- отличное качество, если процент абонентов, испытавших затруднения при разговоре, не превышает 12,5%;
- хорошее качество, если процент абонентов, испытавших затруднения при разговоре, не превышает 25%;
- удовлетворительное качество, если процент абонентов, испытавших затруднения при разговоре, не превышает 50%;
- неудовлетворительное качество, если процент абонентов, испытавших затруднения при разговоре, превышает 50%.

Анализируя полученные результаты, можно сделать следующие выводы.

1. Для оценки мешающего действия токов электрического эха на качество передачи речи можно рекомендовать абонентские методы – метод заметности и затруднений.

2. Допустимым временем замедления, с точки зрения критерия «Заметность эха и искажений», в канале связи с эхо является время равное или менее 50 мс, при котором качество передачи речи оценивается как «хорошее».

Литература

1. Министерство связи РФ. Система автоматизированной телефонной связи общего пользования. Требования по установке эхоподавляющих устройств. Руководящий документ. Утвержден Минсвязи РФ 05.02.97.
2. Министерство РФ по связи и информатизации. Руководящий документ отрасли. Станции телефонные автоматические цифровые междугородные для применения на взаимоувязанной сети связи РФ. Общие технические требования. РД 45.158-2000. ЦНТИ «Информсвязь», Москва, 2002.

ESTIMATION OF STIRRING ACTION OF CURRENTS OF THE ELECTRIC ECHO IN MODERN TELEPHONE SYSTEMS

Ivanov V.I.

By results of a user's estimation of stirring action of currents of an electric echo recommendations about inclusion echo canceller devices in telephone channels on PSTN and networks under the concept «Voice over IP» (VoIP), ATM, cellular are made, etc.

Keywords: *an electric echo, the telephone channel, echo canceller devices, a user's estimation, subjective methods.*

Иванов Вячеслав Ильич, к.т.н., доцент Кафедры «Системы связи» Поволжского государственного университета телекоммуникаций и информатики. Тел. 8-927-260-23-11; (8-846) 202-62-84. E-mail: v.i.ivanov@mail.ru

ТЕХНОЛОГИИ КОМПЬЮТЕРНЫХ СИСТЕМ И СЕТЕЙ

УДК 004.42

ПОСТРОЕНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ПОВЕДЕНИЯ ПРОГРАММНЫХ АГЕНТОВ МЕТОДАМИ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Кольчугина Е.А.

Предложен способ создания параллельных алгоритмов на основе матричной промежуточной структуры с использованием описания алгоритма в виде древовидной структуры. Введены операции над древовидными структурами, позволяющие получать параллельные алгоритмы методами генетического программирования. Выделены и рассмотрены логические уровни, на которых протекает эволюция алгоритмов поведения программных агентов.

Ключевые слова: параллельный алгоритм, генетическое программирование, программный агент, эволюция

Введение

Одним из способов создания самоорганизующихся программных систем является применение такого класса моделей теории искусственной жизни, как создание искусственных миров. Программная система при этом представляет собой совокупность независимых активных единиц – агентов, которые, находясь в специальным образом устроенной среде, постепенно адаптируются к ней, подобно живым организмам в природе. Адаптация происходит либо путем изменения самого агента, либо путем подстройки его параметров, например весов нейронной сети. В первом

случае говорят об эволюции, во втором – об обучении. Критерием успешности адаптации отдельного агента служит значение счетчика «внутренней энергии», определяющее количество процессорного времени, в течение которого возможно продолжение выполнения агента. Следствием высокой приспособленности агента является появление у него «потомков», что приводит к появлению сообщества агентов с идентичным или схожим генотипом. В настоящей работе рассматривается модель, в которой агенты не имеют нейронной сети и развиваются путем эволюции алгоритма поведения агента, протекающей на нескольких уровнях.

Общая структура программного агента

Основные поведенческие свойства программного агента и параметры его состояния задаются последовательностью чисел, которую можно рассматривать как цифровую ДНК [1].

Часть этой последовательности задает макроалгоритм поведения программного агента. При этом номера позиций в последовательности идентифицируют блоки поведения, или элементы макроалгоритма, а значения элементов последовательности определяют программные реализации блоков поведения. Каждая программная реализация блока поведения представляет собой относительно независимую программную единицу, имеющую типовой интерфейс, на уровне которого и происходит взаимодействие с другими единицами.

В состав последовательности чисел, задающей цифровую ДНК, входит подпоследовательность, определяющая значения переменных состояния агента. При этом значения переменных можно рассматривать как нуль-аргументные функции, или как поведенческие блоки, реализующие операции присваивания значений переменным. Как и в случае кодирования алгоритма, номера позиций в последовательности чисел будут однозначным образом соответствовать переменным.

Цифровую ДНК в простейшем случае можно рассматривать как представление линейного алгоритма поведения агента, установив, например, правило, что блоки поведения будут исполняться в порядке возрастания их номеров. Однако в [1] был предложен другой подход, позволяющий строить на базе линейной цифровой ДНК параллельные алгоритмы поведения агентов. Для этого исходная числовая последовательность, представляющая цифровую ДНК, укладывалась в матрицу, называемую в [1] матрицей расписания. В [1] были предложены операции над матрицами расписаний. Расписание представляет собой промежуточную структуру, на основе которой с использованием примитивов, отражающих отношения последовательности выполнения во времени между подмножествами матрицы расписания, таких как seq (строгая последовательность выполнения без пересечений во времени), par (толерантность, или параллелизм) и $exec$ (выполнение одного блока поведения), может быть получено множество различных параллельных алгоритмов. Для получения конкретного параллельного алгоритма необходимо определить конкретный способ прочтения расписания как рекурсивное разбиение исходной матрицы расписания на подмножества.

Далее предлагается и рассматривается метод формирования способов прочтения матрицы расписаний, то есть параллельных алгоритмов, и операции, позволяющие преобразовывать параллельные алгоритмы.

Представление параллельного алгоритма в виде древовидной структуры

Матрица расписаний Q размером $m \times n$ представляет собой промежуточную структуру, которую можно рассматривать как прототип параллельной формы алгоритма. При этом номера столбцов являются аналогами пространственных координат, то есть задают номера процессорных элементов, на которых будут выполняться макродействия, или блоки поведения в составе алгоритма программного агента. Количество столбцов матрицы определяет требуемое количество процессорных элементов.

Множество номеров строк задает топологические шкалы времени для элементов матрицы: для двух элементов матрицы, находящихся в одном столбце, q_{ij} и $q_{(i+k)j}$, при $k > 0$ будет верно, что q_{ij} всегда заканчивает выполнение раньше начала выполнения $q_{(i+k)j}$.

Параллельный алгоритм определяется способом прочтения матрицы расписания, то есть рекурсивным разбиением матрицы на множество матриц меньшего размера, которые также разбиваются на подматрицы и так далее, вплоть до отдельных элементов исходной матрицы. При этом на каждом шаге выполнения разбиения матрица будет разбиваться либо на подмножества строк, либо на подмножества столбцов. Благодаря этому подмножества, образовавшиеся на одном шаге выполнения разбиения, будут находиться между собой в отношении либо seq, либо par, либо $exec$.

Такое разбиение может быть представлено в виде дерева, корнем которого является исходное расписание, листьями – блоки поведения, а промежуточными вершинами – матрицы меньшего, чем Q, размера, но содержащие более одного элемента. Очевидно, что арность дерева определяется как $p \le \max\{n, m\}$. Пример такого дерева, а также записи параллельного алгоритма в виде формулы приведен на рис. 1. Для индексации матриц-подмножеств Q используются индексы верхнего левого и правого нижнего элементов матриц в соответствии с их нумерацией в Q.

Дадим рекурсивное определение дерева, соответствующего способу прочтения расписания, то есть параллельному алгоритму:

$$Tr = \left\{\left\langle r, Q\right\rangle, \left\{Tr_1, \ldots, Tr_k\right\}\right\} \mid \varnothing, \qquad (1)$$

здесь r – отношение между поддеревьями, *seq*, *par* или *exec* (для терминальной вершины), Q – матрица расписания, которой соответствует дерево Tr, а $Tr_1, \ldots, Tr_k$ – поддеревья дерева Tr, каждое из которых соответствует матрице-подмножеству Q, причем $\bigcap_{i=1}^{k} Q_i = \varnothing$, $\bigcup_{i=1}^{k} Q_i = Q$.

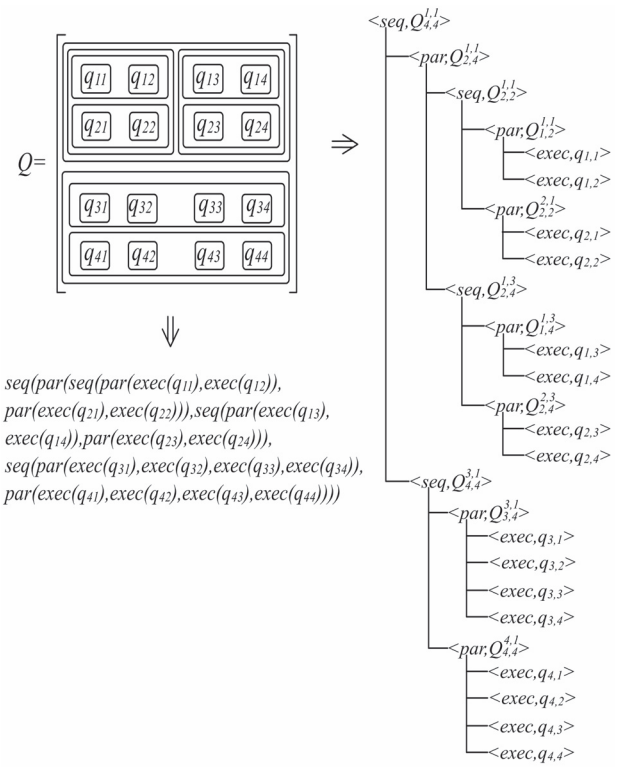

Рис. 1. Построение параллельного алгоритма в виде древовидной структуры и формулы на основе разбиения матрицы расписания Q

Операции над древовидными структурами

Поскольку варианты параллельных алгоритмов представлены в виде множества древовидных структур, для модификации имеющихся алгоритмов и порождения новых можно использовать методы генетического программирования разновидности автоматического программирования, в основе своей родственного генетическим алгоритмам и эволюционным вычислениям [2-3]. В случае генетического программирования каждая хромосома представляет собой программу на подобном языке программирования, например LISP, которая может быть представлена в виде древовидной структуры.

Типичный набор операций, используемых в генетическом программировании, включает в себя [2-3]:

- репродукцию;
- кроссинговер;
- мутацию;
- дупликацию и делецию генов;

Репродукцию можно понимать либо как генерацию новой, полностью случайной древовидной структуры, либо как результат применения кроссинговера, или соединения поддеревьев, к двум или более уже существовавшим древовидным структурам-родителям с возможной последующей мутацией. Мутация означает случайную перестройку поддерева исходной структуры. Операции дупликации и делеции, то есть дублирования или удаления генов, представляется более уместным выполнять на уровне матрицы расписаний. Применение делеции и дупликации на уровне древовидной структуры, задающей способ прочтения матрицы расписаний, то есть параллельный алгоритм, способно существенно усложнить задачу выбраковки агентов с нежелательными свойствами. При условии исключения делеции и дупликации, на основе одной и той же матрицы расписаний могут быть построены деревья с одинаковым корнем и листьями, хотя и имеющие различную высоту, сбалансированность и внутренние вершины.

```
построить_поддерево(матрица Q): дерево
begin
  Tr = new дерево;
  if (Q != q_ij) then
  begin
    x = rand();
    if (x > 0.5) then
      begin
        [Q_1; ...; Q_k] = Q;
        Tr = {⟨seq, Q⟩, {построить_поддерево(Q_1), ...,
             построить_поддерево(Q_k)};
      end
  else
      begin
        [Q_1, ..., Q_m] = Q;
        Tr = {⟨par, Q⟩, {построить_поддерево(Q_1), ...,
             построить_поддерево(Q_m)};
      end
  end
  else
    Tr = {⟨exec, q_ij⟩, {∅}};
return Tr;
end
```

Рис. 2. Операция построения древовидной структуры, описывающей параллельный алгоритм

Очевидно, что в основе реализации операций кроссинговера и мутации лежат две более простые операции:

- построение древовидной структуры случайным образом на основе исходной заданной матрицы расписаний или подмножества ее элементов;

- замена всей древовидной структуры или ее части (поддерева) другой древовидной структурой.

```
замена_поддерева(дерево Tr_i, Tr_j)
begin
  if  (Tr_i.Q == Tr_j.Q)
  then  Tr_i = Tr_j;
  else
    foreach  Tr_m ∈ {Tr_i Tr_1, ..., Tr_i Tr_k}
      замена_поддерева(Tr_m, Tr_j);
end
```

Рис. 3. Операция замены поддерева, описывающего часть параллельного алгоритма

Алгоритмы выполнения этих операций на псевдокоде приведены на рис. 2-3. Операции построения дерева и замены поддерева, так же как операции мутации и кроссинговера для заданной древовидной структуры, описывающей параллельный алгоритм поведения, были реализованы автором в виде программы на языке Perl. Древовидная структура и формула, приведенные на рис. 1, были получены с помощью этой программы.

Виды эволюции параллельных алгоритмов

В случае генетических алгоритмов эволюция программ происходит путем перебора решений из множества допустимых. Критерием отбора решений является фитнесс-функция, как правило, явно заданная. В рассматриваемом в данной статье случае фитнесс-функция задается имплицитно, то есть неявно. Оптимальность или неоптимальность параллельного алгоритма вносит вклад в общую приспособленность программного агента, то есть влияет на величину запаса его «внутренней энергии».

Таким образом, эволюция алгоритма поведения программного агента происходит на трех уровнях:

- на уровне первичной структуры цифровой ДНК, то есть числовой последовательности, кодирующей значения глобальных параметров агента и программные реализации блоков поведения;

- на уровне матрицы расписания, на котором возможны дупликация или исключение выполнения отдельных блоков поведения, а также определяются общие параметры параллельного алгоритма поведения: количество необходимых процессорных элементов и количество ярусов алгоритма;

- на уровне способа прочтения матрицы расписания, то есть конкретно заданного параллельного алгоритма.

Заключение

Предложенный в статье подход к рассмотрению параллельных алгоритмов как управляющих структур, или программ более высокого уровня, по отношению к исходному линейному алгоритму концептуально близок к принципам, которые можно наблюдать в живой природе на уровне белковых молекул. Так, химические свойства белковых молекул определяются не только их качественным составом, но и пространственной организацией. Известно, что в некоторых случаях белковые молекулы могут изменять свою пространственную структуру, заимствуя ее от других молекул. Такие явления служат, в частности, причиной прионной болезни.

Способ построения параллельных алгоритмов и их модификации на основе линейного алгоритмы с использованием матричной промежуточной структуры, предложенный в данной статье, может быть использован не только в теории искусственной жизни, но и для других приложений, в которых требуется автоматическая генерация и модификация параллельных алгоритмов на основе базового множества действий, аналогичных поведенческим блокам в рассмотренной модели.

Литература

1. Кольчугина Е.А. Эволюция расписаний как средство разработки параллельного алгоритма поведения цифрового организма // Известия вузов. Поволжский регион. Технические науки. №1(5), 2008. С. 45-52.
2. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М.: ФИЗМАТ-ЛИТ, 2006. 320 с.
3. Koza J.R., Bennett F.H, Andre D., Keane M.A. Genetic Programming: Biologically Inspired Computation that Creatively Solves Non-Trivial Problems // Evolution as Computation. DIMACS Workshop. Princeton, January 1999. Heidelberg: Springer-Verlag, 2001. P. 15-44.

CONSTRUCTION OF PARALLEL SOFTWARE AGENTS'S BEHAVIOUR ALGORITHMS BY METHODS OF GENETIC PROGRAMMING

Kolchugina E.A.

The way of parallel algorithm's construction on the basis of matrix intermediate structure with the use of the description of algorithm in the form of treelike structure is offered. Operations over the treelike structures are entered, allowing to produce parallel algorithms by methods of genetic programming. Logic levels on which evolution of algorithms of software agent's behaviors proceeds are described and considered.

Keywords: *parallel algorithm, genetic programming, software agent, evolution.*

Кольчугина Елена Анатольевна, к.т.н., доцент Кафедры «Математическое обеспечение и применение ЭВМ» Пензенского государственного университета. Тел. (8-841) 236-82-26; 8-902-354-68-28. E-mail: kea_sci@list.ru

УДК 681.5

ПРИМЕНЕНИЕ МОДЕЛЕЙ МАССОВОГО ОБСЛУЖИВАНИЯ В СИСТЕМАХ МОНИТОРИНГА ЭЛЕКТРОЭНЕРГЕТИЧЕСКИХ ПАРАМЕТРОВ

Воробьев А.Е., Лихтциндер Б.Я., Раскин А.Я.

В статье описывается способ определения предаварийного состояния на объектах электроэнергетики при помощи моделей массового обслуживания с целью снижения объема трафика, генерируемого системой мониторинга. Способ основан на применении алгоритма «текущего ведра» и адаптирован для использования с импульсными счетчиками электроэнергии.

Ключевые слова: теория массового обслуживания, задача о выбросах, системы мониторинга, уменьшение объема трафика, определение предаварийного режима, объекты электроэнергетики, счетчик электроэнергии, алгоритм «текущего ведра».

Рассматриваемые системы мониторинга (СМ) предназначены для организации сбора и отображения данных от территориально распределенных объектов электроэнергетики. Структура системы мониторинга (см. рис. 1) включает центральную часть и периферийные территориально распределенные объекты, объединенные между собой каналами связи.

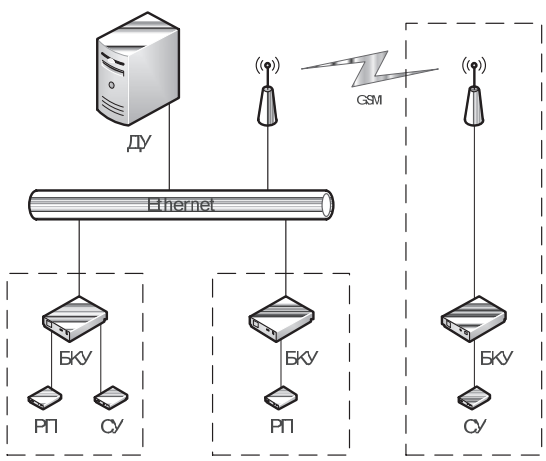

Рис. 1. Структура системы мониторинга

Каждая периферийная часть включает группу рабочих приборов (РП) и сенсорных устройств (СУ), объединенных общим блоком контроля и управления (БКУ). Информация, поступающая в каждый БКУ от соответствующих РП обрабатывается и по каналам связи передается в центральное диспетчерское устройство (ДУ).

Данные о состоянии РП формируются в БКУ и передаются в ДУ. Передача может происходить в одном из двух режимов:

- по запросу, когда все БКУ поочередно опрашиваются и передают информацию в ДУ;
- в независимом режиме, когда данные в ДУ поступают по инициативе БКУ. При этом должно быть обеспечено соответствующее разделение и независимость каналов связи.

При передаче в режиме опроса большого числа БКУ цикл опроса может оказаться слишком длительным. Время между двумя соседними опросами каждого БКУ может оказаться недопустимо большим, поскольку за это время может возникнуть предаварийная и развиться аварийная ситуации. Для уменьшения указанного времени следует увеличить частоту опросов, что влечет за собой увеличение объема передаваемых данных и соответствующее увеличение затрат на их передачу по каналам связи. Затраты на аренду каналов связи особенно возрастают, если для передачи используются GSM-каналы операторов мобильной связи.

В независимом режиме данные передаются от БКУ в ДУ не постоянно, а лишь в случаях возникновения существенных отклонений контролируемых параметров от их номинальных значений. В