УДК 628.321.3

Муравьиный алгоритм определения критических связей в СБИС

Д. Ю. Запорожец, Д.В. Заруба, В.В. Курейчик

Южный федеральный университет, elpilasgsm@gmail.com

***Аннотация* — Статья посвящена разработке и исследованию модифицированного муравьиного алгоритма определения критических связей в СБИС на примере решения задачи о коммивояжере, а также выполнению экспериментальных исследований его характеристик. Данный алгоритм входит в состав метода роевого интеллекта, являющегося одним из биоинспирированных подходов, описывающих коллективное поведение децентрализованной самоорганизующейся системы, которая состоит из множества агентов (муравьев), локально взаимодействующих между собой и с окружающей средой. В статье приведена постановка задачи о коммивояжере. Описан модифицированный муравьиный алгоритм, позволяющий получать наборы квазиоптимальных решений. Проведен вычислительный эксперимент. Проведенные серии тестов и экспериментов позволили уточнить теоретические оценки временной сложности алгоритма оптимизации и их поведение для графов различной структуры. В лучшем случае временная сложность алгоритма (ВСА) ≈ O(n logn), в худшем случае – O(n3).**

***Ключевые слова* — оптимизация, задача коммивояжера, многоагентная система, муравьиный алгоритм.**

I. Введение

За последние годы стало важным освоение рынка электронных технологий в минимально короткие сроки, а также прогнозирование возможного финансового риска, закладываемого в процесс производства нового изделия. Перепрограммируемые вентильные матрицы (Field-Programmable Gate Arrays – FPGA) стали эффективным решением проблем, связанных с прогнозированием риска и временными показателями освоения рынка электронных технологий.

Технология создания FPGA обеспечила возможность «реального» производства гибких схем (прототипов) за минимальную стоимость: время выпуска 1 прототипа занимает несколько минут, а его стоимость составляет около 100 долларов. Перепрограммируемое устройство – это гибкая схема, логическая структура которой может быть скорректирована (задана и изменена) конечным пользователем без использования интегрированной схемы (статичной), собранной выпускающей компанией [1].

Важнейшим этапом в цикле проектирования FPGA является этап конструкторского проектирования, на котором решаются задачи разбиения (компоновки), планирования, размещения, трассировки (разводки), упаковки, верификации [2]. При решении этих задач возникает вопрос об оценке полученных результатов. Одной из таких оценок (критерием) является длина критической связи. Под критической связью, в основном, понимается самая «длинная» или набор критически «длинных» цепей. В общем же виде можно рассмотреть любую цепь, в том числе и критическую, как гамильтонов цикл, исключив из рассмотрения последнее звено цикла. С другой стороны, задача поиска гамильтонова цикла интерпретируется как задача нахождения пути коммивояжера. Авторами предлагается использовать именно такую интерпретацию для поиска критических связей в СБИС.

Задача построения пути коммивояжера (ЗК) является NP-полной и заключается в нахождении кратчайшего гамильтонова цикла с весами на ребрах в графе [2-5]. Задача о коммивояжере (ЗК) имеет теоретическую ценность, являясь асимптотической оценкой исследования различных эвристических алгоритмов. В данной статье предлагается использовать муравьиный алгоритм для определения пути коммивояжера. Муравьиные алгоритмы представляют собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений. Они могут использоваться как для статических, так и для динамических комбинаторных оптимизационных задач. Сходимость гарантирована, то есть в любом случае будет получено квазиоптимальное решение задачи. Проведен вычислительный эксперимент: была протестирована серия из 500 графов на 50, 75, 100, 300, 500, 700 и 1000 вершин. При этом временная сложность алгоритма (ВСА) не выходила из области полиномиальной сложности. В лучшем случае ВСА ≈ O(n logn), в худшем случае – O(n3).

II. Постановка задачи и краткий обзор

В неформальной форме ЗК трактуется следующим образом: коммивояжеру необходимо посетить W городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт по маршруту с минимальной стоимостью. Более строго имеем: полный граф G=(X,U), где |X| = n – множество вершин (города), |U| = m – множество ребер (возможные пути

между городами). Дана матрица смежности R(i,j), где i, j ∈1,2,…,n, задающая стоимости путей из вершины xi в xj. Требуется найти перестановку φ из элементов множества X, такую, что целевая функция (ЦФ)[3]

$$ЦФ(\varphi) \equiv R(\varphi(1),\varphi(n)) + \Sigma\{R(\varphi(i),\varphi(i+1))\} \rightarrow \min.$$

Если граф не является полным, то дополнительными ограничениями на φ являются $< \varphi(i), \varphi(i+1) > \in U$, $\forall$ i ∈ 1,2, … ,n-1 и $< \varphi(1), \varphi(n) > \in U$.

Целевая функция – суммарная стоимость ребер гамильтонова цикла заданного графа. Решение ЗК будем рассматривать для графов с весами на ребрах. Отметим, что значение ЦФ не зависит в частном случае от выбора вершины-начала маршрута. Фиксация вершины-старта может быть использована для сужения пространства поиска, которое в этом случае равно (n - 1)!. Поэтому ЗК чувствительна к выбору начальных условий. Время работы известных алгоритмов ЗК, основанных на полном переборе или на построении дерева решений (методы ветвей и границ и др.), растет экспоненциально в зависимости от числа вершин графа. Эти алгоритмы способны решать ЗК для малого числа вершин (n ≈ 50) за полиномиальное время. Поэтому для решения ЗК с n > 1000 используются и разрабатываются различные эвристики на основе роевого интеллекта. Рой рассматривается как многоагентная система, в которой каждый агент функционирует автономно по довольно примитивным правилам. Примером такой многоагентной системы является муравьиная колония [2,4,6-10].

Идея муравьиного алгоритма – моделирование поведения муравьёв, связанного с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. При своём движении муравей метит путь феромоном, и эта информация используется другими муравьями для выбора пути. Это элементарное правило поведения и определяет способность муравьёв находить новый путь, если старый оказывается недоступным.

В сравнении с генетическими алгоритмами (ГА) [11-13] муравьиные алгоритмы имеют некоторые преимущества:

базируются на памяти обо всей колонии вместо памяти только о предыдущем поколении;

меньше подвержены неоптимальным начальным решениям (из-за случайного выбора пути и памяти колонии) [6,7].

К недостаткам относятся следующие свойства муравьиных алгоритмов:

сходимость гарантируется, но время сходимости неизвестно;

обычно необходимо применение дополнительных методов, таких как локальный поиск;

сильно зависят от настроечных параметров, которые подбираются только исходя из экспериментов.

Важным свойством муравьиных алгоритмов является неконвергентность: даже после большого числа итераций одновременно исследуется множество вариантов решения, вследствие чего не происходит длительных временных задержек в локальных экстремумах [7]. Перспективными путями улучшения муравьиных алгоритмов являются on-line адаптация параметров с помощью базы нечетких правил и их гибридизация с другими методами природных вычислений.

III. Описание архитектуры поиска и основных блоков алгоритма

Пусть число городов будет равно n, тогда решением задачи о коммивояжере будет нахождение замкнутого минимального по длине пути, который проходит по всем городам только однажды.

Расстояние от города i до города j – d_{ij} определяется следующим соотношением:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (1)$$

К исходным данным относится также матрица размерностью (N,E), где N – описывает порядок размещения городов, а E – расстояние между городами (при этом граф должен быть полносвязанным).

Пусть $b_i(t)$ (i = 1, ..., n) – количество муравьев в городе i в момент времени t, при этом общее количество муравьев m:

$$m = \sum_{i=1}^{n} b_i(t). \quad (2)$$

Определим свойства муравья.

1. Каждый муравей обладает собственной «памятью», хранящей список городов $J_{i,k}$ (табу-лист), которые необходимо посетить муравью k, находящемуся в городе i.

2. Муравьи обладают «зрением», обратно пропорциональным длине ребра:

$$\eta_{ij} = 1/D_{ij}. \quad (3)$$

«Зрение» определяет «жадность» выбора муравья. Чем ближе находится вершина графа, тем она лучше «видна» и тем больше муравей стремится попасть в нее.

3. Каждый муравей способен улавливать след феромона, который определяет желание муравья пройти по данному ребру. Уровень феромона в некоторый момент времени t на ребре D_{ij} будет соответствовать величине $\tau_{ij}(t)$.

Интенсивность меток рассчитывается в соответствии с формулой:

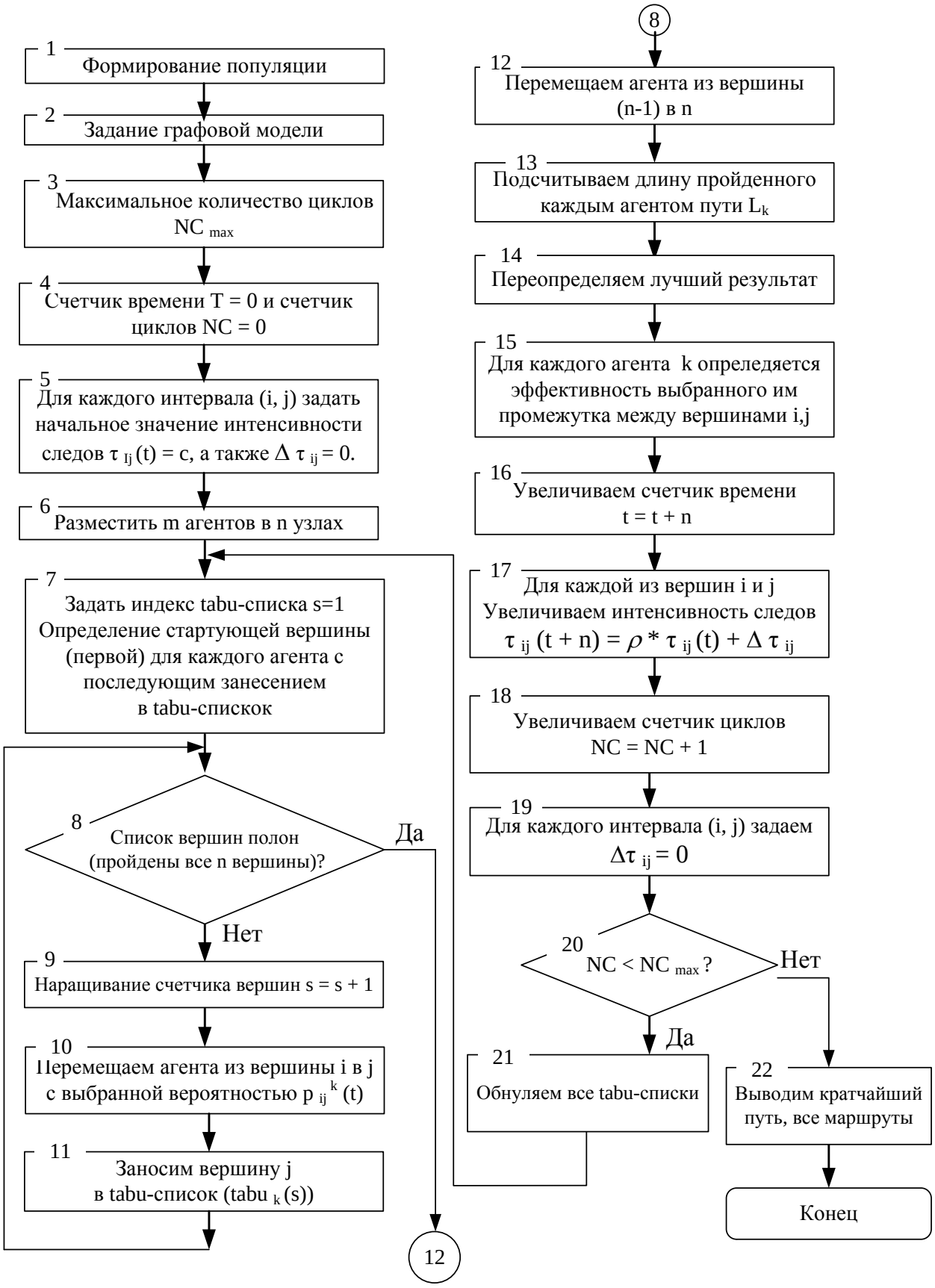

Рис. 1. Схема работы алгоритма

$$\tau_{ij}(t+n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij}, \quad (4)$$

где ρ – некоторый коэффициент, представляющий увеличение интенсивности меток на интервале (t, t+1),

$$\Delta\tau_{ij} = \sum_{i=1}^{m} \Delta\tau_{ij}^{k}, \quad (5)$$

где $\Delta\tau_{ij}^{k}$ – эффективность промежутка (i,j) на протяжении маршрута, выбранного k-тым агентом:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, \\ 0 \end{cases} \quad (6)$$

где L_k – длина маршрута, Q – управляющий коэффициент, подбирающийся экспериментально. Первое условие выполняется, если k агент использует направление (i,j) в момент времени (t, t+1). Второе условие вступает в силу в противном случае.

Коэффициент ρ должен быть меньше 1, чтобы предотвратить неограниченное накопление меток. Предполагается интенсивность следов $\tau_{ij}(t)$ в момент времени t=0, равной некоторой константе:

$$\tau_{ij}(0) = c. \quad (7)$$

Вероятность перехода муравья из вершины i в вершину j определяется следующим соотношением:

$$\begin{cases} P_{ij,k}(t) = \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}(t)]^{\beta}}, j \in J_{i,k}, \\ P_{ij,k}(t) = 0, j \notin J_{i,k} \end{cases} \quad (8)$$

где α, β – это параметры, задающие «вес» следа феромона. При α = 0 муравей стремится выбирать кратчайшее ребро, при β = 0 – ребро с наибольшим количеством феромона.

С течением времени вероятность выбора кратчайшего пути увеличивается, поскольку увеличивается количество откладываемого феромона.

Пусть начальным расположением колонии муравьев будет следующее: количество муравьёв равно числу вершин в графе, а каждому из муравьев соответствует вершина, с которой он начинает свое путешествие. Следовательно, возможность перехода обратно пропорционально зависит от расстояния (т.е. близко расположенные города будут выбраны с большей вероятностью, это отражает выполнение «жадного конструктивно-эвристического подхода») и интенсивности меток в момент времени t [4,5,10].

При рассмотрении схемы работы алгоритма по блочно можно отметить следующее:

Блоки 1-3 – задание начальных параметров работы системы.

Блоки 4-6 – инициализация входных параметров:

- Задать t=0, где t – счетчик времени.
- Для каждого интервала (i, j) задать начальное значение интенсивности следов $\tau_{ij}(t) = c$, а также

$$\Delta\tau_{ij} = 0.$$

- Разместить m агентов в n узлах, где m > n.

Блок 7 – определение стартующей вершины:

- Задать s=1, где s – индекс tabu-списка.
- Для $k \in (1, m)$ занести начальную вершину k агента в список tabu k (s).

Блоки 8-11 – цикл прохождения по всем вершинам с одновременным занесением в tabu-список.

- Продолжать до тех пор, пока tabu-список не станет пустым.
- Задать s=s+1.
- Для $k \in (1, m)$ выбрать вершину j для продолжения движения с вероятностью $p_{ij}^{k}(t)$. (При этом в момент времени t k агент находится в вершине i = tabu k (s-1)).
- Переместить k агента в вершину j.
- Вставить вершину j в tabu k (s).

Блоки 12-16 – обработка ситуации, когда завершился проход по всем вершинам: для всех $k \in (1, m)$.

- Переместить k агента из списка tabu k (n-1) в список tabu k (n).
- Подсчитать длину L маршрута, пройденного k агентом.
- Для каждого интервала (i, j) и для всех $k \in (1, m)$ рассчитать $\Delta\tau_{ij}^{k}$ по формуле 6, где первое условие выполняется, если интервал (i,j) принадлежит маршруту из списка tabu k. Второе условие вступает в силу в противном случае.
- Рассчитаем $\Delta\tau_{ij}$:

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}. \quad (9)$$

- Для каждого интервала (i,j) подсчитать $\tau_{ij}(t+n)$ по формуле 2.
- Задать t = t + 1.

Блоки 18-19 – наращивание счетчика циклов и обнуление интенсивности.

- Задать NC = NC+1.

• Для каждого интервала (i, j) задать $\Delta\tau_{ij} = 0$.

Блоки 20-22 – обработка условия NC < NC $_{max}$.

• Если NC < NC_{max} и не наблюдается стагнационное поведение, тогда:

• обнулить все tabu-списки;

• перейти на шаг 2.

• Иначе:

• выдать наикратчайший путь;

• конец.

Разработанный алгоритм работает до тех пор, пока счетчик числа маршрутов не достигнет максимума, установленного пользователем (NCmax) или все агенты не пойдут по одному пути. На каждой итерации сохраняются все маршруты, пройденные агентами, в рассматриваемой графовой модели. При выполнении следующей итерации алгоритма список маршрутов корректируется – повторяющиеся исключаются.

Среднее время работы модифицированного муравьиного алгоритма зависит от выбранного распределения вероятностей и числа вершин графовой модели. Использование моделей приведенного поиска позволяет за полиномиальное время получать список кратчайших маршрутов, а также определять и запоминать их длину. Время работы предлагаемого алгоритма в лучшем случае имеет порядок роста n, т.е. O(n), где n – число вершин графовой модели.

IV. Экспериментальные исследования

Рассмотрим результаты экспериментальных исследований построения пути коммивояжера. Интерфейс программы рассчитан на функционирование в среде Windows XP/7/8. Он реализован на языке C++ в Borland Buider 6.0.

В ходе проведения экспериментальных исследований были установлены эмпирические зависимости, диапазоны изменения входных параметров и выработан ряд рекомендаций по их оптимальному выбору. Временная сложность T представленного алгоритма рассчитывается как время, затраченное в среднем на одну итерацию алгоритма. Также была применена элитная селекция. Проведен эксперимент по определению зависимости времени решения T от числа генераций алгоритма G. При этом изменялось число циклов (генераций) в пределах от 50 до 200 с шагом 1. Эта зависимость приведена на рис. 2, где L – количество итераций, P – размер популяции, при котором производились исследования, а N – размер популяции. При исследовании стандартного теста Eilon's-50 результаты совпали с наилучшими существующими [2, 3]. Лучший путь, представленный на рис. 3, для Eilon's-50 составлял 425 условных единиц [12], при этом размер популяции составляет 20.

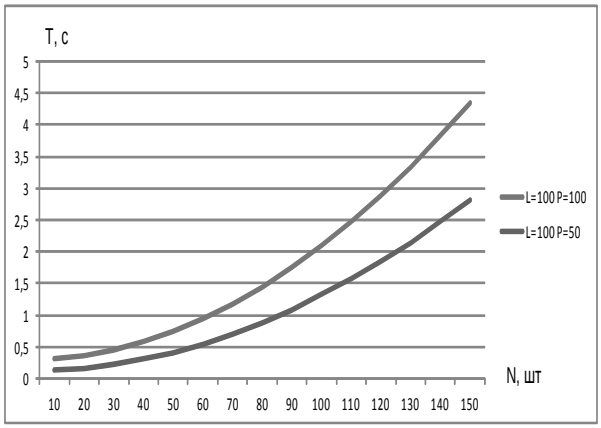

Рис. 2. График зависимости времени решения от числа популяций

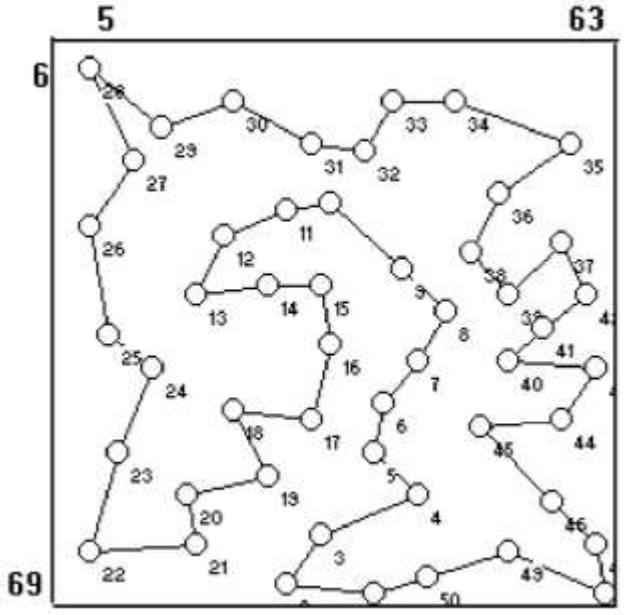

Рис. 3. Наилучший полученный маршрут в ЗК eilon's 50

График зависимости времени получения решений показан на рис. 4.

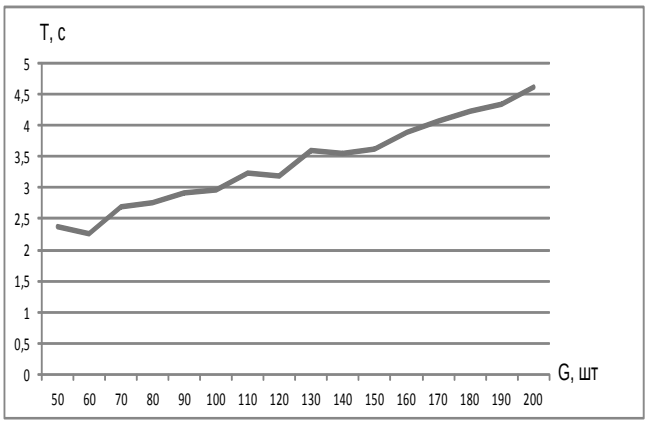

Рис. 4. График зависимости времени решения от числа генераций

На рис. 5 приведен график зависимости длины пути коммивояжера L от числа генераций алгоритма G.

Далее была протестирована серия из 500 графов на 100, 300, 1000, 7000 и 10000 вершин. При этом временная сложность алгоритма не выходила из области полиномиальной зависимости.

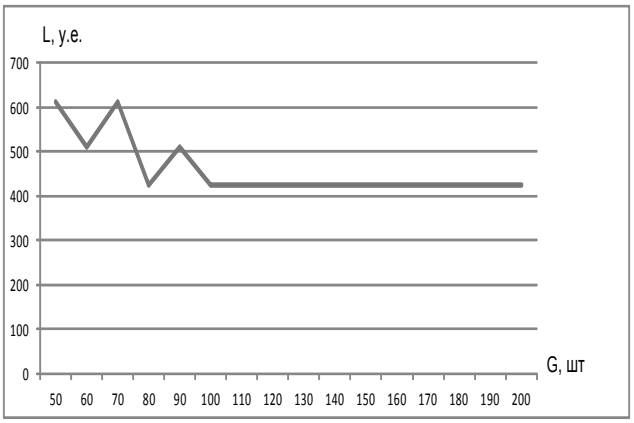

Рис. 5. График зависимости длины пути от числа генераций

V. Заключение

В данной статье рассмотрен муравьиный алгоритм нахождения критических связей в интерпретации решения задачи коммивояжера. Такой подход позволяет сужать область поиска за счет реализации критерия длин критических связей (гамильтоновых цепей), а так же частично решить проблему предварительной сходимости при решении задач конструкторского проектирования.

Проведены экспериментальные исследования разработанного модифицированного муравьиного алгоритма. Результаты моделирования позволяют говорить о ярко выраженном преимуществе данного подхода по сравнению с существующими аналогами. Среднее время работы модифицированного муравьиного алгоритма зависит от выбранного распределения вероятностей, числа вершин графа. Использование моделей приведенного поиска позволяет за кратчайшее время получать список критических связей рассматриваемых коммутационных схем, а также определять и запоминать длину кратчайших маршрутов.

В качестве основного достоинства предложенного модифицированного муравьиного алгоритма можно отметить быстрое нахождение эффективного решения независимо от диапазона изменения варьируемых параметров. Это позволяет говорить о гибкости и устойчивости рассматриваемого метода.

В ходе проведения вычислительного эксперимента были установлены эмпирические зависимости, диапазоны изменения входных параметров и выработан ряд рекомендаций по их оптимальному выбору. Проведенные серии тестов и экспериментов позволили уточнить теоретические оценки временной сложности муравьиного алгоритма. В лучшем случае временная сложность алгоритмов $\approx O(n\ logn)$, в худшем случае – $O(n^3)$.

Литература

[1] Stephen D. Brown. Field-Programmable Gate Arrays. Kluwer Academic Publishers.1992. 210 p.

[2] Kureichik V.M., Malioukov S.P., Kureichik V.V., Malioukov A.S. Algorithms for Applied CAD Problems // Berlin Heidelberg: Springer Verlag. Studies in Computational Intelligens. 2009. Vol. 212. P. 236.

[3] Курейчик В.В., Курейчик В.М. Генетический алгоритм определения пути коммивояжера // Известия Российской академии наук. Теория и системы управления. 2006. № 1. С. 94-100.

[4] Курейчик В.В., Курейчик В.М., Гладков Л.А., Сороколетов П.В. Бионспирированные методы в оптимизации. М.: ФИЗМАЛИТ, 2009. 384 с.

[5] Курейчик В.В. Бионические методы решения задачи коммивояжера // Вестник Южного научного центра РАН. 2005. Т. 1. № 4. С. 87-92.

[6] Abraham A., Grosan G., Ramos V. Swarm Intelligence in Data Mining. Berlin. Heidelberg: Springer Verlag, 2006. P. 267.

[7] Dorigo M., Maniezzo V., Colorni A. Ant System: Optimization by a Colony of Cooperating Agents // IEEE Transactions on Systems, Man, and Cybernetics-Part B. 1996. № 26(1). P. 29-41.

[8] Qing He, Xiu-Rong Zhao, Ping Luo, Zhong-Zhi Shi, Combination methodologies of multi-agent hyper surface classifiers: design and implementation issues. Second international workshop // AIS-ADM 2007, Proceedings. Berlin Heidelberg: Springer-Verlag. 2007. P. 100-113.

[9] Курейчик В.В., Запорожец Д.Ю. Роевой алгоритм в задачах оптимизации // Известия Южного федерального университета. Технические науки. 2010. Т. 108. № 7. С. 28-32.

[10] Курейчик В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений. М.: ФИЗМАТЛИТ, 2012. 260 с.

[11] Kurejchik V.V., Kurejchik V.M. ON GENETIC-BASED CONTROL // Автоматика и телемеханика. 2001. № 10. С. 174-187.

[12] Курейчик В.М., Курейчик В.В., Гладков Л.А. Генетические алгоритмы. М.: ФИЗМАТЛИТ, 2010. 368 с.

[13] Курейчик В.В., Сороколетов П.В. Концептуальная модель представления решений в генетических алгоритмах // Известия Южного федерального университета. Технические науки. 2008. Т. 86. № 9. С. 7-12.