes in Computer Science. — 1987. — **158**. — P. 213—225.
8. *Shen H., Occam H.* Implementation of process-to-processor mapping on the Hathi-2 transputer system // Microprocessing and Microprogramming—1991/1992. — **33**. — P. 173—189.
9. *Shen H.* Self-adjusting mapping: a heuristic mapping algorithm // Developing Transputer Applications, Amsterdam, IOS. — 1989. — P. 89—98.
10. *McFarland M., Parker A., Composano R.* Tutorial on high-level synthesis // Proc. 25th Design Automation Conference. ACM and IEEE. — 1988. — P. 330—336.
11. *Генетические* алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновкий, К.В. Махотило, С.Н. Петрашев, С.А. Сергеев. — ОСНОВА — 1997. — 112 с. — http://neuroschool.narod.ru/books/gannvirt.zip.
12. *Исаев С.А.* Популярно о генетических алгоритмах. — http://softlab.od.ua/algo/neuro/-ga-pop/index.htm.
13. *Батищев Д.И., Исаев С.А.* Оптимизация многоэкстремальных функций с помощью генетических алгоритмов. — http://-softlab.od.ua/algo/neuro/ga-opt1/index.htm.
14. *Исаев С.А.* Генетические алгоритмы — эволюционные методы поиска. — http://-softlab.od.ua/algo/neuro/ga-detail/index.htm.

Об авторах

Гаврилюк Александр Борисович
аспирант

Алексеев Виктор Анатолиевич
канд. техн. наук, зав 19 отделом Института программных систем

Место работы авторов:
Институт программных систем НАН Украины,
просп. Академика Глушкова, 40,
г. Киев-187, 03187, Украина
Тел. (044) 266 6321,
(044) 266 4228
E-mail: alife-soft@yandex.ru,
avictor@d19.isofts.kiev.ua