УДК 681.3.016 DOI: 10.31114/2078-7707-2020-4-58-65

Популяционный алгоритм планирования СБИС методом кристаллизации россыпи альтернатив

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

Южный федеральный университет, г. Таганрог, lebedev.ob@mail.ru

***Аннотация* — Разработана архитектура поискового алгоритма планирования СБИС на основе метода кристаллизации россыпи альтернатив, опирающаяся на структуру популяционного алгоритма, оперирующего с множеством решений, реализующая эволюционную стратегию случайного направленного поиска решения. Алгоритм, связанный с эволюционной памятью, стремится к запоминанию и многократному использованию способов достижения лучших результатов. Рассмотрены ключевые моменты анализа альтернатив в процессе эволюционной коллективной адаптации, названной по аналогии с процессами вычленения объектов (формирования кристаллов) кристаллизацией. Временная сложность алгоритма, полученная экспериментальным путем, лежит в пределах O(n2)-O(n3).**

***Ключевые слова* — оптимизация, роевой интеллект, адаптивное поведение, метод кристаллизации россыпи альтернатив, СБИС, планирование.**

I. Введение

Планирование – это ранняя фаза конструкторского проектирования и заключается в размещении на поле кристалла непересекающихся модулей, имеющих чаще всего прямоугольную форму [1]. Важным шагом при решении задачи планирования является выбор способа отображения геометрических отношений между модулями [2]. В работе используется гильотинный подход к представлению и поиску решения (плана). Гильотинная структура может быть получена путем рекурсивного деления прямоугольника на две части горизонтальными и/или вертикальными разрезами. Гильотинное представление имеет ряд преимуществ, такие как небольшие затраты на кодирование и меньшее пространство поиска, приводящие к более быстрому построению плана [3,4]. Задача планирования относится к классу *NP*. Существуют различные подходы к решению проблемы планирования [5-7]. Анализ существующих подходов к решению поставленной задачи показал, что удачными являются подходы, основанные на методах эволюционного моделирования [8]. Тем не менее, в последнее время для решения различных «сложных» задач, к которым относятся и задачи планирования всё чаще используются способы, основанные на применении биоинспирированных моделей [9-11]. В работе используется новая метаэвристика, имеющая тенденцию к использованию альтернатив из наилучших найденных решений. В процессе эволюционной коллективной адаптации методами дискриминантного анализа формируются оценки приспособленности, используемые при формировании решений. Совокупность данных об альтернативах и их оценках составляет россыпь альтернатив. Дискриминантный анализ альтернатив в процессе эволюционной коллективной адаптации назван по аналогии с процессами формирования кристаллов - кристаллизацией. Отсюда название метода оптимизации − метод кристаллизации россыпи альтернатив (КРА), (Crystallization of alternatives field (CAF)) [12-13].

II. Постановка задачи планирования

План для множества модулей M представляет собой прямоугольник, разрезанный вертикальными и горизонтальными линиями на множество блоков r_i, в каждый из которых помещается соответственно модуль $_{mi}$. Каждый блок r_i, предназначенная для размещения модуля m_i, имеет размеры x_i и y_i. Размеры блока должны соответствовать ограничениям (1):

$$h_i \le y_i,\ w_i \le x_i,\ S_i \le x_i \cdot y_i\ , \qquad (1)$$

где h_i и w_i – размеры модуля, S_i – площадь модуля.

Цель оптимизации – минимизация общей площади плана U, при соблюдении ограничений (1).

Будем считать, что связи между модулями m_i и m_j связывают центры соответствующих областей r_i и r_j [1,3]. Обозначим через d_{ij} длину связей между m_i и m_j а через c_{ij} – стоимость связей. Тогда критерий оптимизации при планировании имеет вид:

$$F = \sum_{i=1}^{n} x_i \cdot y_i \ + \lambda \sum_{i,j=1}^{n} c_{ij} \cdot d_{ij} \qquad (2)$$

Задать план R это: задать структуру дерева разрезов, т.е. последовательность бинарных разрезов z_j; для внутренних вершин дерева, соответствующих разрезам, указать для каждого разреза z_j тип H-горизонтальный или V-вертикальный; пометить листья дерева номерами блоков модулей; для модулей с фиксированными размерами указать их ориентацию [3,4]. На рис. 1 представлено дерево разрезов D. Геометрическое представление плана формируется путем последовательной бинарной свертки блоков по дереву разрезов, начиная от листьев дерева по направлению к корневой вершине [1]. На каждом шаге свертки в результате слияния блоков

r_i и r_j формируется блок r_k, производится определение его размеров и новых размеров для r_i и r_j [2]. Пусть $R=\{r_i|i=1,2,...,n_R\}$ множество элементов, соответствующих листьям дерева (блокам), рассматриваемых в качестве операндов, а множество $Z=\{H,V\}$ – соответствует типам разрезов, рассматриваемых в качестве операторов [1]. Представим польскую запись в виде шаблона – вектора P_i=<o o $\underline{1}$ o $\underline{2}$ o $\underline{3}$ o $\underline{4}$ ... o $\underline{m}$> с пронумерованными позициями между знаками o для разрезов от 1 до m. Знаки типа (o) соответствуют листьям дерева (блокам), а знаки типа (•) соответствуют внутренним вершинам (разрезам) дерева. Знаки (o) и (•) в составе вектора P_k идентифицированы и упорядочены. В постфиксной записи представления дерева разрезов слева от знака • число знаков o всегда больше числа знаков •. Пусть число знаков (o), соответствующих блокам – $n_R=6$, число знаков (•), соответствующих операторам разреза, – $n_Z=5$. Возможная польское выражение имеет вид P_k=<o o $\underline{\bullet}$ o o $\underline{\bullet\bullet}$ o $\underline{\bullet}$ o $\underline{\bullet}$>.

Процесс восстановления дерева по польскому выражению достаточно прост [2]. Последовательно слева направо просматривается польское выражение, и отыскиваются элементы H или V, соответствующие разрезам. Каждый разрез объединяет два ближайших, образованных на предыдущих шагах, подграфа, расположенных в записи слева от букв H или V. Польское выражение для дерева, представленного на рис. 1 имеет вид:

$$P_i=<r_1\ r_2\ V\ r_3\ V\ r_4\ r_5 V\ H\ r_6\ r_7\ V\ r_8 H\ r_9\ r_{10}\ r_{11}\ H\ V\ H\ V>$$

Отметим основные свойства польского выражения для представления дерева разрезов, выполнение которых необходимо, чтобы записи соответствовало дерево разрезов. Для того, чтобы польское выражение являлось представлением бинарного дерева разрезов необходимо выполнение следующих условий:

1. Польское выражение представляет упорядоченную последовательность символов.
2. В состав выражения входят по одному разу все элементы множества $R=\{r_i|i=1,2,...,n_R\}$, соответствующих блокам, и n_Z элементов z_j, соответствующих разрезам ***H*** или ***V***.
3. Для дерева разрезов всегда выполняется равенство $n_R=n_Z+1$. Это соотношение строго выполняется в польском выражении.
4. В польском выражении слева от любого элемента число элементов, соответствующих блокам, больше числа элементов, соответствующих разрезам, минимум, на единицу.

Если польское выражение P_k удовлетворяет вышеперечисленным условиям 1-4, то оно легитимно, и является символьным представлением решения задачи планирования. Различные решения получаются путём комбинирования взаимным расположением элементов упорядоченного списка, удовлетворяющим условиям (1-4) и конкретной идентификацией вершин дерева разрезов. В работе пространство решений представляется множеством легитимных выражений P. Поиск решения сводится к поиску легитимного выражения $P_k \in P$ с оптимальным значением показателя качества. Показатель качества рассчитывается после выполнения свертки блоков по дереву разрезов.

III. Синтез плана методом кристаллизации россыпи альтернатив

Разработка алгоритма на основе метаэвристики КРА предполагает выполнение следующих этапов [12, 13]. На этапе анализа задачи выявляется:

Совокупность объектов, составляющих решение;
Набор признаков и отношений между объектами;
Значения отношений, признаков;
Особенности и свойства решений.

На основе результатов анализа каждое решение задачи представляется множеством агентов $E=\{e_\alpha|\alpha=1,2,...,n_e\}$, где n_e – число агентов. Каждый агент e_α может находиться в одном из альтернативных состояний множества $S_\alpha=\{s_{\alpha i}|i=1,2,...,n_{\alpha i}\}$, где $n_{\alpha i}$ – число состояний агента e_α. Решение P_k определяется совокупностью альтернативных состояний множества агентов E. Обозначим как s^k_α альтернативное состояние агента e_α в решении P_k. Тогда $SR_k=\{s^k_\alpha|\alpha=1,2,...,n_e\}$ – множество состояний всех агентов в решении P_k. Для организации коллективной эволюционной памяти (КЭП), хранящей информацию об альтернативах агентов в данном решении и об оценке этого решения, разрабатывается структура базы данных, называемая россыпью альтернатив (РА).

КЭП алгоритма построения плана представляет собой набор статистических показателей, отражающих для каждой реализованной альтернативы число θ ее вхождений в лучшие решения на предыдущих итерациях алгоритма и число δ показывающее полезность реализованной альтернативы при построении решений на предыдущих итерациях алгоритма. В процессе поиска в КЭП множество оценок решений трансформируется в интегральные оценки альтернатив. На каждом шаге поиска производится генерация новых решений и пересчет интегральных оценок. Для этих целей на основе эвристик разработан конструктивный алгоритма синтеза решения, в соответствии с данными КЭП. При этом происходит рост оценок лучших альтернатив и снижение оценок худших альтернатив.

Поиск решения сводится к поиску такого легитимного выражения P_i, которое оптимизирует показатель качества (1-4). В постфиксной записи представления дерева разрезов слева от знака • число знаков **o** всегда больше числа знаков •. Пронумеруем позиции между знаками **o**:

o o $\underline{1}$ **o** $\underline{2}$ **o** $\underline{3}$ **o** $\underline{4}$ … $\underline{n_x\text{-}2}$ **o** $\underline{n_x\text{-}1}$.

Максимально возможное число знаков • в позиции, равно номеру позиции. Если польское выражение P_i удовлетворяет вышеперечисленным условиям 1-4, то оно обладает следующими свойствами:

- В упорядоченной последовательности элементов польского выражения первые два элемента соответствуют блокам, (условие 4);
- Последний элемент упорядоченной последовательности элементов польского выражения соответствует разрезу ***H*** или ***V***, (условие 4);
- Максимальное количество элементов, соответствующим разрезам, в позиции i шаблона, равно i, (условие 4).

Пусть имеется некоторая польская запись P_k, в виде упорядоченной последовательности элементов, которой соответствует дерево разрезов D_k на рис 1.

$$P_k = <\underset{1}{r_4}\ \underset{2}{r_3}\ \underset{3}{r_2}\ \underset{4}{H}\ \underset{5}{V}\ \underset{6}{r_1}\ \underset{7}{r_5}\ \underset{8}{r_6}\ \underset{9}{r_7}\ \underset{10}{H}\ \underset{11}{V}\ \underset{12}{H}\ \underset{13}{H}>, \quad (3)$$

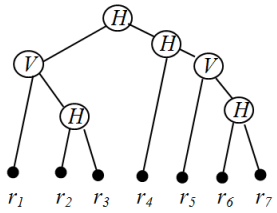

Рис. 1. Дерево разрезов D_k, для польской записи P_k

Каждый агент e_α имеет индивидуальный идентификатор I_α. Состояние агента определяется значением порядкового номера идентификатора I_α агента в польской записи. Таким образом, множество $S_\alpha=\{s_{\alpha i}|i=1,2,\ldots,n_{\alpha i}\}$ альтернативных состояний агента e_α, это множество возможных порядковых номеров i идентификатора агента e_α в польской записи. Под россыпью альтернатив (РА) решения в работе называется структура данных, несущая информацию об альтернативах агентов в данном решении и о значениях их полезностей, используемых в качестве коллективной эволюционной памяти (КЭП). Каждое решение, описываемое выражением P_k, представляется в виде матрицы $R_k=||r_{kji}||_{m\times n}$, называемой россыпью альтернатив (РА), представленной на рис. 2. Матрица R_k составлена для польского выражения (3), которому соответствует дерево разрезов на рис.1.

В работе рассматривается такая постановка задачи, когда несколько агентов могут иметь одну и ту же метку, но каждый агент имеет только одну метку. Номер столбца – j матрицы R_k соответствует идентификатору (имени, значению, номеру) агента. Размерность столбца (вектора $R_{kj}=\{r_{kij}|i=1,2,\ldots,n\}$) определяется числом n агентов (число элементов в польской записи). В каждом столбце (векторе R_{kj}) несколько элементов r_{kij}, могут иметь значение, отличное от нуля. Остальные элементы вектора R_{kj} имеют нулевые значения. Номер строки – i матрицы R_k соответствует порядковому номеру агента, помеченного j-м идентификатором, в упорядоченной последовательности польской записи. Размерность строки (вектора $R_{ki}=\{r_{kij}|j=1,2,\ldots,m\}$) определяется числом m возможных имен агентов e_α (числом идентификаторов агентов). В каждой строке (векторе R_{ki}) только один элемент r_{kij} имеет значение, отличное от нуля, так как каждый агент имеет одно имя.

$R_k =$

	БЛОКИ							РАЗРЕЗЫ	
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	H	V
№	1	2	3	4	5	6	7	8	9
1.				δ_k					
2.			δ_k						
3.		δ_k							
4.								δ_k	
5.									δ_k
6.	δ_k								
7.					δ_k				
8.						δ_k			
9.							δ_k		
10.								δ_k	
11.									δ_k
12.								δ_k	
13.								δ_k	

Рис. 2. Россыпь альтернатив R_k

В матрице на рис. 2 тринадцать агентов используют 9 идентификаторов, 7 агентов, соответствующих блокам, используют 7 типов идентификаторов - (r_1-r_7), 6 агентов, соответствующих разрезам, используют 2 типа идентификаторов (H, V) – из них 4 агента используют идентификатор H, 2 агента используют идентификатор V. Каждому агенту e_α в матрице R_k соответствует своя ячейка r_{kij}, где i – порядковый номер агента e_α в польском выражении P_k, а j – номер столбца матрицы R_k соответствующий идентификатору агента e_α. Этот идентификатор, является значением элемента p_i в польском выражении P_k. Значением элемента $r_{ij}\in R_k$, соответствующего агенту, помеченному j-м идентификатором, и расположенному в i-ой позиции упорядоченной последовательности P_k, является оценка полезности $\delta_k=f(\xi_k)$ решения, построенного в соответствии с польской записью P_k, где ξ_k – оценка плана. Каждому решению P_k соответствует своя матрица R_k.

Множество индивидуальных россыпей альтернатив $R=\{R_k|k=1,2,\ldots,n_k\}$ формируется на базе сгенерированного множества решений, представленных в виде польских выражений, $P=\{P_k|k=1,2,\ldots,n_k\}$. Платформой для организации популяционной процедуры поиска решений является интегральная россыпь альтернатив (ИРА) – R^*, которая формируется путем объединения всех россыпей альтернатив: $R^*=\sum_k R_k$.

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

Фактически r^*_{ij} является суммарным значением полезностей решений, в которых агентом e_j была реализована альтернатива s_{ij}. ИРА используется в качестве коллективной эволюционной памяти и служит базой для формирования новых решений. Поисковый алгоритм на основе метода КРА опирается на структуру популяционного алгоритма, оперирующего с множеством решений, реализующего эволюционную стратегию случайного направленного поиска решения. Алгоритм связан с эволюционной памятью, стремится к запоминанию и многократному использованию способов достижения лучших результатов. Коллектив не имеет централизованного управления, и его особенностями являются наличие непрямого обмена

информацией. Непрямой обмен – стигмержи (stigmergy), представляет собой разнесённое во времени взаимодействие, при котором один агент изменяет некоторые области КЭП, а другие агенты позже используют уже измененную информацию в этих областях. В процессе поиска формируется интегральная россыпь альтернатив (ИРА) – матрица R^*. Процесс поиска решений итерационный. Каждая итерация l включает три этапа.

На первом этапе каждой итерации конструктивным алгоритмом формируется n_k решений P_k. Каждое решение P_k представляется в виде польского выражения, формируемого путем последовательного построения упорядоченной последовательности. Элементами польского выражения являются идентификаторы блоков (типа r_i) и операторов разреза (типа *H, V*). Для повышения эффективности вычислительного процесса формирование каждого решения P_k выполняется на базе *дублера* интегральной россыпи альтернатив, который в начале поиска решения эквивалентен оригиналу, но в процессе поиска подвергается значительным изменениям и по окончании поиска решения P_k стирается из памяти. Поиск решения выполняется множеством агентов E, посредством вероятностного выбора каждым агентом e_α порядкового номера в последовательности P_k. Решение P_k трансформируется в дерево разрезов D_k. Далее осуществляется свертка дерева разрезов D_k. По результатам свертки рассчитываются размеры блоков, оценка ξ_k решения P_k и оценка полезности δ_k множества альтернатив, реализованных агентами в решении P_k. Формируется россыпь альтернатив R_k решения P_k.

На втором этапе каждая оценка полезности δ_k множества альтернатив $SR_k=\{s^k_i|i=1,2,\ldots,n_e\}$ каждого решения P_k, сформированного на первом этапе, прибавляется к интегральным оценкам полезности того же множеств альтернатив в *оригинальной* интегральной россыпи альтернатив R^*. В работе используется циклический метод формирования решений. В этом случае наращивание оценок интегральной полезности δ_{ij}^* в оригинальной интегральной россыпи альтернатив R^* выполняется после полного формирования множества решений P на итерации l. $R^*(l)= R^*(l-1) + \sum_k R_k(l)$. На третьем этапе осуществляется снижение оценок полезности δ_{ij}^* *оригинальной* интегральной россыпи альтернатив R^* на величину μ.

Представим популяционный алгоритм планирования СБИС на основе метода КРА.

1. В соответствии с исходными данными задается множество модулей $M=\{m_i|i=1,2,\ldots,n_R\}$. Для каждого модуля задаются параметры – h_i, w_i, S_i.Определяется число блоков n_R и число операторов разреза $n_Z=n_R-1$. Число идентификаторов блоков – n_R. Число идентификаторов разрезов – 2. Число агентов $n_A=(n_R+n_Z)$, Число идентификаторов всех агентов $n_I=(n_R+2)$.

2. Формируется структура индивидуальной и интегральной матриц россыпи альтернатив R_k и $R^*(0)$. Каждой ячейке r^*_{ij} присваивается начальное значение δ_0.

3. Задаются значения параметров: число итераций N_l. число решений на одной итерации N_k.

4. $l=1$.

5. $k=1$.

6. Формирование конструктивным алгоритмом решения (польской записи) $P_k(l)$ на базе *дублера интегральной россыпи альтернатив:* $R^*(l-1)$ $R^*(l-1)$.

7. Трансформация польской записи $P_k(l)$ в дерево разрезов $D_k(l)$.

8. Свертка дерева разрезов $D_k(l)$.

9. Рассчитываются размеры блоков, оценка $\xi_k(l)$ решения $P_k(l)$ и оценка полезности $\delta_k(l)$ множества альтернатив $SR_k(l)$, реализованных агентами в решении $P_k(l)$.

10. Формирование индивидуальной россыпи альтернатив – матрицы $R_k(l)$.

11. Если ($k<N_k$), то $k=k+1$ и переход к пункту 6, иначе переход к пункту 12.

12. Фиксация лучшего решения.

13. Наращивание оценок интегральной полезности δ_{ij}^* в оригинальной интегральной россыпи альтернатив R^*. $R^*(l)= R^*(l-1)+\sum_k R_k(l)$.

14. Снижение значений оценок интегральной полезности δ_{ij}^* в оригинальной интегральной россыпи альтернатив $R^*(l)$: $\delta_{ij}^*=\delta_{ij}^*-\mu$;

15. Если ($l<N_l$), то $l=l+1$ и переход к пункту 5, иначе переход к пункту 16;

16. Конец работы алгоритма.

Конструктивный алгоритм синтеза польского выражения для представления плана СБИС основе метода КРА формулируется следующим образом. Введем обозначения:

t – номер позиции в польском выражении P_k;
t^* – последняя заполненная позиция в формируемом польском выражении;
n_{R0} – число блоков;
n_{Z0} – число разрезов;
$n_{I0}=(n_{R0}+2)$ – общее число идентификаторов;
$n_R(t)$ – число элементов типа блок в позициях польского выражения с1 по *(t-1)*;
$n_Z(t)$ – число элементов типа разрез в позициях польского выражения с1 по *(t-1);*

i_R, i_Z – вспомогательный элемент для индикации типа агента (блок или разрез; если i_R=1, то агент блок; если i_Z=1, то агент разрез).

1.Формируется дублер интегральной россыпи альтернатив – матрица *R*(l-1).* Далее все действия выполняются на базе дублера матрицы *R*(l-1), подвергающегося изменениям в процессе работы алгоритма.* Всем элементам формируемого польского выражения P_k присваивается нулевое значение. Всем элементам $r_{ij} \in R_k$ присваивается нулевое значение.

2.$t=1$. $i_R=0$, $i_Z=0$

3.$n_R(t)=0$ и $n_Z(t)=0$;

4.Если $n_R(t)=n_{R0}$ (т.е. все идентификаторы блоков размещены в польской записи), то переход к пункту 11, иначе переход пункту 5.

5.Если $(n_R(t)-n_Z(t))\geq 2$, то в соответствии с условием (4) в позиции t допускается размещение элемента типа блок или типа разрез, и переход к пункту 6; иначе переход к пункту 8.

6.В строке $R^*_t(l-1)=\{r^*_{tj}(l-1) \mid j=1,2,\ldots, n_{l0}\}$ *дублера матрицы* $R^*(l-1)$ с вероятностью $\pi=r^*_{tj}(l-1)/\sum R^*_t(l-1)$ выбирается j_m-я ячейка $r^*_{tj\ m}(l-1)$, где $\sum R^*_t(l-1)$ – сумма значений всех ячеек строки $R^*_t(l-1)$ с индексом j от 1 до n_{l0}.

7.Если $j_m \leq n_{R0}$, то j_m-я ячейка $r^*_{tj\ m}(l-1)$ соответствует блоку и i_R=1, иначе j_m соответствует разрезу и i_Z=1. Переход к пункту 10.

8.Если $(n_R(t)-n_Z(t))<2$, то в соответствии с условием (4) в позиции t может быть размещен *только* элемент типа блок.

9.В строке $R^*_t(l-1)=\{r^*_{tj}(l-1) | j=1,2,\ldots, n_{R0}\}$ *дублера матрицы* $R^*(l-1)$ с вероятностью $\pi=r^*_{tj}(l-1)/\sum R^*_t(l-1)$ выбирается j_m-я ячейка $r^*_{tj\ m}(l-1)$,где $\sum R^*_t(l-1)$ – сумма значений всех ячеек строки $R^*_t(l-1)$ с индексом j от1 до n_{R0}. i_R=1.

10.Идентификатор, соответствующий индексу j_m ячейки $r^*_{tjm}(l-1)$ в матрице $R^*(l-1)$, заносится в t-ю позицию польской записи. В t-й строке и j_m–ом столбце матрицы дублере $R^*(l-1)$ значения всех ячеек r^*_{tj} обнуляются. Переход к пункту12.

11.Если $n_Z(t)<n_{Z0}$ (т.е. не все идентификаторы разрезов размещены в польской записи), то переход к пункту 12, иначе переход к пункту 13.

12.Из двух последних ячеек t-*й* строки $R^*_t(l-1)$ с вероятностью $\pi=r^*_{tj}(l-1)/\sum R^*_t(l-1)$ выбирается j_m-я ячейка $r^*_{tj\ m}(l-1)$,где $\sum R^*_t(l-1)$ – сумма значений всех ячеек строки $R^*_t(l-1)$ с индексом j от $(n_{l0}-1$ до $n_{l0})$. Идентификатор разреза – $(H$ или $V)$, соответствующий индексу j_m ячейки $r^*_{tj}(l-1)$ в матрице $R^*(l-1)$, заносится в t-ю позицию польской записи. i_Z=1.

13.Если $t<(n_{R0}+n_{Z0})$, то переход к 14, иначе переход к 16.

14.Если i_R=1, то $n_R(t)=n_R(t)+1$, иначе, если i_Z=1, то $n_Z(t)=n_Z(t)+1$.

15.i_R=0, i_Z=0, t= t+1 и переход к 4.

16.Конец работы алгоритма.

IV. Экспериментальные исследования

На основе разработанного алгоритма разбиения создана программа КРА**.** Для проведения экспериментов программ**ы планирования** была использована процедура синтеза контрольных примеров с известным оптимумом $F_{опт}$ по аналогии с известным методом AFEKO – Floorplanning Examples with Known Optimal area [14-16]. Исследованию подвергались примеры, содержащие до 1000 модулей. Оценкой качества служит ***«степень качества»*** – величина $F_{опт}/F$, где F – оценка полученного решения. На основе обработки экспериментальных исследований была построена средняя зависимость степени качества от размера популяции (рис. 3) и от числа итераций (рис. 4). В результате экспериментов установлено, что при объеме популяции M=100 алгоритм сходится в среднем на 130 итерации. При этом у 75% примеров полученное решение было оптимальным, у 10% примеров решения были на 5% хуже, а у 15% примеров решения были хуже не более, чем на 2%.

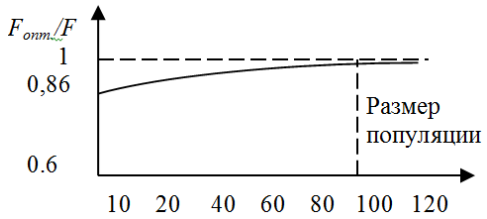

Рис. 3 Зависимость качества решений алгоритма от размера популяции

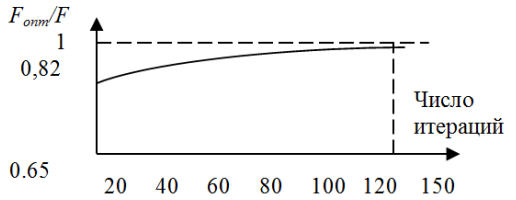

Рис. 4. Зависимость качества решений алгоритма от числа итераций

Для сравнения качества решений задачи планирования кристалла СБИС, применялись стандартные тесты (бенчмарки) для оценки разработанных алгоритмов [17-20]. Эксперименты проводились на MCNC (Microelectronics Center of North Carolina) образцах плат. Были использованы пять типов корпусов, в которых количество элементов варьировалось от 9 до 49. Самый большой корпус – для платы ami49 на 49 элементов, с 42 интерфейсами ввода-вывода, 408 связями и 931 ножкой. Для сравнения были выбраны современные планировщики. Представленный алгоритм находит решения, по площади превосходящие результаты существующих алгоритмов (табл. 1). Сравнения результатов

произведено по показателю “мертвая зона”. Показатель “мертвая зона” определяется как отношение свободной от модулей площади плана к общей площади плана в процентах. В колонках таблицы 1 приведены результаты планировщиков: АД – адаптивный [5]; КЛА – кластерный – [17]; ГЕН – генетический [18]; ГБР – гибридный [19]; МА – меметический [7]; АСО – муравьиный [20]. Отметим, что результаты у сравниваемых зарубежных алгоритмов получены на базе платформы Ultra1-Spark, а эксперименты разработанных алгоритмов проводились на ЭВМ типа IBM PC с процессором Pentium. В колонке [КРА] приведены результаты, полученные алгоритмом кристаллизации россыпи альтернатив. Следует отметить, что экспериментальная временная сложность алгоритма на одной итерации при фиксированных значениях управляющих параметров составляет $O(nlgn)$, а временная сложность существующих алгоритмов [17-20] – $O(n^2)$, где n – число модулей.

Алгоритм на основе парадигмы КРА находит решения, превосходящие результаты существующих алгоритмов по площади, кроме того, они оптимизирует длину соединений.

Таблица 1

Сравнение результатов планировщиков

Тест	Мертвая зона (%)						
	АД	КЛА	ГЕН	ГБР	МА	АСО	КРА
apte	2,11	2,34	2.3	2.2	2.4	2,01	2.00
xerox	3,37	3,67	3.6	3.7	3.36	3,15	3.1
hp	3,39	3,86	3.9	3.3	3.35	3,22	3.12
ami33	3,54	4,38	4.4	4.8	4.1	3,5	3.37
ami49	4,07	4,84	4.8	5.0	4.4	4,01	3.9

Предлагаемый алгоритм планирования позволяет лучше упаковывать модули по сравнению с планировщиками аналогичного уровня (качество упаковки улучшилось в среднем на 3–6%). При больших размерностях временные показатели разработанного алгоритма превосходят показатели сравниваемых алгоритмов при лучших значениях целевой функции.

V. Заключение

Сущность рассматриваемой в работе комбинаторной задачи заключается в том, что решение представляет комбинацию уникальных компонент, каждый из которых выбирается, как правило, из конечного набора конкурирующих между собой вариантов. Целью является поиск оптимальной комбинации вариантов компонент, которые обеспечивают наилучшее решение задачи. Большинство известных алгоритмов используют традиционные итерационные улучшающие структуры, основаны на слепом случайном поиске. Основным недостатком, присущим этому подходу, является вхождение алгоритмов в локальный оптимум, часто далекий от – глобального. Для этого нужны “осмысленные” изменения, направленные в сторону глобального оптимума. Такие свойства как раз присущи адаптивным поисковым процедурам на основе самообучения и самоорганизации. Разработана новая метаэвристика интеллектуальной оптимизации, учитывающая тенденцию к использованию альтернатив из наилучших найденных решений, базирующаяся на моделировании коллективного интеллекта. Предлагаются новые технологии, принципы и механизмы решения задачи планирования, использующие математические методы, в которых заложены принципы природных механизмов принятия решений. Для компактного представления решения задачи планирования используется модифицированная польская запись. Сформулированы необходимые условия, при выполнении которых построенная польская запись представляется в виде легитимного выражения, являющемся решением задачи планирования.

Это позволило создать пространство решений, в рамках которого организован поисковый процесс, базирующийся на моделировании адаптивного поведения популяции агентов. По сравнению с существующими алгоритмами достигнуто улучшение результатов. Временная сложность алгоритма, полученная экспериментальным путем, лежит в пределах $O(n^2)$-$O(n^3)$.

Поддержка

Работа выполнена при финансовой поддержке гранта РФФИ № 20-07-00260 а.

Литература

[1] Курейчик В.М., Лебедев Б.К., Лебедев В.Б. Планирование сверхбольших интегральных схем на основе интеграции моделей адаптивного поиска // Известия РАН. Теория и Системы Управления, №1, 2013. С. 84–101.

[2] Лебедев Б.К., Лебедев О.Б., Лебедев В.Б. Методы, модели и алгоритмы размещения. Ростов-на-Дону: Изд-во ЮФУ, 2015. 150 с.

[3] Young F.Y. et al. Slicing floorplans with boundary constraints // IEEE Transactions on Computer-Aided Design, 1999. pp. 1385–1389.

[4] Lin C.-T. et al. GPE: A New Representation for VLSI Floorplan Problem // IEEE International Conference on Computer Design (ICCD'02), 2002. pp. 42–44.

[5] Qi L. Simulated annealing based thermal-aware floorplanning // International Conference on Electronics, Communications and Control, 2011. pp. 463–466.

[6] Pritha B., Megha S. Floorplanning for Partially Reconfigurable FPGAs // IEEE Trans. Comput. Aided Des. Integr. Circuits Sys., v.30, №1, 2011. pp. 8-17.

[7] Potti S., Pothiraj S. GPGPU Implementation of Parallel Memetic Algorithm for VLSI Floorplanning Problem // Trends in Computer Science, Engineering and Information Technology, Communications in Computer and Information Science, V.204, 2011. pp. 432–441.

[8] Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой:

учебное пособие. М: Издательство МГТУ им. Н.Э. Баумана, 2014. 448 с.
[9] Лебедев Б.К., Лебедев О.Б. Биоинспирированные методы планирования кристалла СБИС // VI Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем». Сборник трудов. М.: ИППМ РАН, 2012. С.171-176.
[10] Лебедев О.Б. Модели адаптивного поведения муравьиной колонии в задачах проектирования. – Таганрог: Изд-во ЮФУ, 2013. 199 с.
[11] Лебедев Б.К., Лебедев В.Б. Планирование на основе роевого интеллекта и генетической эволюции // Известия Южного федерального университета. Технические науки. № 4, 2009. С. 25–33.
[12] Лебедев Б.К., Лебедев В.Б. Оптимизация методом кристаллизации россыпи альтернатив // Известия ЮФУ. Изд-во ТТИ ЮФУ, №7, 2013. С. 11-17.
[13] Лебедев Б.К., Лебедев В.Б. Метод кристаллизации россыпи альтернатив // Сборник научных трудов VII Международной научно-практической конференции “Интегрированные модели и мягкие вычисления в искусственном интеллекте”. М.: Изд-во Физматлит, 2013. С. 903-912.
[14] Cong J., Romesis M., Xie M. UCLA Optimality Study Project. http://cadlab.cs.ucla.edu/~pubbench. 2004.
[15] MCNC Electronic and Information Technologies (Online). Available: www.mcnc.org.
[16] hMetis http://www-users.cs.umn.edu/karypis/metis/ hmet300. HB Floorplan Benchmarks [Online]. Available: http://cadlab.cs.ucla.edu/ cpmo/HBsuite.html.
[17] Ma Q., Young E.F. Multivoltage Floorplan Design // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, №4. 2010. pp. 607–617.
[18] Ерошенко И.Н. Разработка генетического алгоритма кластерного планирования СБИС // Известия Южного федерального университета. Технические науки, №7. 2010. С. 54-60.
[19] Kureichik V.M., Lebedev B.K., Lebedev O.B. Hybrid evolutionary algorithm of planning VLSI // Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference GECCO-2010. Assoc. Comput. Mach., Spec. Interest Group Genet., Evol. Comput. (ACM SIGEVO). Portland, OR, 2010. pp. 821-822.
[20] Лебедев О.Б. Планирование СБИС на основе метода муравьиной колонии // Известия Южного федерального университета. Технические науки. №7, 2010. С.67-73.

Population VLSI Planning Algorithm by the Method of Crystallization of alternatives field

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

Southern Federal University, Taganrog Southern Federal University, lebedev.ob@mail.ru

Abstract — **An architecture has been developed for a search population algorithm for VLSI planning based on a method of crystallization of alternatives field, based on the structure of a population algorithm that operates with a variety of solutions and implements an evolutionary strategy of random directed search for a solution. The algorithm associated with evolutionary memory seeks to memorize and reuse ways to achieve better results.**

The team does not have centralized management, and its features are the presence of indirect exchange of information. Indirect exchange is a time-spaced interaction in which one agent changes some areas of co-evolutionary memory, while other agents later use information that has already been changed in these areas. The totality of data on alternatives and their assessments is a scattering of alternatives. In the Crystallization of alternatives field (CAF) method, each solution is formed by a variety of agents. Each agent has many alternative conditions. Each agent may be in one of the alternative states. The decision is determined by a set of alternative states of many agents. The key points of the analysis of alternatives in the process of evolutionary collective adaptation, called by analogy with the processes of isolating objects (crystal formation) crystallization, are considered. An algorithm based on the crystallization of a placer of alternatives has been successfully applied to solve the VLSI planning problem. Such an approach is an effective way to search for rational solutions for optimization problems that can be interpreted as a scattering of alternatives.

In the search process, an integral placer of alternatives is formed - the matrix R^*. The decision-making process is iterative. Each iteration l includes three steps. At the first stage of each iteration, a constructive algorithm generates n_k solutions P_k,. Each solution P_k is represented in the form of a Polish expression formed by sequentially constructing an ordered sequence. Elements of the Polish expression are the identifiers of blocks (of type r_i) and section operators (of type H,V). The search for a solution is performed by the set of agents E, by means of the probabilistic choice by each agent e_α of the sequence number in the sequence P_k. The solution P_k is transformed into the section tree D_k. Next, convolution of the section tree D_k is carried out. According to the convolution results, the block sizes, the estimate ξ_k of the solution P_k, and the utility estimate δ_k of the set of alternatives implemented by the agents in the solution P_k are calculated. A scattering of alternatives P_k is formed. At the second stage, each utility estimate δ_k of the set of alternatives $SR_k=\{s^k_i|i=1,2,\ldots,n_e\}$ of each solution P_k formed at the first stage is added to the integral utility estimates of the same sets of alternatives in the original integral placer of alternatives. The paper uses the cyclic method of forming solutions. In this case, the build-up of estimates of the integral utility δ_{ij}^* in the original r^*_{ij} of the integral placer of alternatives R^* is performed after the complete formation of the set of solutions P at iteration l. At the third stage, the utility estimates δ_{ij}^* of the original integrated placer of alternatives R^* are reduced by μ.

The proposed planning algorithm allows for better packaging of modules compared to planners of a similar level (packaging quality improved by an average of 3-6%). The temporal complexity of the algorithm obtained experimentally lies in the range of $O(n^2)$-$O(n^3)$.

Keywords - optimization, swarm intelligence, adaptive behavior, method of crystallization of placer alternatives, VLSI, planning.

REFERENCES

[1] Kureichik V.M., Lebedev B.K., Lebedev V.B. Planirovaniye sverkhbol'shikh integral'nykh skhem na osnove integratsii modeley adaptivnogo poiska (Planning for super-large integrated circuits based on the integration of adaptive search models) // Izvestiya RAN. Teoriya i Sistemy Upravleniya, №1, 2013. pp. 84–101.

[2] Lebedev B.K., Lebedev O.B., Lebedev V.B. Metody, modeli i algoritmy razmeshcheniya (Methods, models and placement algorithms). Rostov-na-Donu: Izd-vo YUFU, 2015. 150 p.

[3] Young F.Y. et al. Slicing floorplans with boundary constraints // IEEE Transactions on Computer-Aided Design, 1999. pp. 1385–1389.

[4] Lin C.-T. et al. GPE: A New Representation for VLSI Floorplan Problem // IEEE International Conference on Computer Design (ICCD'02), 2002. pp. 42–44.

[5] Qi L. Simulated annealing based thermal-aware floorplanning // International Conference on Electronics, Communications and Control, 2011. pp. 463–466.

[6] Pritha B., Megha S. Floorplanning for Partially Reconfigurable FPGAs // IEEE Trans. Comput. Aided Des. Integr. Circuits Sys., v.30, №1, 2011. pp. 8-17.

[7] Potti S., Pothiraj S. GPGPU Implementation of Parallel Memetic Algorithm for VLSI Floorplanning Problem // Trends in Computer Science, Engineering and Information Technology, Communications in Computer and Information Science, V.204, 2011. pp. 432–441.

[8] Karpenko A.P. Sovremennyye algoritmy poiskovoy optimizatsii. Algoritmy, vdokhnovlennyye prirodoy: uchebnoye posobiye (Modern search engine optimization algorithms. Algorithms inspired by nature: a tutorial). M: Izdatel'stvo MSTU N.E. Bauman, 2014. 448 p.

[9] Lebedev B.K., Lebedev O.B. Bioinspirirovannyye metody planirovaniya kristalla SBIS (Bioinspired VLSI Crystal Planning Methods) // VI Vserossiyskaya nauchno-tekhnicheskaya konferentsiya «Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem». Sbornik trudov. M.: IPPM RAN, 2012. pp.171-176.

[10] Lebedev O.B. Modeli adaptivnogo povedeniya murav'inoy kolonii v zadachakh proyektirovaniya (Models of adaptive behavior of an ant colony in design problems). Taganrog: Izd-vo YUFU, 2013. 199 p.

[11] Lebedev B.K., Lebedev V.B. Planirovaniye na osnove royevogo intellekta i geneticheskoy evolyutsii (Planning based on swarm intelligence and genetic evolution) // Izvestiya Yuzhnogo federal'nogo univrsiteta. Tekhnicheskiye nauki. № 4, 2009. pp. 25–33.

[12] Lebedev B.K., Lebedev V.B. Optimizatsiya metodom kristallizatsii rossypi al'ternativ (Optimization by the method of crystallization of placers of alternatives) // Izvestiya YUFU. Izd-vo TTI YUFU, №7, 2013. pp. 11-17.

[13] Lebedev B.K., Lebedev V.B. Metod kristallizatsii rossypi al'ternativ (The method of crystallization of placers of alternatives). Sbornik nauchnykh trudov VII Mezhdunarodnoy nauchno-prakticheskoy konferentsii "Integrirovannyye modeli i myagkiye vychisleniya v iskusstvennom intellekte". M.: Izd-vo Fizmatlit, 2013.P. 903-912.

[14] Cong J., Romesis M., Xie M. UCLA Optimality Study Project. http://cadlab.cs.ucla.edu/~pubbench. 2004.

[15] MCNC Electronic and Information Technologies (Online). Available: www.mcnc.org.

[16] hMetis http://www-users.cs.umn.edu/karypis/metis/ hmet300. HB Floorplan Benchmarks [Online]. Available: http://cadlab.cs.ucla.edu/ cpmo/HBsuite.html.

[17]Ma Q., Young E.F. Multivoltage Floorplan Design // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, №4. 2010. pp. 607–617.

[18] Eroshenko I.N. Razrabotka geneticheskogo algoritma klasternogo planirovaniya SBIS (Development of a genetic algorithm for cluster planning VLSI) // Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskiye nauki, №7. 2010. pp. 54-60.

[19] Kureichik V.M., Lebedev B.K., Lebedev O.B. Hybrid evolutionary algorithm of planning VLSI // Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference GECCO-2010. Assoc. Comput. Mach., Spec. Interest Group Genet., Evol. Comput. (ACM SIGEVO). Portland, OR, 2010. pp. 821-822.

[20] Lebedev O.B. Planirovaniye SBIS na osnove metoda murav'inoy kolonii (VLSI planning based on the ant colony method) // Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskiye nauki. №7, 2010. pp. 67-73.