функциями проекта являются автоматизированное составление расписаний для задач машинного типа, а также быстрая работа программы и достоверное решение.

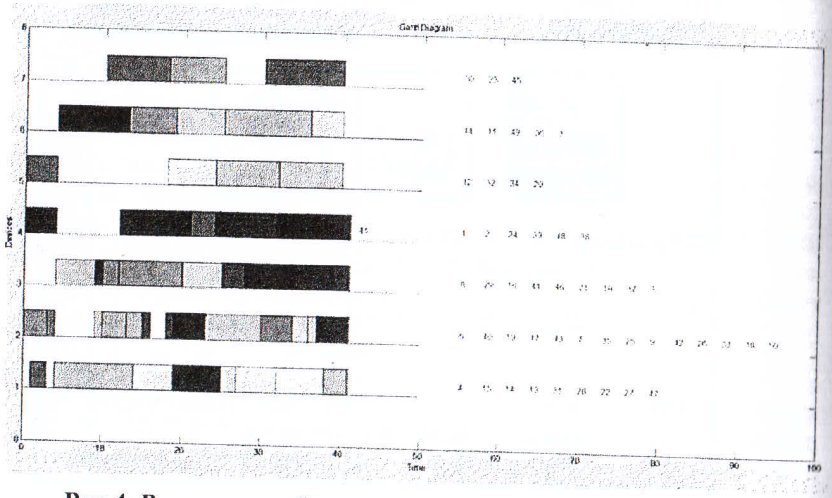

Рис.4. Результат работы программы (диаграмма Ганта)

Литература

1. Glover F., Kochenberger G.A. Handbook of Metaheuristics. – Boston: Kluwer, 2003.
2. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы/ Под ред. С.Н. Васильева. – М.: Изд-во ИПУ РАН, 2011.
3. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи: Пер. с англ. – М: Мир, 1982.
4. Афонин П.В., Громов С.А. Разработка генетического алгоритма для решения задач оперативного планирования производственных заданий// Труды международного конгресса по интеллектуальным системам и информационным технологиям-2010 (AIS-IT'2010, Дивноморское, 2-9 сентября 2010 г.). – М.: Физматлит, 2010. – Т.3. – С.18-23.

МЕТОД ПОСТРОЕНИЯ КОНЕЧНЫХ АВТОМАТОВ НА ОСНОВЕ МУРАВЬИНОГО АЛГОРИТМА

Чивилихин Д.С.,
e-mail: chivilikhin.daniil@gmail.com,
Ульянцев В.И.,
Шалыто А.А.
ulyantsev@rain.ifmo.ru
Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

1. ВВЕДЕНИЕ

В последнее время проводится все больше исследований в области *поисковой инженерии программного обеспечения* SBSE (Search-Based Software Engineering) [1, 2]. В этом подходе методы поисковой оптимизации применяются для автоматизированного построения программ для прикладных задач. Одним из классов таких задач является управление объектами со сложным поведением. Для подобных задач весьма эффективно автоматное программирование [3].

К достоинствам автоматного программирования относятся наглядность, возможность автоматической генерации кода по автомату и формальной верификации автоматных программ [4]. В некоторых случаях автоматы для программ этого класса удается построить эвристически, однако для многих задач ручное построение автоматов затруднительно или невозможно. Поэтому подход, в котором автоматы для программ строятся методами поисковой оптимизации, является актуальным. Основными методами поисковой оптимизации, применяемыми при построении конечных автоматов, являются генетические алгоритмы [5–7] и эволюционные стратегии [8].

В последнее время широкое развитие получили так называемые алгоритмы *роевого интеллекта* (Swarm Intelligence)[9], моделирующие поведение коллективов живых существ в целях решения комбинаторных задач. Подклассом алгоритмов роевого интеллекта являются *муравьиные алгоритмы* [10–14], в основном применяющиеся для решения задач оптимизации на графах.

Авторами настоящей работы был разработан метод построения конечных автоматов, основанный на муравьином алгоритме и представлении пространства поиска в виде ориентированного графа специального вида [15, 16]. В данной работе представлена

усовершенствованная версия этого метода, обладающая большей эффективностью. Экспериментальное исследование эффективности модифицированного метода проводится на задаче об «Умном муравье» [17].

2. КОНЕЧНЫЕ АВТОМАТЫ

Конечный автомат – это шестерка $\langle S, \Sigma, \Delta, \delta, \lambda, s_0 \rangle$, где S – множество состояний, Σ – множество входных событий, Δ – множество выходных воздействий, $\delta: S \times \Sigma \to S$ – функция переходов, $\lambda: S \times \Sigma \to \Delta$ – функция выходов, а $s_0 \in S$ – стартовое состояние.

Для представления автоматов в работе используются полные таблицы переходов и выходов. Пример конечного автомата с двумя состояниями приведен на рис.1. В табл. 1, 2 приведены примеры таблиц переходов и выходов автомата, представленного на рис.1.

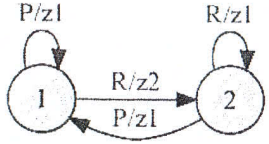

Рис. 1. Пример автомата с двумя состояниями, множеством входных событий $\Sigma = \{P, R\}$, множеством выходов $\Delta = \{z1, z2\}$ и стартовым состоянием $s_0 = 1$

Таблица 1. Пример таблицы переходов автомата из рис. 1

δ	Событие	
Состояние	P	R
1	1	2
2	1	2

Таблица 2. Пример таблицы выходов автомата из рис. 1

Λ	Событие	
Состояние	P	R
1	$z1$	$z2$
2	$z1$	$z1$

Под *мутацией* конечного автомата понимается небольшое изменение его структуры – таблиц, задающих функции переходов и выходов. В данной работе используется следующие два типа мутаций конечных автоматов.

- **Изменение состояния, в которое ведет переход**. Для случайно выбранного перехода в автомате состояние s, в которое он ведет, заменяется другим состоянием, выбранным случайным образом из множества $S \setminus \{s\}$.

- **Изменение действия на переходе**. Действие a на случайно выбранном переходе заменяется на другое действие, выбранное случайным образом из множества $\Delta \setminus \{a\}$.

При использовании рассматриваемых алгоритмов для построения конечных автоматов вводится *функция приспособленности* ФП f автомата, которая является вещественной. Эта функция определена на множестве всех конечных автоматов. Значения ФП пропорциональны близости наблюдаемого поведения автомата к желаемому.

3. МУРАВЬИНЫЕ АЛГОРИТМЫ

Муравьиные алгоритмы – семейство алгоритмов оптимизации, основанных на поведении реальных колоний муравьев. Первый муравьиный алгоритм был разработан М. Дориго (M. Dorigo) и применялся для решения задачи о коммивояжере [10]. С того времени было разработано множество алгоритмов этого класса, таких как, например, *Ant Colony System* [11], *MAX MIN Ant System* [12] и *Rank-Based Ant System* [13]. Муравьиные алгоритмы успешно используются для решения таких сложных комбинаторных задач, как, например, задача о рюкзаке, задача об упаковке в контейнеры, квадратичная задача о назначениях [17].

Для применения муравьиного алгоритма к какой-либо комбинаторной задаче необходимо представить пространство поиска этой задачи в виде графа. В муравьиных алгоритмах решения строятся колонией «муравьев», каждый из которых использует вероятностную стратегию для перемещения по графу. Решения обычно представляют собой пути в графе, в то время как ребра и вершины графа являются компонентами решений. Каждому ребру (u, v) графа (u и v – его вершины) сопоставляется число τ_{uv}, называемое *значением феромона*. Также на ребре может быть задано число η_{uv}, называемое *эвристической информацией*. Различие между этими двумя величинами состоит в том, что эвристическая информация задается изначально, исходя из условий задачи, и не меняется во время выполнения алгоритма, в то время как значения феромона изменяются агентами-муравьями в процессе построения решений. Общая схема любого муравьиного алгоритма состоит из трех последовательных этапов, которые повторяются, пока не будет найдено решение или не выполнится какой-либо критерий останова.

1. **Построение решений муравьями**. Каждый муравей переходит от одной вершины графа к другой, строя некоторый путь. Следующее ребро выбирается с помощью некоторой вероятностной

формулы исходя из значения феромона и эвристической информации на этом ребре.

2. **Обновление значений феромона**. Значения феромона на ребрах графа, посещенных каждым муравьем, увеличиваются пропорционально значению функции приспособленности решения, найденного этим муравьем. Затем происходит испарение феромона – значения феромона на всех ребрах графа уменьшаются в одинаковое число раз.

3. **Дополнительные действия.** На этом необязательном этапе выполняются операции, которые не могут быть выполнены отдельными муравьями. Например, могут производиться локальные оптимизации найденных на данной итерации решений.

4. ПОСТРОЕНИЕ КОНЕЧНЫХ АВТОМАТОВ НА ОСНОВЕ МУРАВЬИНОГО АЛГОРИТМА

Авторами настоящей работы в [15, 16] был предложен метод построения конечных автоматов, основой которого является представление пространства поиска (множества всех автоматов с заданными параметрами) в виде ориентированного графа G, который обладает следующими свойствами.

- Каждая вершина графа G ассоциирована с конечным автоматом.
- Пусть u – вершина, ассоциированная с автоматом A_1, а v – вершина, ассоциированная с автоматом A_2. Тогда, если автомат A_1 может быть получен из автомата A_2 применением одной операции мутации, то граф G содержит ребра $u \rightarrow v$ и $v \rightarrow u$. В противном случае, вершины u и v не связаны ребром. Можно заметить, что для любой пары автоматов A_1 и A_2 и соответствующей пары вершин u и v в графе G существует путь из u в v и из v в u.
- Значения эвристической информации на всех ребрах одинаковы.

Предложенный авторами муравьиный алгоритм, используемый в рассматриваемом методе, существенно отличается от других муравьиных алгоритмов оптимизации, хотя и заимствует их основы. Во-первых, используемые в предлагаемом алгоритме графы могут быть чрезвычайно большими – для некоторых задач они могут содержать миллионы вершин. Такие графы невозможно хранить в оперативной памяти персональных компьютеров, и поэтому в алгоритме используются специальные механизмы для ограничения размеров рассматриваемых подграфов этих графов. Во-вторых, в

большинстве случаев в муравьиных алгоритмах как вершины, так и ребра графа являются компонентами решений – путей в графе. В этом алгоритме каждая вершина графа полностью представляет собой решение (конечный автомат), в то время как муравьи переходят от одного решения к другому в поисках оптимального. Опишем метод, предложенный в [15, 16].

4.1. Построение начального решения.

В начале работы алгоритма строится некоторое начальное решение, которое добавляется в граф и становится его первой вершиной. При построении этого решения сначала генерируется случайный конечный автомат с заданным числом состояний $|S|$. Таблицы переходов и выходов конечного автомата заполняются случайным образом.

4.2. Построение решений муравьями.

Процедуру построения решений муравьями можно разделить на два этапа. На первом этапе выбираются вершины графа, из которых муравьи начнут построение путей. Эти вершины выбираются из числа вершин пути муравья, который ведет в лучшую на момент выбора вершину. На втором этапе совершается одна итерация колонии, в ходе которой каждый муравей обходит граф, начиная с соответствующей стартовой вершины. Пусть муравей находится в вершине u, ассоциированной с автоматом A. Если у вершины u существуют инцидентные ей ребра, то муравей выполняет одно из следующих действий.

1. **Построение новых решений**. С вероятностью p_{new} муравей пытается создать новые ребра и вершины графа G путем выполнения фиксированного числа N_{mut} мутаций автомата A. После выполнения муравьем всех N_{mut} мутаций он выбирает лучшую из построенных вершин и переходит в нее. Процесс обработки одной мутации автомата A таков:
   - выполнить мутацию автомата A, получить автомат A_{mut};
   - найти в графе G вершину t, ассоциированную с автоматом A_{mut}. Если такой вершины не существует, создать ее и добавить в граф;
   - добавить в граф ребро (u, t).
2. **Вероятностный выбор**. С вероятностью $1 - p_{new}$ муравей вероятностным образом выбирает следующую вершину из множества N_u ребер, инцидентных вершине u. Вершина v выбирается с вероятностью, определяемой классической в муравьиных алгоритмах формулой:

$$p_{uv} = \frac{\tau_{uv}^{\alpha} \cdot \eta_{uv}^{\beta}}{\sum_{w \in N_u} \tau_{uw}^{\alpha} \cdot \eta_{uw}^{\beta}}, \quad \alpha, \beta \in [0,1].$$

Если у вершины u нет инцидентных ей ребер, то муравей всегда выполняет построение новых решений. Все муравьи в колонии запускаются «параллельно» – каждый из них делает по одному шагу до тех пор, пока все муравьи не остановятся. Каждый муравей может выполнить максимум n_{stag} шагов, на каждом из которых происходит построение новых решений либо вероятностный выбор, без увеличения своего значения ФП. После этого муравей будет остановлен. Аналогично, колония муравьев может выполнить максимум N_{stag} итераций без увеличения максимального значения ФП. После этого алгоритм перезапускается.

4.3. Обновление значений феромона.

Правило обновления значений феромона, применяемое в данной работе, основано на правиле, использующемся в алгоритме *Elitist Ant System* [14]. Значение феромона, которое муравей откладывает на ребрах своего пути, равно максимальному значению ФП всех автоматов, посещенных этим муравьем. Для каждого ребра (u, v) графа G хранится число τ_{uv}^{best} – наибольшее из значений феромона, когда-либо отложенных на этом ребре. Последовательно рассматриваются все пути муравьев на текущей итерации алгоритма. Для каждого пути муравья выделяется отрезок от начала пути до вершины, содержащей автомат с наибольшим на пути значением ФП. Для всех вершин из этого отрезка обновляются значения τ_{uv}^{best}, а значения феромона на всех ребрах графа G обновляются по классической формуле:

$$\tau_{uv} = \rho \tau_{uv} + \tau_{uv}^{best},$$

где $\rho \in [0,1]$ – скорость испарения феромона.

4.4. Недостатки изложенного метода и предлагаемые модификации.

Описанный метод обладает несколькими недостатками. В данном разделе приводится описание этих недостатков и способы их устранения, предложенные в данной работе.

На первых итерациях алгоритма расходуется слишком много вычислений ФП. Это связано с тем, что в начале работы алгоритма значения ФП решений далеки от оптимального, а следовательно, нет смысла рассматривать большое число соседей этих решений. Для

устранения этого недостатка предлагается к построенному случайному решению, алгоритм получения которого приведен в разд. 0, применять простейшую (1+1)-эволюционную стратегию [8] в течение небольшого фиксированного числа итераций. Эволюционная стратегия работает следующим образом: к текущему решению применяется оператор мутации, случайно выбранный из набора операторов, перечисленных в разд. 0, и вычисляется значение ФП измененного решения. Если значение ФП нового решения не меньше значения ФП текущего решения, то текущее решение заменяется новым. Процедура повторяется до тех пор, пока не будет достигнуто отведенное эволюционной стратегии число вычислений ФП.

В описанном алгоритме не используется эвристическая информация. В данной работе предлагается на каждом ребре (u, v) графа задать значение эвристической информации:

$$\eta_{uv} = \max(\eta_{\min}, f(v) - f(u)),$$

где $\eta_{\min}$ – небольшая положительная константа, например, 0,001. Максимум разности значений ФП и положительной константы берется для того, чтобы значения эвристической информации всегда были неотрицательными.

Неограниченное убывание значений феромона. Для устранения этого недостатка, по аналогии с *MAX MIN Ant System* [12], вводится фиксированная нижняя граница значений феромона $\tau_{\min}=0{,}001$, а формула обновления феромона приобретает вид:

$$\tau_{uv} = \max\left(\tau_{\min}, \rho \tau_{uv} + \tau_{uv}^{best}\right).$$

Кроме того, экспериментально было установлено, что одним из наиболее эффективных правил выбора стартовых вершин является выбор вершины, ассоциированной с лучшим на момент выбора решением, в качестве единственной стартовой вершины для всех муравьев. Это правило является более эффективным, чем правило, описанное в разд. 0.

5. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ

Для сравнения предложенного модифицированного метода построения конечных автоматов, основанного на муравьином алгоритме, с другими алгоритмами рассматривается задача об «Умном муравье» [17]. Она является одной из стандартных задач для сравнительного анализа эволюционных алгоритмов.

В задаче об «Умном муравье» задано тороидальное поле размером 32×32 клетки. На поле вдоль некоторой ломаной распределены «яблоки». В данной работе рассматриваются две конфигурации

задачи – поле Санта-Фе и поле Джона Мура (рис. 2). Оба поля содержат по 89 клеток с яблоками. Черные клетки содержат яблоки, серые клетки обозначают пустые клетки оптимального пути, а белые клетки пусты.

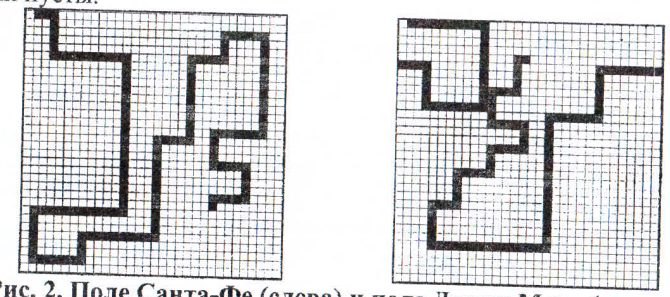

Рис. 2. Поле Санта-Фе (слева) и поле Джона Мура (справа)

Муравей изначально располагается в левой верхней клетке и «смотрит» на восток. Находясь в некоторой клетке, он может определить, есть ли в следующей клетке яблоко. На каждом временном шаге муравей может повернуть налево, повернуть направо или перейти вперед на одну клетку. Если в клетке, в которую переходит муравей, находится яблоко, муравей его «съедает». Число временных шагов s_{max}, доступных муравью, зависит от постановки эксперимента.

В описанной задаче целью является построение системы управления муравьем, которая позволит ему съесть все яблоки за отведенное число ходов. Используемая ФП имеет вид:

$$f = n_{food} + \frac{s_{max} - s_{last} - 1}{s_{max}},$$

где n_{food} – число яблок, съеденных муравьем, s_{max} – максимальное число ходов, предоставленных муравью и s_{last} – номер хода, на котором было съедено последнее яблоко.

Система управления муравьем строится в виде конечного автомата. У такого автомата есть два входных события – F (следующая клетка содержит яблоко), $!F$ (следующая клетка пуста) и три выходных воздействия: L (повернуть налево), R (повернуть направо) и M (сделать шаг вперед).

Во всех экспериментах использовались следующие значения параметров алгоритма: число муравьев $N_{ants} = 10$, скорость испарения феромона $\rho = 0.7$, параметр стагнации муравья $n_{stag} = 50$, параметр стагнации колонии $N_{stag} = 200$, число мутаций $N_{mut} = 60$, вероятность мутации $p_{new} = 0.6$, $\alpha = \beta = 1.0$.

5.1. Поле Санта-Фе

Результаты работы предложенного метода для поля Санта-Фе сравнивались с результатами, полученными в [19], которые на данный момент являются лучшими из всех опубликованных результатов для этой задачи при $s_{max} = 600$. Стоит отметить, что в указанной работе строятся не конечные автоматы, а так называемые *S*-выражения из языка программирования *LISP*. Тем не менее, результаты сравнения можно считать корректными, так как существует алгоритм трансформации *S*-выражения в конечный автомат [20].

При проведении экспериментов варьировалось как число состояний в целевом автомате N, так и наибольшее число доступных муравью шагов s_{max}. Эксперимент для каждой комбинации параметров N и s_{max} был повторен 10000 раз. В каждом из экспериментов предложенный метод нашел автомат, позволяющий муравью съесть всю еду. Например, для автоматов из семи состояний при $s_{max} = 600$ предложенный метод строит автомат, позволяющий муравью съесть всю еду, в среднем за 7203 вычисления ФП, в то время как в работе [19] для построения аналогичного *S*-выражения требуется 20696 вычислений ФП. На рис.3 приведены графики зависимости среднего числа вычислений ФП от числа состояний искомого автомата.

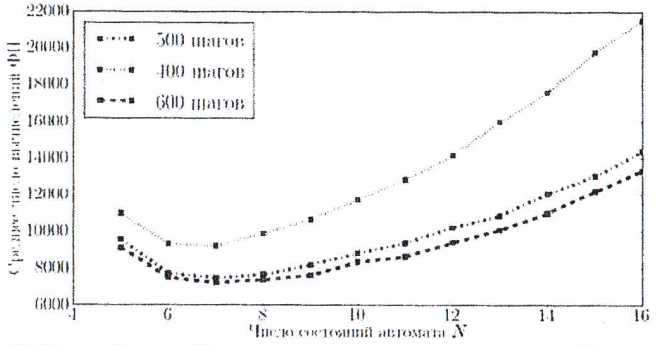

Рис. 3. Поле Санта-Фе: среднее число вычислений ФП при числе состояний автомата $N = 5..16$ и $s_{max} = \{400, 500, 600\}$

Сравнение полученных результатов с приведенными в [19] позволяет утверждать, что предложенный метод генерирует автоматы для поля Санта-Фе быстрее, чем любой другой из ранее опубликованных алгоритмов.

5.2. Поле Джона Мура

В первом эксперименте для поля Джона Мура результаты работы предложенного метода сравнивались с результатами работы генетического алгоритма [5]. В экспериментах было установлено ограничение $s_{max} = 200$, а число состояний автомата варьировалось от семи до 16. Для каждого числа состояний целевого автомата предложенный метод и генетический алгоритм запускались по 100 раз. Для автоматов из семи состояний предложенный метод находит решение в среднем за 31×10^6 вычислений ФП, что приблизительно в 60 раз быстрее, чем генетический алгоритм. Результаты для остальных размеров автоматов приведены на рис.4. Отметим также, что первоначальная версия предложенного в [15, 16] метода позволяла в данной постановке задачи получить решение, содержащее семь состояний, приблизительно за 200×10^6 вычислений ФП.

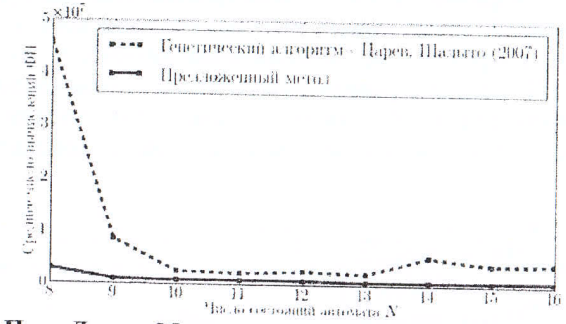

Рис.4. Поле Джона Мура: среднее число вычислений ФП при $s_{max} = 200$

Во втором эксперименте результаты работы предложенного метода сравнивались с результатами, приведенными в [21] при $s_{max} = 600$. В этой работе для построения автоматов применялся эволюционный алгоритм без кроссовера. Наряду с конечными автоматами в [21] строились также системы вложенных конечных автоматов. Кроме того, в эволюционном алгоритме использовались операторы добавления и удаления состояний, поэтому число состояний автоматов изменялось в течение работы алгоритма. Конечные автоматы при достижении идеального поведения содержали в среднем по 12 состояний. В случае систем вложенных автоматов, автоматы верхнего уровня содержали в среднем семь состояний, в то время как вложенные автоматы содержали до 12 состояний.

Предложенный метод не использует мутации изменения числа состояний, поэтому в целях проведения сравнения было проведено несколько экспериментов для различного числа состояний целевого автомата. Результаты экспериментов, а также результаты, полученные в [21], представлены на рис.5. Эксперименты показали, что предложенный метод позволяет решить задачу в среднем в несколько раз быстрее, чем эволюционный алгоритм из [21].

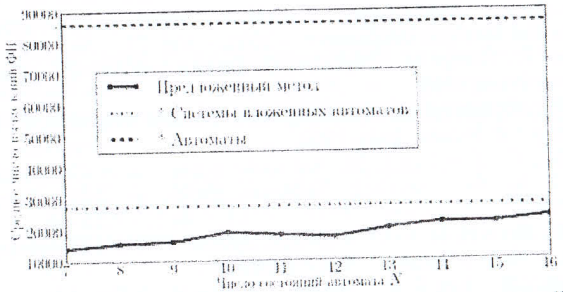

Рис.5. Поле Джона Мура: среднее число вычислений ФП при $s_{max} = 600$. [21]

6. ЗАКЛЮЧЕНИЕ

В данной работе предложена модификация метода построения конечных автоматов, основанного на муравьином алгоритме. Полученные экспериментальные результаты свидетельствуют о том, что для рассмотренных случаев задачи об «Умном муравье» предложенный метод эффективнее, чем генетическое программирование, генетические алгоритмы и эволюционные стратегии.

Литература

1. Clarke J., Dolado J., Harman M., Hierons R., Jones B., Lumkin M., Mitchell B., Mancoridis S., Rees K., Roper M., Shepperd M. Reformulating Software Engineering as a Search Problem // IEEE Proceedings – Software. – 2003. – Vol.150, №3. – P. 161–175.
2. Harman M. Software Engineering Meets Evolutionary Computation// Computer. – 2011. – Vol.44, № 11. – P. 31–39.
3. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб.: Питер, 2011.
4. Вельдер С. Э., Лукин М. А., Шалыто А. А., Яминов Б. Р. Верификация автоматных программ. – СПб.: Наука, 2011.
5. Царев Ф. Н., Шалыто А. А. О построении автоматов с минимальным числом состояний для задачи об «Умном муравье»// Сборник докладов XI-й международной конференции по мягким вычислениям и измерениям.

(SCM'2008, Санкт-Петербург, 23- 25 июня2008). – Изд-во СПбГЭТУ «ЛЭТИ». – 2007. – Т.2. – С. 88–91.
6. Tsarev F., Egorov K. Finite- State Machine Induction Using Genetic Algorithm Based on Testing and Model Checking // Proceedings of the 2011 GECCO Conference Companion on Genetic and Evolutionary Computation (GECCO'11). – 2011. –P.759–762.
7. Lucas S., Reynolds J. Learning Finite State Transducers: Evolution versus Heuristic State Merging // IEEE Transactions on Evolutionary Computation. – 2007. – Vol.11. – № 3. – P.308-325.
8. Lucas S., Reynolds J. Learning Deterministic Finite Automata with a Smart State Labeling Evolutionary Algorithm// IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2005. – Vol. 27. – № 7. – P. 1-12.
9. Bonabeau E., Dorigo M., Theraulaz G. Swarm Intelligence. – Oxford: Oxford University Press, 1999.
10. Dorigo M. Optimization, Learning and Natural Algorithms. PhD Thesis. Polytechnico di Milano, Italy, 1992.
11. Dorigo M., Gambardella L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem// IEEE Transactions on Evolutionary Computation. – 1997. – Vol.1. – № 1. – P.55–66.
12. Stützle T., Hoos H. H. MAX MIN Ant System // Future Generation Computer Systems. – 2000. – Vol.16. – №8. – P.889-914.
13. Bullnheimer B., Hartl R. F., Strauss C. A New Rank-Based Version of the Ant System: A Computational Study // Central European Journal for Operations Research and Economics. – 1999. – Vol.7, №1. – P.25-38.
14. Dorigo M., Maniezzo V., Colorni A. Ant System: Optimization by a Colony of Cooperating Agents // IEEE Transactions on Systems, Man, and Cybernetics. – 1996. –Vol.26, № 1. – Part B. – P.29-41.
15. Chivilikhin D., Ulyantsev V., Tsarev F. Test-Based Extended Finite-State Machines Induction with Evolutionary Algorithms and Ant Colony Optimization // Proceedings of the 14th International Conference on Genetic and Evolutionary Computation (GECCO'12). – 2012. – P.603-606.
16. Chivilikhin D., Ulyantsev V. Learning Finite-State Machines with Ant Colony Optimization// Proceedings of the 8th International Conference ANTS'12. Lecture Notes in Computer Science. Vol.7461. – Berlin: Springer, 2012. – P.268–275.
17. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. Evolution as a Theme in Artificial Life: The Genesys/Tracker System. Technical report. – 1990.
18. Dorigo M., Stützle T. Ant Colony Optimization. – Cambridge MA: MIT Press, 2004.
19. Christensen K., Oppacher F. Solving the Artificial Ant on the Santa Fe Trail Problem in 20,696 Fitness Evaluations// Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07). – 2007. – P. 1574 – 1579.
20. Kim D. Memory Analysis and Significance Test for Agent Behaviours // Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. – 2006. – P. 151–158.
21. Chellapilla K., Czarnecki D. A Preliminary Investigation into Evolving Modular Finite State Machines // Proceedings of the 1999 Congress on Evolutionary Computation. – 1999. – Vol.2. – P.1349-1356.

БИОНИЧЕСКИЙ ПОДХОД К РЕШЕНИЮ ЗАДАЧИ РАЗМЕЩЕНИЯ

***Щеглов С.Н.**, к.т.н., доцент*
Южный федеральный университет
e-mail: leo@tsure.ru

1. ВВЕДЕНИЕ

Проектирование БИС и СБИС в настоящее время зависит от резкого возрастания информационной сложности объектов автоматизированного проектирования и практически исключает возможность использования традиционных методов. Распространение в САПР начинает получать новая технология – эволюционное моделирование, основная на методах, инспирированных природными системами. Это особенно актуально при переходе на нанометровые нормы проектирования. При конструкторском проектировании основные задачи это компоновка, размещение, трассировка, верификация и экстракция. В настоящее время широкое распространение получили новые идеи и методы, реализующие задачи компоновки, размещения, а также построения стандартных и IP-блоков [1]. Принципиально новая методология проектирования систем на кристалле (СнК) основана на многократном повторном использовании на всех этапах ранее созданных IP-блоков и их совокупности [2]. Это аналогично эволюции в живой и неживой природе и обосновывает разработку и применение бионических алгоритмов в САПР.

Одним из важнейших этапов автоматизации проектирования является размещение элементов на непрерывном монтажном пространстве [1-3]. Каждый год появляются новые технологические процессы со сложнейшим перечнем технологических ограничений, которые делают процесс проектирования более трудоемким. Уменьшение размеров, переход к проектированию в наносекундном диапазоне приводит к тому, что на первый план выходят проблемы учета задержек распространения сигналов и энергопотребления блоков. Это приводит к необходимости разработки новых параллельно-последовательных алгоритмов размещения с доработкой технологии.

2. ПОСТАНОВКА ЗАДАЧИ

Пусть дано множество конструктивных элементов (блоков), связанных между собой в соответствии с принципиальной схемой