УДК 681.3

Алгоритмы эволюционного роевого интеллекта в решении задачи разбиения графа

В.М. Курейчик, А.А. Кажаров

Технологический институт Южного федерального университета в г. Таганрог,

kur@tsure.ru, persianland@mail.ru

Аннотация **- В настоящее время методы, инспирированные природными системами, широко применяются практически во всех отраслях науки и техники. Это связано с тем, что природа за миллионы лет эволюции выработала эффективные принципы и технологии оптимизации, использование которых в технических системах позволяет принимать эффективные решения. Одной из перспективных технологий является метод роевого интеллекта. Он описывает коллективное поведение децентрализованной самоорганизующейся системы. Структурная схема роевого интеллекта может быть описана в виде графа или гиперграфа, как правило, состоящего из множества агентов, локально взаимодействующих между собой и с окружающей средой. Сами агенты обычно довольно просты, но все вместе, локально взаимодействуя, создают так называемый «роевой интеллект». В работе исследованы и проанализированы алгоритмы роевого интеллекта для решения задачи компоновки блоков ЭВА, планирования СБИС. Экспериментальные исследования показали эффективность роевых алгоритмов для решения оптимизационных задач по сравнению со стандартными итерационными, эвристическими и генетическими алгоритмами.**

Ключевые слова **– роевой интеллект, разбиение графа, муравьиный алгоритм, пчелиный алгоритм, компоновка блоков ЭВА.**

I. Введение

В статье представлены результаты исследований и разработок различных роевых алгоритмов для решения задачи разбиения графа. Данная задача является NP-полной задачей, а, значит, решение в разумные сроки невозможно для задач размерностью более 20-30 вершин в графе. В связи с этим решение задачи, как правило, сводится к поиску квазиоптимального решения на основе вероятностно-направленных алгоритмов. Одними из таких являются алгоритмы роевого интеллекта. Понятие роевого интеллекта (Swarm intelligence) был введен Херардо Бени и Ван Цзином в 1989 году [1]. Под роевым интеллектом понимают самоорганизующуюся систему, состоящую из множества агентов. Агенты подчиняются простым правилам поведения в окружающей среде. Их простое взаимодействие определяет коллективную адаптацию. Таким образом, на поведении простых агентов формируется роевой интеллект. Примерами таких систем могут быть муравьиная колония, пчелиный рой, стая птиц, рыб и т.д. В работе рассмотрены следующие алгоритмы роевого интеллекта: муравьиный алгоритм, пчелиный алгоритм, алгоритм роя частиц. Данные алгоритмы были реализованы для решения различных графовых NP-полных задач, проведены экспериментальные исследования и сравнения. Решение задачи разбиения графа имеет широкое применение в области микроэлектроники, САПР СБИС. В частности, к этой задаче сводят компоновку блоков ЭВА.

II. Муравьиный алгоритм

Данный класс алгоритмов разрабатывался в рамках научного направления, которое можно назвать «природные вычисления» [2]. Исследования в этой области начались в середине 90-х годов XX века, автором идеи является Марко Дориго [3, 4]. В основе этой идеи лежит моделирование поведения колонии муравьев. Колония муравьев представляет собой систему с очень простыми правилами автономного поведения особей. Однако, несмотря на примитивность поведения каждого отдельного муравья, поведение всей колонии оказывается достаточно разумным [5]. Основой поведения муравьиной колонии служит низкоуровневое взаимодействие, благодаря которому, в целом, колония представляет собой разумную многоагентную систему. Взаимодействие определяется через специальное химическое вещество – феромон, откладываемый муравьями на пройденном пути. При выборе направления движения муравей исходит не только из желания пройти кратчайший путь, но и из опыта других муравьев, информацию о котором получает непосредственно через уровень феромонов на каждом пути. Концентрация феромона определяет желание особи выбрать тот или иной путь [6-8].

При таком подходе неизбежно попадание в локальный оптимум, т.к. на начальном этапе выбор агентами направления движения носит случайный характер. Эта проблема решается благодаря испарению феромонов, которое является отрицательной обратной связью. Испарение происходит равномерно по всему ареалу, ограничивая тем самым уровень концентрации феромо-

нов и обратную связь. Таким образом, путь, на который было потрачено меньше времени, менее подвержен испарению, а значит, концентрация феромонов на оптимальном маршруте будет дольше сохраняться [6].

Разработаны модификации муравьиного алгоритма [7], на основе которых получены более качественные решения для задачи поиска минимального гамильтонова цикла на тестах Эйлона для 50, 75 и 98 вершин по сравнению с известными [7]. Для теста Эйлона с 30 вершинами оптимальное решение находится за 1-2 секунды.

III. Пчелиный алгоритм

Идея пчелиного алгоритма заключается в том, что все пчёлы (агенты) на каждом шаге будут выбирать как элитные участки для исследования, так и участки в окрестности элитных [9]. Приведем основные понятия пчелиного алгоритма:

1. источник нектара (цветок, участок);
2. фуражиры (рабочие пчелы);
3. пчелы-разведчики.

Источник нектара характеризуется значимостью, определяемой различными параметрами. Фуражиры закреплены за источниками нектара. Количество всех пчел в этих участках больше, чем на остальных. Среднее количество разведчиков в рое составляет 5-10%. Вернувшись в улей, пчелы «обмениваются информацией» посредством танцев на, так называемой, закрытой площадке для танцев [10]. Если разведчики нашли лучшие источники нектара, то за ними могут быть закреплены фуражиры.

Рассмотрим все эти определения на примере задачи поиска глобального минимума функции:

$$f(x, y) = x^2+y^2,$$

модель которого приведена на рисунке 1.

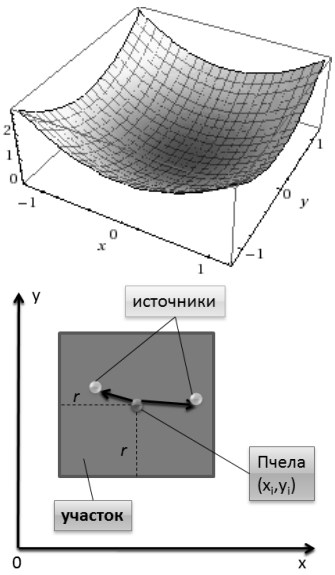

Рис. 1. Модель участка в пространстве поиска

При решении задач нахождения глобальных экстремумов сложных многомерных функций границы участка определяются как $[x_1-r; x_1+r][x_2-r; x_2+r]\ldots[x_n-r; x_n+r]$, где x_i – координата пчелы (параметр функции), r – радиус (размер) участка, n – количество параметров функции, в данном случае – 2 (x, y). Пусть решение представляет собой вектор H. Областью поиска нектара для пчел будет являться пространство поиска решений, размерностью $n!$ (количество всех возможных перестановок вектора H). Расположение источника нектара характеризуется конкретной перестановкой H, решением. Таким образом, координатами источника является решение H. Количество нектара на источнике обратно пропорционально целевой функции (ЦФ). Участок имеет размеры, где размер – количество решений «близких» к H. Близость между векторами определяется значением расстояния Хемминга между ними. К примеру, решения {5,2,7,3,4,1,6} и {5,4,7,3,2,1,6} являются «близкими», т.е. в пространстве поиска они располагаются рядом, находятся на одном «участке». Рассмотрим пример перемещения пчелы для данной задачи на рисунке 2.

i	1	2	3	4	5	6
A_i	4	2	5	6	3	1
B_i	4	6	5	2	3	1

Рис. 2. Модель участка в пространстве поиска для задачи разбиения графа

Рассмотрим кодирование решения для задачи разбиения. Входные данные: число вершин n = 13, мощности подграфов M = {4, 5, 4}. Решение представляется в виде перестановки вершин A_i, причем первые M_1 вершин относятся к первому подграфу, следующие M_2 вершин относятся ко второму подграфу и т.д., как показано в табл. 1.

Таблица 1

Кодирование решения для задачи разбиения

i	1	2	3	4	5	6	7	8	9	10	11	12	13
	подграф №1				подграф №2					подграф №3			
A_i	4	5	8	7	13	9	1	2	11	3	12	10	6

Значение целевой функции, т.е. количество межузловых связей, для данного решения равно 8. На рис. 3 представлено графическое решение.

Определим основные положения пчелиного алгоритма.

1. Пространство поиска нектара будет представлять собой всевозможные перестановки A из n чисел, а его размерность равна $n!$.

2. Количество нектара в источнике *A* обратно пропорционально ЦФ *F(A)*.
3. Источник нектара определяется одним решением из пространства поиска и представляется в виде некоторой перестановки.
4. Два источника (*A, B*) находятся на одном участке, если расстояние $L(A, B) \le r$, где r – заданный параметр ПА, размер участка.
5. Расстояние L между перестановками *A* и *B* определяется как расстояние Хемминга и равно числу различий в позициях между ними.

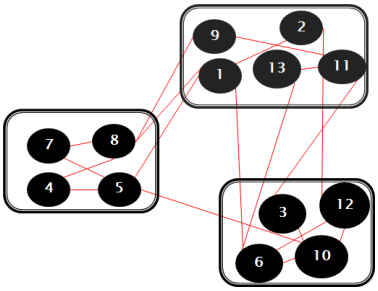

Рис. 3. Кодирование решения

Рассмотрим на примере вычисление расстояния L между источниками (табл. 2). Пусть $r = 4$.

Таблица 2

Пример вычисления расстояния Хемминга

i	1	2	3	4	5	6	7	8	9	10	11	12	13
	подграф №1				подграф №2					подграф №3			
A_i	4	5	8	7	13	9	1	2	11	3	12	10	6
B_i	4	11	8	7	13	9	1	2	3	5	12	10	6
$A_i \neq B_i$	0	1	0	0	0	0	0	0	1	1	0	0	0
$L(A,B)$	3												

Серым цветом выделены отличия в решениях *A* и *B*. Как видно из таблицы, расстояние Хемминга между решениями равно 3, т.е. $L(A, B) \le r$. Значит, «источники нектара» *A* и *B* находятся в одном участке.

IV. Метод роя частиц

Метод роя частиц (МРЧ, Particle Swarm Optimization – PSO) — метод численной оптимизации, для использования которого не требуется знать точного градиента оптимизируемой функции. МРЧ был доказан Кеннеди, Эберхартом и Ши [11, 12] и изначально предназначался для имитации социального поведения. Обширное исследование приложений МРЧ сделано Поли [13, 14]. МРЧ оптимизирует функцию, поддерживая популяцию возможных решений, называемых частицами, и перемещая эти частицы в пространстве решений согласно простой формуле. Перемещения подчиняются принципу наилучшего найденного в этом пространстве положения, которое постоянно изменяется при нахождении частицами более выгодных положений [15].

В МРЧ агентами являются частицы в пространстве параметров задачи оптимизации. В каждый момент времени частицы имеют в этом пространстве некоторое положение и вектор скорости. Для каждого положения частицы вычисляется соответствующее значение целевой функции, и на этой основе по определенным правилам частица меняет свое положение и скорость в пространстве поиска [16,17].

На каждой итерации в классическом методе роя частиц выполняются следующие операции:

$$X_{i,t+1} = X_{i,t} + V_{i,t} \quad (4)$$

$$V_{i,t+1} = V_{i,t} + U[0,\beta] * (x^b_{i,t} - x_{i,t}) \quad (5)$$

где t – момент времени, номер итерации;

x_i – вектор координаты i-й частицы;

x^b – вектор координаты лучшей частицы;

$U[0,\beta]$ – вектор псевдослучайных чисел в интервале $U[0,\beta]$;

V_i – скорость i-й частицы.

Представленный алгоритм был реализован для решения задачи планирования СБИС, разбиения графа. На рис. 4 показан пример работы метода роя частиц в задаче планирования СБИС. Критерий – площадь кристалла, цель оптимизации – минимизация критерия. Для кодирования решения используется алгоритм гильотинного разреза. Решение состоит из двух векторов: один задает порядок разреза и расположения элементов (вертикальный или горизонтальный), а второй – расположение элементов. Пример кодирования приведен в табл. 3.

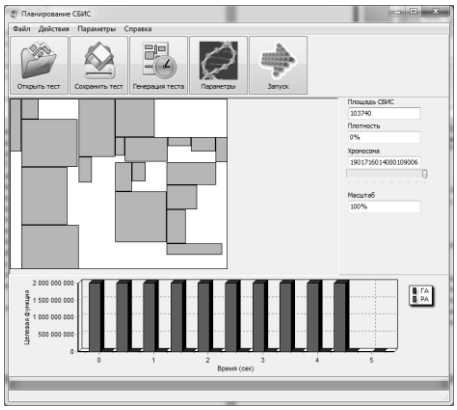

Рис. 4. Пример планирования СБИС на основе метода роя частиц

Таблица 3

Пример кодирования

i	1	2	3	4	5	6	7	8	9	10
A_i	1	3	7	1	3	2	2	8	6	
B_i	4	8	7	1	9	3	2	5	6	10

Вектор A задает порядок разреза и согласно алгоритму гильотинного разреза имеет длину, равную n-1, где n – количество разрезаемых элементов. Это связано с древовидным представлением разрезов, где в узлах порядок разреза, а в листьях элементы СБИС. A_i принимает значения от 1 до 8. По значению A_i можно определить тип разреза (вертикальное или горизонтальное), а также расположение каждого из двух элементов – горизонтально или вертикально, соответственно восемь различных комбинаций. B_i задает номер элемента.

Проведены экспериментальные сравнения с генетическим алгоритмом. Решения, полученные с помощью PSO, на 5-10% дают меньшую площадь, чем с помощью генетического алгоритма. В генетических алгоритмах применялось то же кодирование, что и для PSO, т.е. хромосома состояла из двух стрингов. Но решения, полученные для задачи разбиения, значительно хуже, чем у других эволюционных алгоритмов.

Для решения других комбинаторных задач также можно использовать кодировку, предложенную для пчелиного алгоритма. При этом скорость перемещения частицы будет прямо пропорциональна количеству изменяемых позиций. Для примера, показанного в таблице 2, скорость перемещения от точки A к B составляет 3. Т.е. чем больше скорость, тем больше позиций изменяется в решении. Скорость вычисляется по следующей формуле:

$$V_{i,t+1} = (1\text{-}c)*L(X_{i,t}, X^b) \quad (6)$$

где c – коэффициент инертности, варьирующийся от 0 до 1;

V_i – скорость i-й частицы;

$L(X_{i,t}, X^b)$ – расстояние Хемминга между i-й частицей и лучшей.

При перемещении частицы в векторе решения будут меняться $V_{i,t+1}$ позиций, причем в те, в которых есть отличия между лучшим решением и текущим i-м.

Данная кодировка и интерпретация естественного поведения роя позволяет решать многие графовые задачи, где прямое кодирование представляется в виде некоторой перестановки.

V. Экспериментальные исследования

Вышеописанные алгоритмы были разработаны и реализованы в средах разработки Code Gear C++ Builder, Microsoft Visual Studio для решения следующих задач:

1) разбиение графа;
2) раскраска графа;
3) задача коммивояжера;
4) маршрутизация автотранспорта;
5) планирование СБИС;
6) задача о паросочетаниях;
7) задача о назначениях.

Экспериментальные исследования показали эффективность алгоритмов роевого интеллекта по сравнению с генетическими и эволюционными алгоритмами [18]. На рис. 6 приведен пример решения классической задачи разбиения графа на подграфы с заданными мощностями. Для решения задачи были реализованы следующие алгоритмы: итерационный, эволюционный, генетический, муравьиный, пчелиный и метод роя частиц. Таким образом, решения, полученные различными алгоритмами, сравнивались для одной задачи с одинаковыми тестами, т.е. в равных условиях. При реализации генетического алгоритма использовались упорядочивающий кроссинговер, мутации обменом и инверсией, элитная и пропорциональная редукция.

Качество найденных решений ранжируется следующим образом (начиная с худшего): итерационный, эволюционный, МРЧ, генетический, муравьиный, пчелиный. При этом пчелиный и муравьиный алгоритмы выдают в среднем одинаковые решения. Но при этом муравьиный алгоритм является метаэвристикой и решение основано на знаниях о задаче. В отличие от него пчелиный алгоритм не использует особенностей задачи, кроме как в кодировании решении, из-за чего и выглядит предпочтительней. Т.е. пчелиный алгоритм легко адаптируется под многие задачи. При увеличении размерности задач пчелиный алгоритм по сравнению с муравьиным показывает намного лучшее решение. Это связано с тем, что временная сложность пчелиного алгоритма меньше, чем у муравьиного, и поэтому в нем удается выполнить больше итераций. Высокая степень стохастичности в алгоритмах роевого поиска позволяет быстрее выходить из локальных оптимумов по сравнению с ГА и алгоритмом отжига. Итерационный и эволюционный алгоритмы сравнительно чаще попадают в состояние стагнации. Тестирования проводились для графов размерностью от 50 до 2000 вершин.

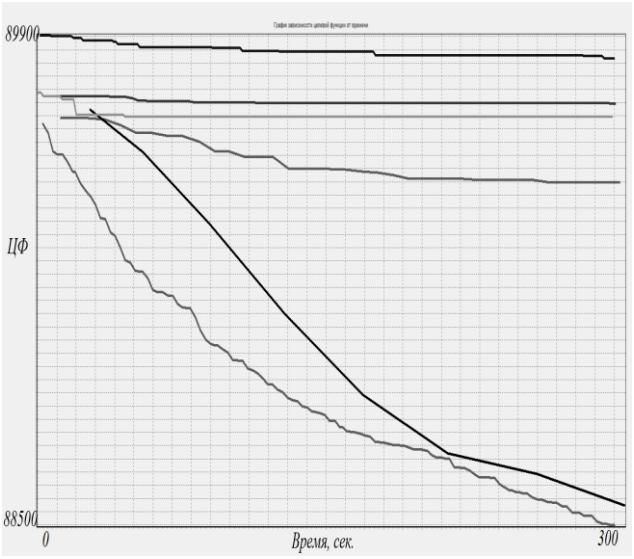

Рис. 5. Разбиение графа, графики зависимости целевых функций от времени для итерационного, эволюционного, генетического, муравьиного и пчелиного алгоритмов

На рис. 5 приведен один из тестов. Количество вершин в графе – 1000, число подграфов – 10, мощность каждого подграфа – 100 вершин, время работы для всех алгоритмов – 100 секунд. Критерий оптимизации – число внешних ребер. Цель задачи – минимизация критерия. Итерационным алгоритмом получено решение с целевой функцией равной 89817, для эволюционного – 89644, генетического – 89347, муравьиного – 88123, пчелиного – 88048, метод роя частиц - 89595. Как видно на графике рис. 5, наилучшие решения при решении задачи разбиения получаются с помощью муравьиного и пчелиного алгоритмов.

Также экспериментальные сравнения были проведены с алгоритмами hMetis и его модификациями (shMetis, khMetis) [19], впервые представленными в университете Миннесоты в 1998 году. В табл. 4 приведены значения целевых функций для разных алгоритмов на 5 различных тестах:

Таблица 4

Экспериментальные данные

Число вершин	Число подграфов	Результат МА	Результат ПА	hMetis
50	5	154	156	306
100	10	815	824	1510
500	5	19681	19401	35768
2000	20	16696	19129	30520
2000	2	-	9893	13676

Максимальное время работы муравьиного (МА) и пчелиного алгоритмов (ПА) на данных тестах 750 секунд. Максимальное время работы алгоритма hMetis занчительно ниже – 15 секунд. На графах большей размерности (более 10000 вершин) лучшие результаты ожидаемо выдает hMetis. Это связано с низкой временной сложностью алгоритма (ВСА), в то время как ПА и МА имеют большую ВСА. Из-за высокой ВСА предложенные алгоритмы едва ли успеют провести несколько итераций для больших графов. Для увеличения скорости работы алгоритмов необходимо снизить такие параметры как размер колонии, другие эвристические параметры, что влечет за собой снижение качества решения. Таким образом, предложенные алгоритмы следуют использовать для укрупненных графов при решении задачи компоновки блоков ЭВА большой численности. Для укрупнения графа можно использовать такие рекурсивные алгоритмы как Кернигана-Лина, hMetis [20].

VI. Заключение

В ходе проделанной работы была создана программа на ЭВМ, реализующая различные модели и подходы методов роевого интеллекта. Экспериментальные исследования подтвердили эффективность методов роевого интеллекта и разработанной кодировки решения как для пчелиного алгоритма, так и для муравьиного, по сравнению со стандартными эволюционными подходами. Для многих графовых NP-полных задач предпочтительным является использование пчелиного алгоритма для их решения, так как в отличие от муравьиного он не использует эвристик, основанных на знаниях о задаче. Предложенная кодировка может быть использована и для других задач, где решение представляется в виде перестановки.

Литература

[1] Beni G., Wang, J. Swarm Intelligence in Cellular Robotic Systems // Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany. Italy, June 26–30 (1989).

[2] Штовба С.Д. Муравьиные алгоритмы// Exponenta Pro. Математика в приложениях, 2003, №4. – С. 70-75.

[3] Bonavear F., Dorigo M. Swarm Intelligence: from Natural to Artificial Systems. Oxford university Press. 1999.

[4] Corne D., Dorigo M., Glover F. New Ideas in Optimization. McGrav-Hill. 1999.

[5] МакКоннелл Дж. Основы современных алгоритмов. М.: Техносфера, 2004.

[6] Кажаров А.А., Курейчик В.М. Муравьиные алгоритмы для решения транспортных задач // Теория и системы управления. М.: Наука, 2010. № 1.

[7] Курейчик В.М., Кажаров А.А. О некоторых модификациях муравьиного алгоритма // Известия ЮФУ. Технические науки. 2008.

[8] Балюк Л.В., Курейчик В.В. Вероятностный генетический алгоритм Ant Colony и его приложение для решения задач // Труды Международных научно-технических конференций «Интеллектуальные системы '04» (AIS'04) и «Интеллектуальные САПР - 2004» (CAD-2004). М.: Изд-во Физматлит, 2004. С. 53-65.

[9] Курейчик В.В., Полупанова Е.Е. Эволюционная оптимизация на основе алгоритма колонии пчёл. 2009.

[10] Субботин С.А., Олейник Ан.А., Олейник Ал.А. PSO-метод // Интеллектуальные мультиагентные методы (Swarm Intelligence). 2006. № 3. С. 55-70.

[11] Kennedy J. Particle Swarm Optimization // Proceedings of IEEE International Conference on Neural Networks IV: 1942-1948.

[12] Y. Shi, R.C. Eberhart. A modified particle swarm optimizer // Proceedings of IEEE International Conference on Evolutionary Computation: 69-73.

[13] Poli, R. (2007). An analysis of publications on particle swarm optimisation applications // Department of Computer Science, University of Essex, UK.

[14] Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation // Journal of Artificial Evolution and Applications: 1-10. DOI:10.1155/2008/685175.

[15] Mendes R, Kennedy J., Neves J. Watch thy neighbor or how the swarm can learn from its environment // Proceedings of Swarm Intelligence Symposium 2003. IEEE, 2003. P. 88-94.

[16] Kennedy J., Mendes R. Population structure and particle swarm performance // Proceedings of the 2002 Evolutionary Computation Congress. Washington, IEEE Computer Society. P. 1671-1676.

[17] Гальченко В.Я., Якимов А.Н. Использование эволюционного метода роевого интеллекта для решения задач принятия решений // Искусственный интеллект. Интеллектуальные системы ИИ-2010: Материалы международной научно-технической конференции. Донецк: ИППИ «Наука і освіта». 2010. Т. 2. 368с.

[18] Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы.– М.: ФИЗМАТЛИТ, 2006г.

[19] G. Karypis, V. Kumar. Multilevel k-way hypergraph partitioning. Technical Report TR 98-036, Department of Computer Science, University of Minnesota. 1998.

[20] G. Karypis, R. Aggarwal, V Kumar, S. Shekhar, Multilevel Hypergraph Partitioning: Applications in VLSI Domain // IEEE Trans. VLSI Syst. March 1999. Vol. 7. № 1. P. 69-79.