УДК 681.3.06

Д.Ю. Запорожец

КОМБИНИРОВАННЫЙ АЛГОРИТМ РЕШЕНИЯ ТРАНСВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ*

На сегодняшний день трансвычислительные задачи активно используются в различных областях науки и техники. В статье рассматривается решение одной из важных задач конструкторского проектирования – задачи размещения блоков ЭВА. Данная задача относится к классу NP-трудных и NP-сложных задач, поэтому для ее решения необходима разработка перспективных эвристических методов получения квазиоптимальных решений за полиномиальное время. В статье приведена классическая постановка задачи размещения элементов СБИС, а также сформулирован критерий оптимизации. Предложен новый комбинированный подход к решению задачи размещения на основе моделирования поведения стаи волков. Разработанный биоинспирированный алгоритм значительно повышает эффективность поиска квазиоптимальных решений и позволяет эффективно управлять процессом поиска. Для проведения экспериментальных исследований была разработана программная среда, реализующая разработанный алгоритм. Были проведены исследования по сравнению разработанного алгоритма с оптимизационными методами на основе популяции, а именно, с генетическим, муравьиным и пчелиным алгоритмами. Результаты экспериментальных исследований *выявили зависимость времени работы алгоритма от параметров модифицированного генетического алгоритма. Серии экспериментов позволили уточнить теоретическую временную сложность алгоритма, которая ориентировочно составляет* $O(n^2)$.

Биоинспирированные алгоритмы, трансвычислительные задачи, алгоритм стаи серых волков, генетический алгоритм.

Yu.Yu. Zaporozhets

COMBINED ALGORITHM FOR TRANSCOMPUTATIONAL PROBLEMS

To date, transcomputational tasks are actively used in various fields of science and technology. The article discusses the solution of one of the important tasks of design engineering - the task of placing blocks of EVA. This task belongs to the class of NP-difficult and NP-complex problems, therefore, to solve it, it is necessary to develop promising heuristic methods for obtaining quasi-optimal solutions in polynomial time. The article presents the classical formulation of the problem of locating the VLSI elements, as well as the optimization criterion. A new combined approach to solving the placement problem based on modeling the behavior of a pack of wolves is proposed. The developed bioinspired algorithm significantly increases the search efficiency for quasi-optimal solutions and allows you to effectively manage the search process. For

* Работа выполнена при финансовой поддержке гранта президента Российской Федерации №МК-92.2017.8 (внутренний №ГрПр-10-2017/8МК).

experimental studies, a software environment was developed that implements the developed algorithm. Studies have been conducted comparing the developed algorithm with optimization methods based on population, namely, with genetic, ant and bee algorithms. The results of experimental studies revealed the dependence of the time of the algorithm on the parameters of the modified genetic algorithm. A series of experiments allowed to clarify the theoretical time complexity of the algorithm, which is approximately O (n^2).

Bioinspired algorithms, transcomputational tasks, algorithm of a pack of gray wolves, genetic algorithm.

Введение. Крупномасштабные тяжелые комбинаторные проблемы оптимизации возникают во многих областях. Одна из них – автоматизация физического проектирования. Такие проблемы, как разбиение, упаковка, размещение, маршрутизация, уплотнение – очень сложны. С математической точки зрения эти проблемы относятся к трансвычислительным задачам [1, 2].

На данный момент наиболее эффективными в решении трансвычислительных задач являются вероятностные алгоритмы, вдохновленные природой: роевые и эволюционные алгоритмы.

Таким образом, при композиции этих алгоритмов мы получим более совершенный, сочетающий в себе только лучшие свойства алгоритм. Полученный комбинированный алгоритм является наилучшим инструментом для решения трансвычислительной задачи размещения.

Трансвычислительной задачей называется задача, для решения которой необходимо обработать более 1093 бит информации. Данное число бит информации называется «пределом Бремерманна», и представляет собой общее число бит, обрабатываемых компьютером, размер которого равен размеру Земли, за период времени, равный существованию Земли.

Постановка задачи размещения. Основная цель размещения – организовать элементы таким образом, чтобы площадь кристалла была минимальна. А также минимизировать общую длину межсоединений для уменьшения временных задержек, возникающих в длинных цепях, тем самым, увеличив скорость обработки информации в СБИС. Создание необходимых и достаточных условий для трассировки [2–4].

Коммутационное поле (КП) организовано как регулярная структура. Ячейки (слоты) организованы как матрица строк и столбцов. Внешние ячейки ввода/вывода расположены вокруг. Области между слотами предназначены для маршрутизации (рис. 1).

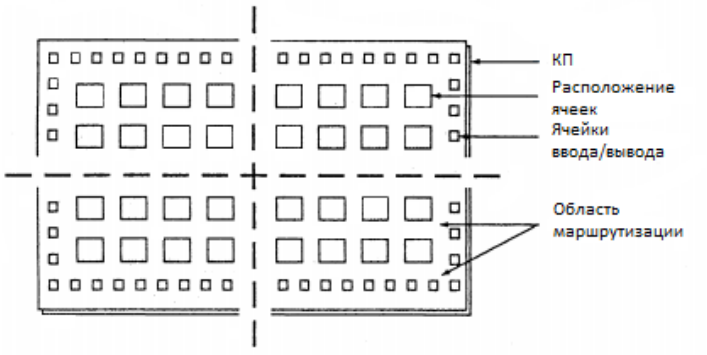

Рис. 1. Макет коммутационного поля

При ограниченном размере кристалла, который задаётся вручную, одной из важных задач является размещение элементов таким образом, чтобы они не пересекались и не накладывались друг на друга. Обычно для оценки качества размещений используют функцию качества, в которую включают оценки суммарной длины соединений и различные штрафы, включая пересечение или наложение элементов, а также описанные выше критерии.

В общем виде задача размещения формулируется следующим образом: в монтажном пространстве, задаётся область, которая разбивается на некоторое множество позиций (также применяется термин посадочных мест) $p = \{p_1, p_2, \dots, p_q\}$, число которых должно ровняться или быть больше числа размещаемых элементов. Каждый элемент может занимать только одно посадочное место. Расстояние между посадочными местами описывается симметричной матрицей расстояний $D = \|d_{ij}\|$. Имеющееся множество элементов $X = \{x_1, x_2, \dots, x_n\}$, которые связывается с множеством электрических цепей $E = \{e_1, e_2, \dots, e_n\}$, необходимо отобразить на множестве P таким образом, чтобы был достигнут экстремум целевой функции[2, 4–6].

Методы, используемые для решения задачи размещения. Алгоритмы для решения задачи размещения можно разделить на два основных класса:

- конструктивное размещение;
- итеративное улучшение.

Метод конструктивного размещения используется для создания места размещения с нуля; при итеративном улучшении алгоритмы начинаются с первоначального размещения и многократно изменяют его в поисках лучшего решения. Если изменение приводит к минимизации ЦФ, то модификация принимается, в противном случае отклоняется.

Ранние конструктивные алгоритмы размещения обычно основывались на примитивных правилах подключения. Т.е. модули выбирались по очереди в порядке их соединения с размещенными модулями (наиболее плотно соединенными в первую очередь) и размещались в свободном месте рядом с размещенными модулями, так чтобы длина провода сводится к минимуму.

Такие алгоритмы, как правило, очень быстрые, но обычно приводят к плохим результатам [5–7].

Эти алгоритмы теперь используются для генерации исходного размещения для алгоритмов итеративного улучшения. Основной причиной их использования является их скорость. Они занимают незначительное количество времени вычислений по сравнению с итерационными алгоритмами улучшения и обеспечивают хорошую отправную точку для них.

Более новые конструктивные алгоритмы размещения, такие как методы цифровой оптимизации, размещение по разбиению и т.п., дают лучшие результаты, но требуют значительно большего времени процессора.

Итеративные алгоритмы улучшения обычно создают хорошие размещения элементов, но требуют огромных вычислительных ресурсов. Простейшая стратегия итеративного улучшения изменяет случайно выбранные пары модулей и принимает обмен, если это приводит к снижению стоимости. Алгоритм прекращается, если в течение заданного большого количества испытаний дальнейшее улучшение не происходит. Улучшением этого алгоритма является повторное итеративное усовершенствование, при котором процесс итерационного улучшения повторяется несколько раз с различными исходными конфигурациями в надежде получить хорошую конфигурацию в одном из испытаний.

В настоящее время популярными алгоритмами итеративного улучшения являются метод моделирования отжига, генетический алгоритм и некоторые силонаправленные алгоритмы.

Генетический алгоритм решения трансвычислительной задачи. Генетические алгоритмы (ГА) – это адаптивные эвристические алгоритмы поиска, которые принадлежат большей части эволюционных алгоритмов. Генетические алгоритмы основаны на идеях естественного отбора и генетики. Они обычно используются для создания высококачественных решений для задач оптимизации и поиска [1–3].

В информатике существует большой набор трансвычислительных задач. Это в основном означает, что даже самые мощные вычислительные системы тратят очень много времени (даже годы!), чтобы решить одну трансвычислительную проблему. При таком ходе дел, ГА являются эффективным инструментом для обеспечения практически оптимальных решений за короткий промежуток времени.

Генетические алгоритмы имитируют процесс естественного отбора, что означает, что те виды, которые могут адаптироваться к изменениям в своей среде, способны выжить и размножаться, и перейти к следующему поколению.

Говоря простыми словами, они имитируют «выживание наиболее приспособленных» среди отдельных последовательных поколений для решения проблемы. Каждое поколение состоит из популяции отдельных индивидуумов, и каждый индивид представляет точку в пространстве поиска и возможное решение. Каждый индивид представлен аналогично хромосоме.

Генетические алгоритмы основаны на аналогии с генетической структурой и поведением хромосомы популяции. Ниже приведена база ГА, основанная на этой аналогии:

1. Индивид в популяции конкурирует за ресурсы и самку;
2. Те индивиды, которые являются сильнейшими, затем спариваются, чтобы создать больше потомства, чем другие;
3. Гены от «наиболее приспособленного» родителя распространяются по всему поколению, то есть родители создают потомство, которое лучше, чем любой из родителей;
4. Таким образом, каждое последующее поколение лучше приспосабливается к окружающей среде.

Обзор биоинспирированных алгоритмов. Естественные науки, и особенно биология, представляют собой богатый источник парадигм моделирования. Определенные области искусственного интеллекта (генетические алгоритмы, нейронные сети), математика и теоретическая информатика (L-системы, ДНК-вычисления) в значительной степени зависят от поведения различных биологических объектов и явлений.

В последние декады области, так называемых, естественных вычислений идентифицируют новые (нетрадиционные) вычислительные парадигмы в разных формах. Предпринимаются попытки определить и исследовать новые математические или теоретические модели, вдохновленные природой, а также исследования по определению парадигм программирования, которые реализуют вычислительные подходы, предлагаемые биохимическими явлениями.

Исследования получили новую перспективу после того, как Леонард Адлеман продемонстрировал возможность эффективно решать классическую комбинаторную «задачу о коммивояжере», используя пробирку с ДНК.

Можно надеяться, что глобальное поведение на системном уровне может быть переведено во взаимодействие множества элементов с простым поведением и ограниченными вычислительными и коммуникационными возможностями, которые могут выражать и решать с помощью различных оптимизаций сложные проблемы.

Биоинспирированные алгоритмы – это методы поиска, которые имитируют естественную биологическую эволюцию или поведение биологических объектов [1–5].

Биоинспирированные алгоритмы имеют широкий спектр приложений, охватывающих все наиболее распространенные области, включая:

- Компьютерную сеть;
- Безопасность;
- Робототехнику;
- Биомедицинскую технику;
- Системы контроля;
- Параллельную обработку;
- Сбор данных;
- Системы питания;
- Технологии производства и многое другое.

Оптимизация – часто встречающаяся математическая проблема во всех инженерных дисциплинах. Это буквально означает поиск наилучшего желательного решения. Проблемы с оптимизацией разнообразны и многочисленны.

Традиционные методы решения задач оптимизации требуют огромных вычислительных ресурсов, и, как правило, терпят неудачу по мере увеличения размерности задачи.

Данный факт стал толчком к использованию биоинспирированных алгоритмов оптимизации в качестве эффективной альтернативы детерминированному подходу.

Разработка биоинспирированного алгоритма, основанного на поведении стаи серых волков. Вдохновленный волчьей колонией в природе, Ву Хушенг предложил алгоритм «волчьей стаи» (ABC). Пространство искусственных волков можно обозначить как евклидово пространство $N \times D$, где N – число волков, а D – количество переменных. Вектор $X_i = (x_i^1, x_i^2, ..., x_i^D)$ представляет собой положение i-го искусственного волка, а x_i^d – это d-я переменная величины X_i. $Y = f(X)$ представляет целевое значение функции X, которое можно рассматривать как концентрацию запаха жертвы, воспринимаемого искусственными волками [7–9].

Волки являются типичными социальными животными, имеющими четкое разделение социальной работы. Волки в ABC можно разделить на три категории: волк-вожак, волки-разведчики и дикие волки. Все хищное поведение волчьей стаи сводится к трем интеллектуальным поведениям – разведке, охоте и окружению. Также существуют еще два правила: "победитель получает все" для вожака и правило обновления "выживает сильнейший" для остальной стаи. Далее приведено подробное описание этих действий и правил [9–11].

1) *Правило генерации "победитель получает все" для вожака:*

Вожак – волк с наилучшей целевой функцией. Таким образом, позицию вожака можно рассматривать как позицию жертвы. Во время каждой итерации значение функции вожака будет сравниваться со значением функции другого

волка; если значение Y_{lead} не самое лучшее, то вожак будет заменен. Тогда лучший волк становится новым вожаком. Вожаку не нужно выполнять три интеллектуальных поведения, он сразу переходит в следующую итерацию, пока другой, лучший волк его не заменит.

2) *Разведка:*

Не считая вожака, несколько элитных волков действуют как волки-разведчики. У волков-разведчиков значения функций лучшие, чем у диких волков. Y_i и Y_{lead} представляют собой целевые значения функций волка-разведчика и вожака соответственно. Если $Y_i > Y_{lead}$, это означает, что значение вожака не лучшее и вожак будет заменен волком-разведчиком, т.е. $Y_i = Y_{lead}$. Если $Y_i < Y_{lead}$, волки-разведчики будут действовать следующим образом.

Сначала -ый волк-разведчик пытается сделать шаг в сторону h различными путями и запомнить значения функции в каждом направлении. После шага в -ом направлении состояние i-го разведчика формулируется с помощью уравнения:

$$x_{i,d}^{p} = x_{i,d} + \sin(2\pi \cdot \frac{p}{h}) \cdot step_a.$$

Затем волк-разведчик выбирает направление, при котором значение целевой функции наилучшее и обновляет X_i. После этого поведение разведчика повторяется, пока $Y_i > Y_{lead}$ или не достигнуто максимальное количество повторений T_{max}. Следует отметить, что h для каждого волка различно и является целым числом, выбранным из $[h_{min}, h_{max}]$. Оно направлено на стимулирование различных стратегий охоты у разных волков. Переменная $step_a$ – это длина шага.

3) *Охота (погоня, преследование):*

Вожак воет и зазывает оставшихся диких волков собраться вокруг добычи. Здесь, позиция вожака рассматривается как возможная позиция добычи, чтобы дикие волки собрались к вожаку. Переменная $step_b$ – длина шага; g_d^k – положение вожака в d-ом переменном пространстве на -ой итерации, k – число итераций. Позиция дикого волка обновляется согласно следующему уравнению:

$$x_{i,d}^{k+1} = x_{i,d}^{k} + step_b \cdot \frac{g_d^k - x_{i,d}^k}{\left|g_d^k - x_{i,d}^k\right|}.$$

Если $Y_i > Y_{lead}$, дикий волк становится вожаком и $Y_i = Y_{lead}$; в таком случае новый вожак принимает охоту на себя. Если $Y_i < Y_{lead}$, дикие волки продолжают группироваться вокруг вожака с большой скоростью, пока $L(i, lead) < L_{near}$, а затем волк примет осаждающее поведение. $L(i, lead)$ представляет собой расстояние между вожаком и i-ым волком; L_{near} – это расстояние окружения, которое можно рассматривать в качестве условия суждения. Оно определяет, изменяет ли волк поведение с погони на окружение. L_{near} формулируется уравнением (9):

$$L_{near} = \frac{1}{D \cdot w} \cdot \sum_{d=1}^{D} |max_d - min_d|,$$

где w – коэффициент определения расстояния, а $[min_d, max_d]$ – диапазон значения переменной d.

4) *Окружение добычи:*

После движения большими шагами по направлению к вожаку, дикие волки близки к добыче. Затем они изменят свое поведение с погони на окружение добычи. Положение i-го дикого волка обновляется в соответствии со следующим уравнением:

$$x_{i,d}^{k+1} = x_{i,d}^{k} + \lambda \cdot step_c \cdot \left|g_d^k - x_{i,d}^k\right|,$$

где λ – случайное число, лежащее в интервале [-1,1]; $step_c$ – это длина шага при окружении добычи. Если $Y_i^{new} > Y_i^{old}$ после того, как волк начал окружать

добычу, то положение X_i обновляется; в противном случае его не следует изменять. Взаимосвязь $step_a$, $step_b$ и $step_c$ реализована следующим образом:

$$step_a = \frac{step_b}{2} = 2 \cdot step_c = S,$$

где S – это коэффициент шага и представляет собой сложную степень искусственных волков, охотящихся за добычей в пространстве решений.

5) *Правило обновления волчьей стаи "выживает сильнейший":*

Это правило имитирует естественный отбор Дарвина. По мере того, как добыча распределяется от сильных к слабым особям, это приведет к гибели некоторых слабых волков. Алгоритм будет регенерировать волков случайным образом при удалении R волков, которые недостаточно хороши. R представляет собой целое число и случайным образом выбирается в интервале $[N/2\beta, N/\beta]$. β – пропорциональный коэффициент обновления популяции.

Структурная схема алгоритма представлена на рис. 2.

Опишем основные шаги разработанного алгоритма:

1. Инициализация популяции серых волков Xi (i = 1, 2, ..., n).
2. Инициализация параметров a, A и C.
3. Оценка пригодность каждого агента поиска.
   Инициализация первого лучшего решения как X_α,
   второго лучшего решения как X_β, третьего лучшего решения как X_δ.
4. Максимальное количество итераций задается в начальных данных.
5. Инициализация цикла для n поисковых агентов.
6. Обновление позиции i-го поискового агента.
7. Обновление параметров a, A и C.
8. Оценка функции пригодности всех поисковых агентов.
9. Обновление лучших агентов X_α, X_β, X_δ.
10. Завершение алгоритма и визуализация значения первого лучшего агента X_α, найденного на данный момент.

Преимуществом разработанного комбинированного алгоритма является возможность улучшения каждой последующей стадии решения задачи размещения. Под стадией решения задачи размещения понимается ранее разработанные и использованные в рамках программного комплекса варианты решения задачи размещения, учитывая все заданные критерии.

Экспериментальные исследования. Для оценки производительности разработанного алгоритма был проведен ряд экспериментальных исследований для решения проблемы глобальной числовой оптимизации. Сравнение проводилось с оптимизационными методами на основе популяции, а именно с генетическим, муравьиным и пчелиным алгоритмами (ГА, МА и ПА соответственно).

Параметры ГА, МА, ПА и разработанного комбинированного алгоритма (КА) остаются одинаковыми в каждом эксперименте. Параметры КА устанавливаются следующим образом: размер популяции всех алгоритмов равен 100, количество итераций -300, вершин графа – 100. Параметры для ГА: *кроссинговер* = 0.95, *мутация* = 0.1.

В общем можно сказать, что КА обладает хорошей оптимизирующей способностью в сравнении с другими рассмотренными алгоритмами. Хотя на начальных итерациях КА показывал не лучшие результаты, но при увеличении итераций он показывает свою конкурентоспособность и обходит другие алгоритмы.

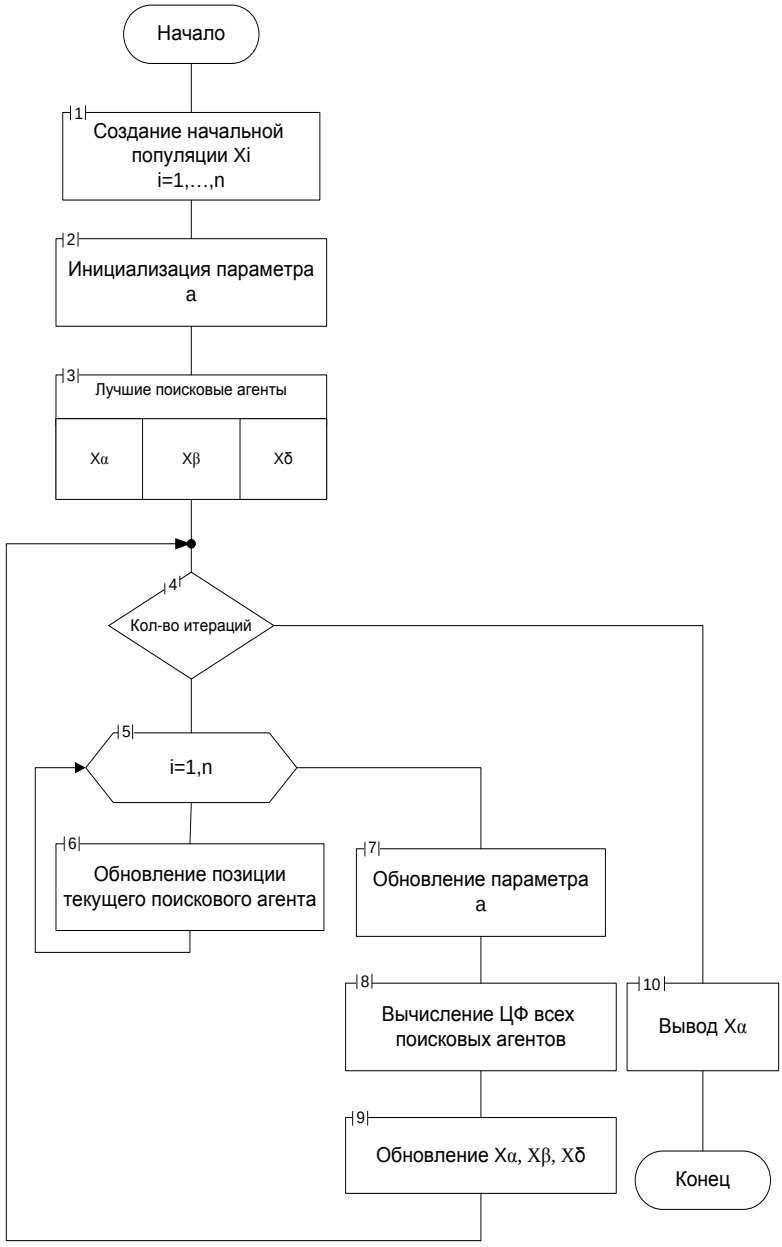

Рис. 2. Схема алгоритма волчьей стаи

В табл. 1 показаны результаты экспериментальных исследований четырех алгоритмов: генетического, муравьиного, пчелиного и комбинированного алгоритмов. Из этих данных можно сделать вывод, что худшие результаты показал генетический алгоритм, хотя он довольно быстрый. Комбинированный же алгоритм показал лучшие результаты, как по времени работы алгоритма, так и по его эффективности.

Таблица 1. **Оценка экспериментальных результатов**

Алгоритмы	Время работы, сек.	Значение ЦФ
ГА	9.553	1285
МА	38.417	1123
ПА	36.62	1063
КА	9.375	915

Согласно данному анализу, легко сделать вывод, что комбинированный алгоритм имеет лучшую точность сходимости, обладая при этом более высокой способностью выходить из локального оптимума. Очевидно, что он превосходит большинство других методов для решения задач оптимизации разрешения элементов СБИС.

Определение временной сложности комбинированного алгоритма. Одной из целей проведения экспериментов было установить значение временной сложности алгоритма (ВСА). Для этого эксперименты проводились над схемами СБИС с различным количеством размещаемых элементов, от 10000 до 100000.

В данной работе временная сложность рассчитывается, исходя из зависимости времени работы алгоритмов от количества размещаемых ими элементов.

Полученные в ходе проведенных экспериментов данные представлены в табл. 2.

Таблица 2. **Оценка временной сложности алгоритмов**

Алгоритм	Количество элементов								
	100	200	300	400	500	600	700	800	900
Генетический (с)	1,05	2,57	3,78	5,01	7,15	9,01	12,6	16,5	18,7
Муравьиный (с)	3,58	8,45	15,56	24,78	35,7	49,9	58,9	69,8	85,7
Комбинированный (с)	2,4	5,12	7,82	11,1	18,5	20,9	29,8	35,6	40,9

Для более наглядного отображения также представлен график зависимости времени работы алгоритмов от числа элементов (рис. 3).

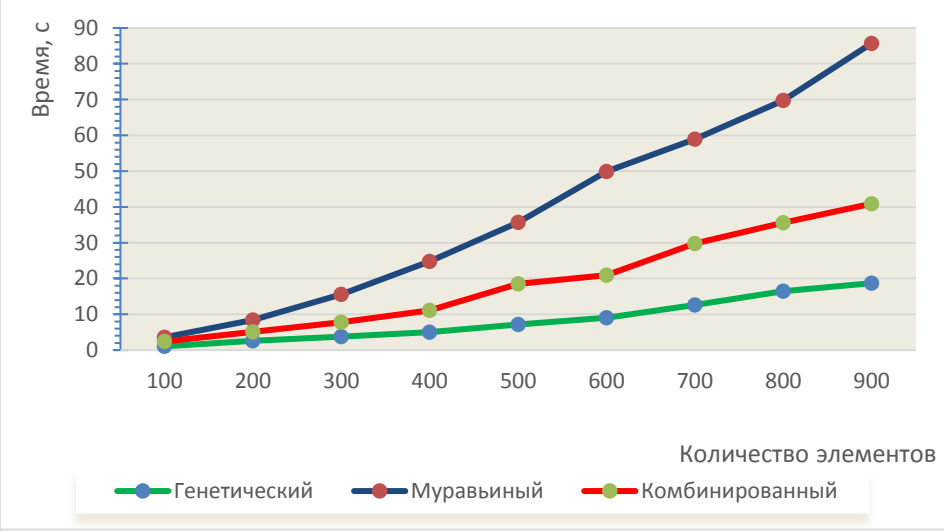

Рис. 3. График зависимости времени работы от количества элементов

На представленном графике, полученном на основе экспериментальных данных, отлично прослеживается квадратичная временная сложность алгоритма. То есть, можно сделать вывод, что ВСА разработанного комбинированного алгоритма равна $O(n^2)$.

Заключение. В данной работе была определена постановка задачи размещения элементов СБИС, а также сформулирован критерий оптимизации. Для решения поставленной задачи был предложен комбинированный биоинспирированный алгоритм, ориентированный на решение задачи размещения элементов СБИС. Разработанный биоинспирированный алгоритм на основе использования эволюционной модели поведения стаи серых волков позволяет эффективно управлять процессом поиска и получать оптимальные и квазиоптимальные результаты за полиномиальное время.

Для проведения экспериментальных исследований была разработана программная среда, реализующая разработанный алгоритм. Очевидно, что при фиксированных значениях параметров разработанный алгоритм имеет полиномиальную теоретическую временную сложность и, следовательно, является эффективным при решении трансвычислительных задач большой размерности.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Гладков Л.А., Курейчик В.В., Курейчик В.М., Сороколетов П.В*. Биоинспирированные методы в оптимизации. – М.: Физматлит, 2009. – 384 с.
2. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы. – М.: Физматлит, 2010. – 368 с.
3. *Курейчик В.В., Курейчик В.М., Зинченко Л.А.* Бионические информационные системы и их практические применения. – М.: Физматлит, 2011.
4. *Норенков И.П.* Основы автоматизированного проектирования. – М.: Изд-во МГТУ имени Н.Э. Баумана, 2000. – 360 с.
5. *Курейчик В.М., Курейчик В.В., Родзин С.И., Гладков Л.А.* Основы теории эволюционных вычислений. – Ростов-на-Дону: Южный федеральный университет, Технологический институт, 2010.
6. *Запорожец Д.Ю., Заруба Д.В., Лежебоков А.А.* Об одном способе кодирования решения для задачи размещения // Известия ЮФУ. Технические науки. – 2012. – № 11 (136). – С. 183-188.
7. *Курейчик В.В., Заруба Д.В., Запорожец Д.Ю.* Алгоритм параметрической оптимизации на основе модели поведения роя светлячков // Известия ЮФУ. Технические науки. – 2015. – № 6 (167) – С. 6-15.
8. *Курейчик В.В., Заруба Д.В., Запорожец Д.Ю.* Иерархический подход при размещении компонентов СБИС // Известия ЮФУ. Технические науки. – 2014. – № 7 (156). – С. 75-84.
9. *Кулиев Э.В., Курейчик В.В., Курситыс И.О.* Принятие решений в задаче размещения компонентов сбис на основе модели поведения стаи волков // Международная конференция по мягким вычислениям и измерениям. – 2018. – Т. 1. – С. 712-715.
10. *Кулиев Э.В., Щеглов С.Н., Пантелюк Е.А., Кулиева Н.В.* Адаптивный алгоритм стаи серых волков для решения задач проектирования // Известия ЮФУ. Технические науки. – 2017. – № 7 (192). – С. 28-38.
11. *Курейчик В.В., Заруба Д.В., Запорожец Д.Ю.* Биоинспирированный алгоритм компоновки блоков ЭВА на основе модифицированной раскраски графа // Известия ЮФУ. Технические науки. – 2015. – № 4 (165). – С. 6-14.

Статью рекомендовал к опубликованию д.т.н., профессор Ю.А. Гатчин.

Запорожец Дмитрий Юрьевич
Федеральное государственное автономное образовательное учреждения высшего образования «Южный федеральный университет».
e-mail: elpilasgsm@gmail.com.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 8(8634)371-651.
Кафедра систем автоматизированного проектирования, доцент каф. САПР.

Zaporoghetz Dmitri Yurievich
Federal State-Owned Educational Establishment of Higher Education “Southern Federal University”.
E-mail: elpilasgsm@gmail.com.
44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: +7(8634)371-651.
The Department of Computer Aided Design; associate professor.