Раздел III. Эволюционное моделирование

УДК 004.896 DOI 10.18522/2311-3103-2020-2-146-157

Б.К. Лебедев, О.Б. Лебедев, Е.О. Лебедева

ЭВОЛЮЦИОННЫЙ АЛГОРИТМ РАЗБИЕНИЯ МЕТОДОМ КРИСТАЛЛИЗАЦИИ РОССЫПИ АЛЬТЕРНАТИВ*

Работа алгоритма разбиения базируется на использовании коллективной эволюционной памяти, под которой подразумевается информация, отражающая историю поиска решения и хранится независимо от индивидуумов. Алгоритм, связанный с эволюционной памятью, стремится к запоминанию и многократному использованию способов достижения лучших результатов. Коллективная эволюционная память алгоритма разбиения состоит из некоторого количества статистических индикаторов, отображающих для каждого выполненного варианта число θ его вхождений в состав лучших решений на выполненных генерациях алгоритма и число, δ определяющее насколько полезна реализованная альтернатива при формировании результатов на прошлых генерациях алгоритма. Коллектив не имеет централизованного управления, и в связи с этим используется непрямой обмен информацией. Непрямой обмен состоит в выполнении неких действий, в различное время, при которых происходит изменение некоторых частей эволюционной памяти одним агентом. В дальнейшем происходит использование этой измененной информации другими агентами, в этих частях. Вначале на каждой итерации конструктивным алгоритмом формируется n_k решений Q_k. Каждое решение Q_k является отображением $F_k=V\rightarrow X$, представляется в виде двудольного подграфа D^k и формируется путем последовательного назначения элементов в узлы. Формирование каждого решения Q_k выполняется множеством агентов A, посредством вероятностного выбора каждым агентом a_i узла v_j. Процесс назначения элемента в узел включает две стадии. На первой стадии выбирается агент a_i, а на второй стадии − узел v_j. При этом должно выполняться ограничение: каждому агенту множества A соответствует один единственный узел множества V. Рассчитывается оценка ξ_k решения Q_k и оценка полезности δ_k множества альтернатив, реализованных агентами в решении Q_k. На втором этапе агенты увеличивают в интегральной россыпи альтернатив R^* интегральную полезность множества альтернатив на величину δ_k. На третьем этапе осуществляется снижение оценок полезности δ_k интегральной россыпи альтернатив на величину μ. В работе используется циклический метод формирования решений. В этом случае наращивание оценок интегральной полезности δ_k множества позиций P выполняется после полного формирования множества решений Q на итерации l. Экспериментальные исследования проводились на основе сформированных тестовых примеров с полученным ранее оптимальным решением. Полученные результаты сравнивались с результатами полученными другими известными алгоритмами разбиения схем на части. Для сравнения был сформирован набор стандартных бенчмарок. Проанализировав полученные результаты, можно сделать вывод, что предложенный метод позволяет получать на 4–5 % решения качественнее, чем его аналоги.

Роевой интеллект; адаптация; кристаллизация; СБИС; разбиение; россыпь альтернатив; двудольный граф.

* Работа выполнена при финансовой поддержке гранта РФФИ № 20–07–00260 А.

B.K. Lebedev, O.B. Lebedev, E.O. Lebedeva

EVOLUTION ALGORITHM FOR PARTITION BY METHOD OF CRYSTALLIZATION OF ALTERNATIVES FIELD

The operation of the partitioning algorithm is based on the use of collective evolutionary memory, which means information that reflects the history of the search for a solution and is stored independently of individuals. The algorithm associated with evolutionary memory seeks to memorize and reuse ways to achieve better results. The collective evolutionary memory of the partitioning algorithm is a set of statistical indicators that reflect, for each implemented alternative, the number θ of its occurrences in the best solutions at previous iterations of the algorithm and the number δ indicating the usefulness of the implemented alternative when constructing solutions at previous iterations of the algorithm. The team does not have centralized management, and its features are the presence of indirect exchange of information. Indirect exchange consists in performing certain actions, at different times, during which some parts of evolutionary memory change by one agent. In the future, this changed information is used by other agents in these parts. First, at each iteration, a constructive algorithm generates n_k solutions Q_k. Each solution Q_k is a mapping $F_k=V\rightarrow X$, is represented as a bipartite subgraph D^k and is formed by sequentially assigning elements to nodes. The formation of each solution Q_k is performed by the set of agents A, by means of the probabilistic choice by each agent a_i of the node v_j. The process of assigning an element to a node involves two stages. In the first stage, agent a_i is selected, and in the second stage, the node. In this case, the restriction must be fulfilled: each agent of the set A corresponds to one unique node of the set V. The estimate ξ_k of the solution Q_k and the utility estimate δ_k of the set of alternatives implemented by the agents in the solution Q_k are calculated. At the second stage, the agents increase the integral utility of the set of alternatives in the integral placer of alternatives R^* by the value δ_k. At the third stage, the utility estimates δ_k of the integral placer of alternatives are reduced by μ. The paper uses the cyclic method of forming decisions. In this case, the building up of estimates of the integral utility δ_k of the set of positions P is performed after the complete formation of the set of solutions Q at iteration l. Experimental studies were carried out on the basis of formed test cases with the optimal solution obtained earlier. The results obtained were compared with the results obtained by other well-known algorithms for dividing circuits into parts. For comparison, a set of standard benchmarks was formed. After analyzing the results, we can conclude that the proposed method allows you to get 4–5 % better solutions than its analogues.

Swarm intelligence; adaptation; crystallization; VLSI; partitioning; placer alternatives; dicotyledonous graph.

Введение. При проектировании топологии СБИС выполняется ряд этапов связанных с решением оптимизационных задач. Одной из таких задач является задача разбиения схем на части. Поиск оптимального решения задачи разбиения схем на части, имеет важное значение, т.к. от найденного результата напрямую зависит качество выполнения другой задачи проектирования – трассировки межсоединений, что приводит к эффективной работоспособности всей схемы [1–2]. Задача разбиения – это класс *NP*-полных задач. В задачах такого класса невозможно определить решение точными алгоритмами [1, 3, 4]. К методам, позволяющим решать оптимизационные задачи такого типа, можно отнести алгоритмы с использованием процедур роевого интеллекта, например метод моделирования поведения муравьиной колонии [4–10].

В работе используется метаэвристика [10], учитывающая тенденцию к использованию альтернатив (вариантов компонентов) из наилучших найденных решений. Особенностями являются наличие непрямого обмена информацией – стигмержи [11]. Совокупность данных об альтернативах и их оценках составляет россыпь альтернатив. Рассмотрены ключевые моменты анализа альтернатив в процессе эволюционной коллективной адаптации [12, 13]. В процессе эволюционной коллективной адаптации методами дискриминантного анализа формируются оценки приспособленности альтернатив. Приспособленность альтернатив рас-

сматривается как вероятность ее использования в формируемом решении. Дискриминантный анализ и поиск эффективных альтернатив, в процессе эволюционной коллективной адаптации назван по аналогии с процессами вычленения объектов (формирования кристаллов) кристаллизацией. Другими словами, в процессе эволюционной коллективной адаптации производится вычленение из множества вариантов наиболее приспособленных альтернатив. Отсюда название метода оптимизации – метод кристаллизации россыпи альтернатив (КРА).

Достоинство предложенного подхода состоит в возможности определять решение высокого качества, для задач большой размерности. Методика вычисления целевой функции отличается простотой, что позволяет легко оперировать с видоизмененными решениями в процессе поиска.

1. Постановка задачи разбиения. Пусть дан граф $G(X, U)$, где X – множество вершин, $|X|=n$, U – множество ребер. Необходимо разбить множество X на m непустых и непересекающихся подмножества X_j, $\cup X_j=X$, $X_i \cap X_j=\varnothing$, $X_j \neq \varnothing$. На формируемые узлы накладываются ограничения: $|X_j|=n_j$, $\sum n_j=n$. Будем считать, что множество вершин X_i помещается в узел v_i. Критерий оптимизации – суммарное число связей D между узлами. Цель оптимизации – минимизация критерия D [14–16].

Решить задачу разбиения равносильно построению всюду определенного, сюръективного и инъективного соответствия $F=V \rightarrow X$, где X – множество вершин множество вершин, а V – узлов.

Соответствие $F=V \rightarrow X$ представляется в виде двудольного графа $H=(V \cup X,E)$, где $V=\{v_v|v=1,2,...,m\}$ – множество вершин (первая доля), соответствующих множеству узлов $V=\{v_j|j=1,2,...,m\}$, а $X=\{x_i|i=1,2,...,n\}$ – множество вершин (вторая доля), соответствующих множеству вершин графа $G(X,U)$, E – множество ребер (x_i, v_j) связывающих вершины $x_i \in X$ с вершинами множества V. $|E|=nm$, рис. 1.

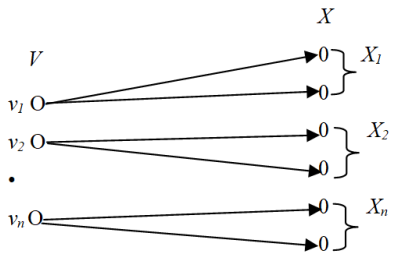

Рис. 1. Модель задачи разбиения

Образом узла v_j является множество вершин $X_j \subset X$, включенных в узел v_j, $F(v_j)=\{X_j\}$. Прообразом вершины $x_i \in X_j$ является узел v_j, $F^{-1}(v_j)=\{x_i\}$.

2. Разбиение схем на части с использованием метода кристаллизации россыпи альтернатив. В методе КРА [10–14] каждый результат Q_k синтезируется множеством представитель коллектива a_i отражает некоторое количество возможных состояний $C_i=\{c_{ij}|j=1,2,...,s\}$. Каждый a_i может быть в одном из возможных состояний. Решение Q_k будет сформировано набором возможных состояний множества представителей коллектива.

Представление данной задачи полагает, что членами сообщества (агентами) выступают вершины графа, а альтернативами – множество узлов $V=\{v_j|j=1,2,...,m\}$, где m – число узлов, в которые помещаются вершины. Можно сказать, что члены сообщества $A=\{a_i|i=1,2,...,n\}$, сопоставляются множеству всех вершин. Варианты возможных состояний $C_i=\{c_{ij}|j=1,2,...,s\}$ члена сообщества a_i сопоставляются множеству возможных вариантов распределения агента a_i. Вариант c_{ij} соответствует вхождению агента a_i в узел v_j [8].

В качестве графа поиска решений используется полный двудольный граф $H=(V \cup X,E)$, где $X=\{x_i|i=1,2,\ldots,n\}$ – множество вершин (первая доля), соответствующих множеству вершин графа $H(X,U)$, а $V=\{v_j|j=1,2,\ldots,m\}$, – множество вершин (вторая доля), соответствующих множеству узлов $|V|=m$. E – множество ребер (x_i, v_j) связывающих вершины $x_i \in X$ с вершинами множества V. $|E|=nm$.

Задача алгоритма разбиения сводится к поиску на полном двудольном графе $H=(V \cup X,E)$ двудольного подграфа $H^k=(V \cup X,E^k)$, $E^k=\{E^k_j \ |j=1,2,\ldots,m\}$ для которого оценка D имеет минимальное значение. Двудольный подграф $H^k=(V \cup X,E^k)$, обладает ограничениями:

- число ребер множества E^k_j, инцидентных вершине v_j, равно числу вершин множества X^k_j, $|X^k_j|=n_j$;
- каждое ребро $(x_i,v_j)=e^k_{ij} \in E^k_j$, с одной стороны, инцидентно вершине $v_j \in V$, с другой стороны – инцидентно одной и только одной вершине $x_i \in X_j$; $\cup X_j=X$;
- локальная степень любой вершины $x_i \in X$ равна единице, $\rho(x_i)=1$;
- локальная степень вершины v_j равна мощности множества X_j, $\rho(v_j)=|X_j|$.

Представим структуру данных для отображения одного решения Q_k, формируемого множеством агентов A, в виде матрицы $R_k=||r_{kji}||_{m \times n}$. Столбцы матрицы соответствуют агентам, строки – узлам, в которые может назначаться агент (состояниям агентов).

Размерность строки (вектора $R_{kj}=\{r_{kji}|i=1,2,\ldots,n\}$) определяется числом n агентов. Размерность столбца (вектора $R_{ki}=\{r_{kji}|j=1,2,\ldots,m\}$) – число m альтернатив состояний агента a_i.

Каждый, отличный от нуля элемент r_{kji} матрицы $R_k=||r_{kji}||_{m \times n}$, имеет значение, равное полезности δ_k решения, при котором агент a_i назначен в узел v_j. $\delta_k=f(\xi_k)$, где ξ_k – оценка решения Q_k, а δ_k – оценка полезности этого решения.

Пусть для решения задачи разбиения построено множество решений $Q=\{Q_k|k=1,2,\ldots,n_k\}$. Для каждого решения Q_k рассчитаны оценки ξ_k и δ_k и сформирована индивидуальная россыпь альтернатив R_k.

Представим структуру данных для отображения 6 решений Q_1-Q_k, сформированных множеством агентов A, в виде совокупности матриц R_1-R_4, которую назовем россыпью альтернатив (РА).

Пример. Пусть имеется решение Q_k синтезированное десятью агентами. Агенты имеют 3 варианта состояния. В решении Q_k агентами реализовано разбиение множества вершин X, $|X|=n=10$ на 3 подмножества – X_1, X_2, X_3: $|X_1|=4$,$|X_2=3|$, $|X_3|=3$. Решения имеют следующие оценки полезности: $\delta_1=4$, $\delta_2=6$, $\delta_3=2$, $\delta_4=8$.

Россыпь альтернатив для решений R_k имеет вид, представленный на рис. 2.

Множество индивидуальных россыпей альтернатив $R=\{R_k|k=1,2,\ldots,n_k\}$ формируется на базе сгенерированного множества решений $Q=\{Q_k|k=1,2,\ldots,n_k\}$.

Платформой для организации эволюционной процедуры поиска результата выступает интегральная россыпь альтернатив (ИРА) – R^*, получаемая при учете всех россыпей альтернатив:

$$R^*=||r^*_{ji}||_{m \times n}, \text{ где } r^*_{ji}=\sum_k(r_{kji}),\ k=\{1,n_k\}.$$

По сути r^*_{ji} – это общая оценка полезностей результатов, в которых агентом a_j был выполнен вариант c_{ji}. Интегральная россыпь альтернатив используется в качестве коллективной эволюционной памяти и служит базой для формирования новых решений.

R_1=

	агенты											
блоки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	n_j	
v_1	4	0	0	4	0	0	4	0	4	0	4	x_1, x_4, x_7, x_9
v_2	0	4	4	0	0	0	0	0	0	4	3	x_2, x_3, x_{10}
v_3	0	0	0	0	4	4	0	4	0	0	3	x_5, x_6, x_8

R_2=

	агенты											
блоки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	n_j	
v_1	6	0	6	0	0	0	0	6	0	6	4	x_1, x_3, x_8, x_{10}
v_2	0	6	0	0	6	0	6	0	0	0	3	x_2, x_5, x_7
v_3	0	0	0	6	0	6	0	0	6	0	3	x_4, x_6, x_9

R_3=

	агенты											
блоки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	n_j	
v_1	2	2	0	0	0	0	2	0	0	2	4	x_1, x_2, x_7, x_{10}
v_2	0	0	2	2	0	0	0	0	2	0	3	x_3, x_4, x_9
v_3	0	0	0	0	2	2	0	2	0	0	3	x_1, x_3, x_8, x_{10}

R_4=

	агенты											
блоки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	n_j	
v_1	8	0	8	0	0	8	0	8	0	0	4	x_1, x_3, x_6, x_8
v_2	0	8	0	8	8	0	0	0	0	0	3	x_2, x_4, x_5
v_3	0	0	0	0	0	0	8	0	8	8	3	x_3, x_9, x_{10}

Рис. 2. Россыпи альтернатив R_k

Пример. На рис. 3. представлена интегральная россыпь альтернатив, R^*, объединяющая россыпи альтернатив $R_1 - R_4$

R^*=

	агенты												
блоки	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	n_j	max	min
v_1	20	2	14	4	0	8	6	14	4	8	4	20	2
v_2	0	18	6	10	14	0	7	0	2	10	3	18	2
v_3	0	0	0	6	6	12	8	6	14	8	3	14	6

Рис. 3. Интегральная россыпь альтернатив R^*

Лучшее значение интегральной оценки полезности решений $f^{\#}$=20, худшее значение f^0=2. Выбирается параметр ε, лежащий в границах $f^0 \le \varepsilon \le f^{\#}$. Пусть ε =1.

Работа алгоритма разбиения базируется на использовании коллективной эволюционной памяти (КЭП), под которой подразумевается информация, отражающая историю поиска решения, и хранится независимо от индивидуумов. Алгоритм, связанный с эволюционной памятью, стремится к запоминанию и многократному использованию способов достижения лучших результатов. Коллективная эволюционная память алгоритма разбиения – это набор индикаторов, показывающих для каждой альтернативы число θ ее вхождений в наилучшие решения на предшествующих этому генерациях работы алгоритма и число δ отражающее полезность реализованной альтернативы при формировании результата на прошлых генерациях работы алгоритма. В основе такой системы лежит процесс самоорганизации, который позволяет получить общий результат коллектива, при помощи низкоуровневых совместных действий. У данного сообщества не имеется центрального управления, в связи, с чем присутствует непрямой обмен информацией. Такой обмен информацией, выглядит как распределенные во времени взаимные действия, т.е. такие действия, когда один из членов сообщества производит изменения в каких-то областях ко-эволюционной памяти, а другие члены этого сообщества в дальнейшем применяют эту видоизмененную информации содержащуюся в областях [8, 9].

Процесс поиска решений итерационный. На первом этапе каждой итерации конструктивным алгоритмом формируется n_k решений Q_k,. Каждое решение Q_k является отображением $F_k=V\rightarrow X$, представляется в виде двудольного подграфа D^k и формируется путем последовательного назначения элементов в узлы. Формирование каждого решения Q_k выполняется множеством агентов A, посредством вероятностного выбора каждым агентом a_i узла v_j. *Процесс* назначения элемента в узел включает две стадии. На первой стадии выбирается агент a_i, а на второй стадии − узел v_j. При этом должно выполняться ограничение: каждому агенту множества A соответствует один единственный узел множества V. Рассчитывается оценка ξ_k решения Q_k и оценка полезности δ_k множества альтернатив, реализованных агентами в решении Q_k. На втором этапе агенты увеличивают в интегральной россыпи альтернатив R^* интегральную полезность множества альтернатив на величину δ_k,. На третьем этапе осуществляется снижение оценок полезности δ_k интегральной россыпи альтернатив на величину μ. В работе используется циклический метод формирования решений. В этом случае наращивание оценок интегральной полезности δ_k множества позиций P выполняется после полного формирования множества решений Q на итерации l [12, 13].

После создания начальной ИРА некоторые полезности вариантов могут быть равны нулю. Следовательно, вероятность того, что эти варианты будут выбраны, также равна нулю. Следовательно, такие варианты сразу будут «отброшены» из списка рассматриваемых вариантов и не будут участвовать в процессе поиска решений. Для того, чтобы такие варианты не были удалены из рассматриваемого списка, выполняется корректировка начальной ИРА. Корректировка заключается в присвоении элементам имеющим значение «ноль», значения $f^0\leq\varepsilon\leq f^{\#}$.

3. Алгоритм разбиения методом кристаллизации россыпи альтернатив. Введем некоторые обозначения:

$A(t)$ − множество не размещенных в узлах элементов на шаге t.

$\rho_j(t)$ − локальная степень узла v_j на шаге t.

$V^*(t)\subset V)$ − множество узлов на шаге t таких, что для каждого узла $v_j\in V^*(t)$ его текущая локальная степень $\rho_j(t)< n_j$.

Представим алгоритм разбиения методом кристаллизации россыпи альтернатив.

1. Задается объем обучающей популяции решений – n_k.

2. Вначале формируется популяция решений $Q=\{Q_k\ |k=1,2,\ldots,n\}$ с помощью самопроизвольного определения вариантов c_{ij}. Вычисление параметров ξ_k и δ_k для всех решений Q_k популяции Q.

3. Уменьшение сформированной популяции решений Q к начальному размеру $|Q|=n_k$ с помощью выбора n_k наилучших результатов (с наибольшей обозначенной оценкой полезности δ_k).

Поиск в полученной популяции Q решения $Q^{\#}$, имеющего наилучшую оценку $\xi^{\#}$ и решения Q^0, имеющего наихудшую оценку – ξ^0.

4. Синтез исключительной россыпи альтернатив R_k для каждого решения $Q_k\in Q$.

5. Синтез вводной интегральной россыпи альтернатив $R^*(1)$ при помощи сведения в совокупность всех россыпей альтернатив R_k:

$r^*_{ji}(1)=\sum_k(r_{kji})$, $k=\{1,n_k\}$, $\quad R^*(1)=||r^*_{ji}||_{m\times n}$.

6. Корректировка вводной интегральной россыпи альтернатив $R^*(1)$ при помощи возрастания элементов r^*_{ji} с нулевым значением на величину ε, $f^0\leq\varepsilon\leq f^{\#}$.

7. Задается число итераций – n_l. $l=1$. *(l – номер итерации).*

8. $k=1$. *(k – номер решения).*

9. Обнуление всех россыпей альтернатив R_k,. $t=1$. *(t – номер шага).*

10. Приводится в начальное состояние служебная память системы: $A(t)=A$, $\forall j(\rho_j(t)=0$. (*Отметим, что $\rho_j(t)$ растет с ростом t*). $V(t)=V$.

11. Первая стадия. На первой стадии шага t для каждого $a_i \in A(t)$ формируется множество $V^*(t)$ узлов таких, что для каждого узла $v_j \in V^*(t)$ его текущая локальная степень $\rho_j(t)<n_j$.

12. Для каждого $a_i \in A(t)$ среди узлов множества $V^*(t)$ отыскивается узел с максимальным значением полезности назначения в него a_i, которое назовем стоимостью $\pi_i(t)$ агента a_i на шаге t.

13. Среди $a_i \in A(t)$ с вероятностью $P_{ik}(t)=\pi_i(t)/\sum_i(\pi_i(t))$, пропорциональной стоимости $\pi_i(t)$ агента a_i выбирается агент a^*_i.

14. Вторая стадия. Среди множества узлов $V^*(t)$, доступных на шаге t агенту a^*_i, с вероятностью – $P_{ij}(t)=r_{ji} / \sum_j (r_{ji})$ $(j|v^*_j \in V^*(t))$, пропорциональной полезности r_{ji} назначения a^*_i в узел v^*_j, выбирается узел $v^*_j \in V^*(t)$.

15. Агент a^*_i назначается в узел v^*_j.

16. $t=t+1$ локальная степень узла $v^*_j \in$ увеличивается на 1, $\rho_j(t)= \rho_j(t-1)+1$; a^*_i исключается из $A(t-1)$, $A(t)=A(t-1)\backslash a^*_i$.

17. Если все элементы a_i назначены в узлы, т.е. $A(t)=\varnothing$, то переход к 16, иначе переход к 9.

18. Расчет оценок ξ_k решения Q_k, $\delta_k=f(\xi_k)$.

19. Запоминание лучшей оценки. Если оценка ξ_k решения Q_k лучше оценки $\xi^{\#}$ решения $Q^{\#}$, то $\xi^{\#}=\xi_k$, а $Q^{\#}=Q_k$.

20. Формируется индивидуальная россыпь альтернатив R_k для Q_k.

21. Если $(k<n_k)$ то $k=k+1$ и переход к 9, иначе переход к 22.

22. Если $(l<N_l)$, то $l=l+1$ и переход к 23, иначе переход к 25.

23. Суммирования оценок полезности интегральной россыпи альтернатив $R(l-1)^*$, построенной на предыдущей итерации $(l-1)$, с соответствующими оценками полезности всех индивидуальных россыпей альтернатив R_k, сформированных на итерации l.

24. Снижение всех интегральных оценок полезности r^*_{ij} интегральной россыпи альтернатив $R^*(l)$на величину δ^*.

25. Завершение работы алгоритма. Фиксация и вывод лучшего решения $Q^{\#}$.

Чтобы увеличить вероятность выбора альтернативы, разработана (модификация) следующая методика. В ИРА отыскивается вариант имеющий минимум $(r^*_{ij})_{min}$. Отыскивается параметр $q<(r^*_{ij})_{min}$. После модификации формула примет вид:

$$p_{ij} = (rr^*_{ij} - q)/(\sum_j (r^*_{ij} - q)).$$

4. Проведение тестовых испытаний. Создана программа Crystallization of alternatives field partitioning (CAFP), в основу работы которой заложены новые разработанные метаэвристики, позволяющие находить оптимальные решения.

Экспериментальные исследования проводились на основе сформированных тестовых примеров с полученным ранее оптимальным решением. Полученные результаты сравнивались с результатами полученными другими известными алгоритмами разбиения схем на части. Для сравнения был сформирован набор стандартных бенчмарок. Проанализировав полученные результаты, можно сделать вывод, что предложенный метод позволяет получать на 4–5 % решения качественнее, чем его аналоги [14, 17–20].

Для исследования синтезировались примеры, содержащие до 1000 вершин. Вес вершин и вес всех рёбер принимался равным единице. При этом графы «разбивались» на подграфы с разным количеством вершин в каждом подграфе.

В качестве оценки качества используется безразмерная величина $F_{опт}/F$, где F – оценка решения полученного, полученного с помощью разработанного алгоритма. Выполнив тестовые испытания и проведя их анализ, построена графическая зависимость, представленная на рис.4 и отражающая процесс получения качест-

венного решения от количества выполненных генераций. Также, на рис. 5 приведена графическая зависимость, отражающая процесс получения качественного решения от размера популяции. Установлено что начальное количество оценки полезности должно быть в 16 раз больше среднего количества значения оценки полезности, добавляемого в интегральную россыпь альтернатив на каждой итерации. Коэффициент обновления интегральной россыпи альтернатив ρ=0,88.

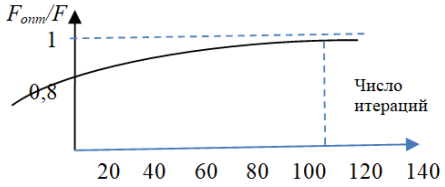

Рис. 4. Зависимость качества решений алгоритма CAFP от числа итераций

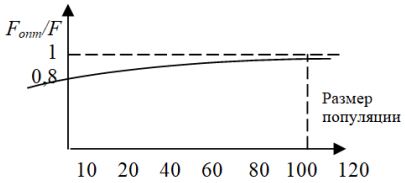

Рис. 5. Зависимость качества решений алгоритма CAFP от размера популяции

В ходе тестовых испытаний определено, что имея множество решений равным сто, схождение разработанного алгоритма разбиения схем на части, происходит примерно на сто десятой генерации. Отметим, что у ряда экспериментов число итераций меньше среднего составляло до 8 %, а больше с среднего до 30 %. Временная сложность процедуры на одной выполняемой генерации алгоритма равна *O(n)*.

Сравнение полученных результатов алгоритма разбиения схем на части, с разработанной метаэвристикой интеллектуальной оптимизации, и известными в литературе другими алгоритмами разбиения, осуществлялся с помощью набора стандартных бенчмарок [14–16]. Эти тестовые бенчмарки являются используемыми в промышленности фрагментами схем, отдельными блоками, сверхбольшими интегральными схемами и т.д. Параметрами, используемыми для сравнения выступали:

- Cut – количество межсоединений, оказавшихся в разрезе после разбиения схемы на части;
- CPU –время работы, требуемое для нахождения результата.

Выполнена сравнительная оценка работы разработанных алгоритмов на бенчмарках cordic(881), misex3(1349), X3(1369). c6288(2435) s15850(4321), frisc(4400), elliptic(4711), (в скобках приведено число вентилей) с оценками известных из литературы алгоритмов (оценки получены на тех же бенчмарках): Pure hMetis [15]; Net&Path Based [16]; PPFF [16]; VPR [15]. Результаты показаны в табл. 1.

Проведя тестовые испытания и выполнив анализ полученных решений, можно сказать, что алгоритм разбиения схем на части с встроенной в него разработанной новой метаэвристикой интеллектуальной оптимизации, позволяет получать результаты не хуже, а в ряде случаев лучше имеющихся известных алгоритмов на 4–5 %.

Например, разработанный алгоритм дает значительно лучшие результаты, чем hMetis. Проведенный тест схемы Cordic показал, что решение, полученное в результате выполнения алгоритмом CAFP, лучше на 5,1 % чем решение, полученное в результате выполнения алгоритмом hMetis, который считается алгоритмом позволяющим находить оптимальные решения.

Таблица 1

Сравнительная оценка работы алгоритмов

		Cordic	**misex3**	**X3**	**c6288**	**s15850**	**Frisc**	**Elliptic**
Pure hMetis	**Cut**	327	543	211	182	617	838	508
	CPU	2	5	4	31	28	34	16
Net& Path	**Cut**	278	614	261	216	671	869	619
	CPU	5	13	6	80	37	61	22
PPFF	**Cut**	313	540	228	201	615	851	510
	CPU	6	14	8	79	43	66	25
VPR	**Cut**	319	543	216	204	622	843	513
	CPU	10	18	13	85	47	71	27
CAFP	**Cut**	270	600	251	207	621	812	517
	CPU	4	11	6	74	300	46	14

Заключение. В работе выполнен обзор известных в литературе алгоритмов, применяемых для решения задачи разбиения схем на части. Проведя анализ этих алгоритмов, выбран метод интеллектуальной оптимизации, на основе которого строился представленный алгоритм.

Разработан новый способ решения оптимизационных задач с использованием интеллектуальных процедур. Этот способ заключается в применении выбранных лучших из найденных вариантов решений. В основу этого метода положено построение моделей коллективного интеллекта. Набор данных о вариантах реализации и их целевых функциях, является совокупностью вариантов. Идея метода заключается в следующем. Каждое решение строится агентами, при этом у агента имеется множество альтернатив состояний или же он находится в какой то конкретной альтернативе. Формирование решения происходит на основе возможных вариантов состояний всех агентов.

Выполнен анализ различных вариантов состояний, возникающих в ходе коэволюционной адаптации. Этот процесс назван кристаллизацией, т.к. является аналогом вычленения объектов (формирования кристаллов). Разработанный метод достаточно эффективен при выполнении поиска решений в оптимизационных задачах, особенно в тех задачах, где возможно их представление в виде россыпи альтернатив.

Проведенные тестовые испытания позволили сделать вывод, что применяя алгоритмы на основе разработанного метода, для решения оптимизационных задач, полученные результаты дают улучшение целевой функции до 5%. Временная зависимость работы алгоритма с встроенной в него разработанной метаэвристикой, находится в интервале $O(n^2)$-$O(n^3)$.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Немудров В., Мартин Г.* Системы-на-кристалле. Проектирование и развитие. – М.: Техносфера, 2004. – 157 с.
2. *Казеннов Г.Г.* Основы проектирования интегральных схем и систем. – М.: Бином. Лаборатория знаний, 2005. – 295 с.
3. *Alpert C.J., Mehta D.P., Sapatnekar S.S.* Handbook of Algorithms for Physical Design Automation. Boston. – MA: Auerbach, 2009. – 245 p.
4. *Карпенко А.П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учеб. пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2014. – 448 с.

5. *Курейчик В.М., Лебедев Б.К., Лебедев О.Б.* Гибридный алгоритм разбиения на основе природных механизмов принятия решений // Искусственный интеллект и принятие решений. – М.: Изд-во Институт системного анализа РАН, 2012. – С. 3-15.
6. *Лебедев Б.К., Лебедев О.Б.* Муравьиные алгоритмы разбиения, использующие представление задачи, отличные от канонического // Вестник Ростовского государственного университета путей сообщения. – 2016. – № 2 (62). – С. 71-77.
7. *Лебедев Б.К., Лебедев О.Б.* Разбиение на основе многоуровневой параллельной эволюционной адаптации // Проблемы разработки перспективных микро- и наноэлектронных систем – 2008: Сб. научных трудов / под ред. А.Л. Стемпковского. – М.: ИППМ РАН, 2008. – С. 36-41.
8. *Ворбьева Е.Ю., Карпенко А.П., Селиверстов Е.Ю.* Ко-гибридизация алгоритмов роя частиц // Наука и жизнь. – 2012. – № 4.
9. *Zhou Y., Pei Sh.* A hybrid co-evolutionary particle swarm optimization algorithm for solving constrained engineering design problems // Journal of computers (JCP). – 2010. – Vol. 5, No. 6. – P. 965-972.
10. *Лебедев Б.К., Лебедев В.Б.* Оптимизация методом кристаллизации россыпи альтернатив // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 11-17.
11. *Clerc M.* Particle Swarm Optimization. – ISTE, London, UK, 2006. – 198 p.
12. *Poli R.* Analysis of the publications on the applications of particle swarm optimization // Journal of Artificial Evolution and Applications, Article ID 685175, 2008.
13. *Лебедев Б.К., Лебедев В.Б.* Построение кратчайших связывающих соединений методом кристаллизации россыпи альтернатив // МЭС-2014. VI Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем – 2014»: Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. – М.: ИППМ РАН, 2014. Ч. I. – С. 177-183.
14. *Cong J., Romesis M., Xie M.* Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms // Proc. of the International Symposium on Physical Design. – Monterey, CA, 2003. – P. 88-94.
15. *Selvakkumaran N., Karypis G.* Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization // ICCAD. – 2003. – P. 234-249.
16. *Ababei C., Selvakkumaran N., Bazargan K., Karypis G.* Multi-objective Circuit Partitioning for Cutsize and Path-Based Delay Minimization // ICCAD. – 2002. – P. 118-137.
17. *Агасиев Т.А., Карпенко А.П.* Современные техники глобальной оптимизации // Информационные технологии. – 2018. – № 6. – С. 370-386.
18. *Норенков И.П., Уваров М.Ю.* Поддержка принятия решений на основе паттернов проектирования // Наука и образование: научное издание. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2011. – № 9. – С. 25-32.
19. *Божко А.Н.* Методы структурного анализа сложных изделий в интегрированных CFD/CAM-системах // Информационные технологии. – 2018. – № 8. – С. 499-506.
20. *Столяров А.И., Донецкая Ю.В., Гатчин Ю.А.* Особенности САПР при проектировании комплекса // Вестник Иркутского государственного технического университета. – 2018. – № 7 (138). – С. 88-95.

REFERENCES

1. *Nemudrov V., Martin G.* Sistemy-na-kristalle. Proektirovanie i razvitie [Systems-on-a-chip. Design and development]. Moscow: Tekhnosfera, 2004, 157 p.
2. *Kazennov G.G.* Osnovy proektirovaniya integral'nykh skhem i system [Fundamentals of design of integrated circuits and systems]. Moscow: Binom. Laboratoriya znaniy, 2005, 295 p.
3. *Alpert C.J., Mehta D.P., Sapatnekar S.S.* Handbook of Algorithms for Physical Design Automation. Boston. MA: Auerbach, 2009, 245 p.
4. *Karpenko A.P.* Sovremennye algoritmy poiskovoy optimizatsii. Algoritmy, vdokhnovlennye prirodoy: ucheb. posobie [Modern search engine optimization algorithms. Algorithms inspired by nature]. Moscow: Izd-vo MGTU im. N.E. Baumana, 2014, 448 p.
5. *Kureychik V.M., Lebedev B.K., Lebedev O.B.* Gibridnyy algoritm razbieniya na osnove prirodnykh mekhanizmov prinyatiya resheniy [Hybrid partition algorithm based on natural decision-making mechanisms], *Iskusstvennyy intellekt i prinyatie resheniy* [Artificial intelligence and decision making]. Moscow: Izd-vo Institut sistemnogo analiza RAN, 2012, p. 3-15.

6. *Lebedev B.K., Lebedev O.B.* Murav'inye algoritmy razbieniya, ispol'zuyushchie predstav-lenie zadachi, otlichnye ot kanonicheskogo [Ant partitioning algorithms using a task representation other than the canonical], *Vestnik Rostovskogo gosudarstvennogo universiteta putey soobshcheniya* [Bulletin of the Rostov State University of Railways], 2016, No. 2 (62), pp. 71-77.
7. *Lebedev B.K., Lebedev O.B.* Razbienie na osnove mnogourovnevoy parallel'noy evolyutsionnoy adaptatsii [Partitioning on the basis of multi-level parallel evolutionary adaptation], *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh system* [Problems of developing promising micro- and nanoelectronic systems], *2008: Sb. nauchnykh trudov* [Problems of developing promising micro- and nanoelectronic systems – 2008: Collection of scientific papers], ed. by A.L. Stempkovskogo. Moscow: IPPM RAN, 2008, pp. 36-41.
8. *Vorb'eva E.Yu., Karpenko A.P., Seliverstov E.Yu.* Ko-gibridizatsiya algoritmov roya chastits [Co-hybridization of particle swarm algorithms], *Nauka i zhizn'* [Science and Life], 2012, No/ 4.
9. *Zhou Y., Pei Sh.* A hybrid co-evolutionary particle swarm optimization algorithm for solving constrained engineering design problems, *Journal of computers (JCP)*, 2010, Vol. 5, No. 6, pp. 965-972.
10. *Lebedev B.K., Lebedev V.B.* Optimizatsiya metodom kristallizatsii rossypi al'ternativ [Optimization by the method of crystallization of alternatives field], *Izvestiya YUFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2013, No. 7 (144), pp. 11-17.
11. *Clerc M.* Particle Swarm Optimization. ISTE, London, UK, 2006, 198 p.
12. *Poli R.* Analysis of the publications on the applications of particle swarm optimization, *Journal of Artificial Evolution and Applications, Article ID 685175, 2008*.
13. *Lebedev B.K., Lebedev V.B.* Postroenie kratchayshikh svyazyvayushchikh soedineniy metodom kristallizatsii rossypi al'ternativ [Construction of the shortest binding compounds by the method of crystallization of alternatives field], *MES-2014. VI Vserossiyskaya nauchno-tekhnicheskaya konferentsiya «Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem – 2014»: Sb. trudov* [MES-2014. VI All-Russian scientific and technical conference "Problems of developing promising micro- and nanoelectronic systems – 2014": Proceedings], under the total. ed. akademika RAN A.L. Stempkovskogo. Moscow: IPPM RAN, 2014. Part I, pp. 177-183.
14. *Cong J., Romesis M., Xie M.* Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms, *Proc. of the International Symposium on Physical Design*. Monterey, CA, 2003, pp. 88-94.
15. *Selvakkumaran N., Karypis G.* Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization, *ICCAD*, 2003, pp. 234-249.
16. *Ababei C., Selvakkumaran N., Bazargan K., Karypis G.* Multi-objective Circuit Partitioning for Cutsize and Path-Based Delay Minimization, *ICCAD*, 2002, pp. 118-137.
17. *Agasiev T.A., Karpenko A.P.* Sovremennye tekhniki global'noy optimizatsii, *Informatsionnye tekhnologii*, 2018, No. 6, pp. 370-386.
18. *Norenkov I.P., Uvarov M.Yu.* Podderzhka prinyatiya resheniy na osnove patternov proektirovaniya [Decision support based on design patterns], *Nauka i obrazovanie: nauchnoe izdanie* [Science and Education: Scientific publication]. Moscow: Izd-vo MGTU im. N.E. Baumana, 2011, No. 9, pp. 25-32.
19. *Bozhko A.N.* Metody strukturnogo analiza slozhnykh izdeliy v integrirovannykh CFD/CAM-sistemakh [Structural analysis methods for complex products in integrated CAD], *Informatsionnye tekhnologii* [CAM-systems. Information Technologies]. Moscow: Izd-vo MGTU im. N.E. Baumana, 2018, No. 8, pp. 499-506.
20. *Stolyarov A.I., Donetskaya Yu.V., Gatchin Yu.A.* Osobennosti SAPR pri proektirovanii kompleksa [Features of CAD in the design of the complex], *Vestnik Irkutskogo gosudarstvennogo tekhnicheskogo universiteta* [Bulletin of the Irkutsk State Technical University], 2018, No. 7 (138), pp. 88-95.

Статью рекомендовал к опубликованию д.т.н., профессор А.Г. Коробейников.

Лебедев Борис Константинович – Южный федеральный университет; e-mail: lebedev.b.k@gmail.com; 347928, г. Таганрог, пер. Некрасовский, 44; тел.: 89282897933; кафедра систем автоматизированного проектирования; профессор.

Лебедев Олег Борисович – e-mail: lebedev.ob@mail.ru; тел.: 89085135512; кафедра систем автоматизированного проектирования; доцент.

Лебедева Екатерина Олеговна – e-mail: lbedevakate@mail.ru; тел.: 89289591426; кафедра систем автоматизированного проектирования; аспирант.

Lebedev Boris Konstantinovich – Southern Federal University; e-mail: lebedev.b.k@gmail.com; 44, Nekrasovsky, Taganrog, 347928, Russia; phone: +79282897933; the department of computer aided design; professor.

Lebedev Oleg Borisovich – e-mail: lebedev.ob@mail.ru; phone: +79085135512; the department of computer aided design; associate professor.

Lebedeva Ekaterina Olegovna – e-mail: lbedevakate@mail.ru; phone: +79289591426; the department of computer aided design; graduate student.

УДК 004.021 DOI 10.18522/2311-3103-2020-2-157-169

А.А. Могилев, В.М. Курейчик

МОДИФИЦИРОВАННЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ПЛАНИРОВАНИЯ ПРОЕКТОВ, РЕАЛИЗОВАННЫЙ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ*

Предложена структура модифицированного генетического алгоритма для решения задачи построения расписания проекта с учетом ограниченности ресурсов, реализованного с использованием облачных вычислений, проведен вычислительный эксперимент, в ходе которого было произведено сравнение результатов работы предложенного алгоритма с лучшими из известных, на данный момент, результатами. Исходя из результатов эксперимента был сделан вывод, о том, что предложенный алгоритм может быть использован для планирования работ реальных проектов, так как с его помощью возможно составлять расписания для проектов с количеством работ n = 90 за приемлемый промежуток времени. При планировании проектов с количеством работ n = 30, n = 60, n = 90, 120 время выполнения предложенного алгоритма было меньше, чем время выполнения стандартного генетического алгоритма в 2.8, в 4, в 5.5 и 6.8 раз соответственно. В связи с тем, что задача построения расписания проекта с учетом ограниченности ресурсов является NP-трудной, проблема создания новых и модификации существующих методов её решения по-прежнему остается актуальной. Для планирования проектов с большим количеством работ целесообразно использовать облачные вычисления, так как планирование таких проектов может потребовать много времени и вычислительных ресурсов. Использование облачных вычислений позволит сократить время выполнения генетического алгоритма за счет предоставления поставщиком облачного сервиса больших вычислительных ресурсов. В связи с этим, предложенный в данной работе алгоритм отличается от уже имеющихся использованием облачных вычислений для распределения нагрузки между рабочими станциями, на которых одновременно выполняется данный алгоритм. Применение в генетическом алгоритме модифицированных операторов, а также использование облачной инфраструктуры как услуги для реализации генетического алгоритма при решении задачи планирования проектов определяет научную новизну исследования.

Теория расписаний; задача построения расписания для проекта с учетом ограничений на ресурсы; генетический алгоритм.

* Работа выполнена при финансировании гранта РФФИ №18-07-00050.