Двухуровневый биоинспирированный алгоритм размещения фрагментов СБИС

Д.Ю. Запорожец[1], Д.В. Заруба[2], Н.В. Кулиева[3]
1Южный Федеральный Университет
[1]elpilasgsm@gmail.com, [2]daria.zaruba@gmail.com,[3]holopova@sfedu.ru

***Аннотация*. В работе рассматривается одна из основных задач конструкторского проектирования СБИС – размещение фрагментов СБИС. Данная задача является NP-полной и не имеет детерминированного алгоритма решения. В связи с этим разработка новых методов поиска является актуальной задачей. Среди всего множества вероятностных методов блок биоинспирированных методов уже показал свою эффективность при решении комбинаторных оптимизационных задач. В статье представлен двухуровневый биоинспирированный алгоритм решения задачи размещения фрагментов СБИС. Предлагается подход, основанный на применении пчелиного алгоритма на первом уровне для диверсификации поиска и параллельный генетический алгоритм на втором уровне для обеспечения поиска в глубину в окрестностях точек. Приведены постановка задачи и подробное описание шагов алгоритма. Проведена серия вычислительных экспериментов, в рамках которой эмпирически была подтверждена эффективность предложенного подхода на основе бэнчмарок фирмы IBM. Сравнение проводилось с известными аналогичными алгоритмами размещения фрагментов СБИС: Capo 8.6, Feng Shui 2.0, Dragon 2.23. Прирост качества полученных решений в среднем составил 19%.**

Ключевые слова: биоинспирированный алгоритм, параллельные вычисления, конструкторское проектирование

I. Введение

Основой научно-технического прогресса является широкое использование электронно-вычислительной аппаратуры (ЭВА) во всех областях техники и народного хозяйства [1,2]. Это сложный комплекс устройств вычислительной техники, предназначенный для электронной обработки информации и отображении ее в форму, удобную для восприятия пользователем. В настоящее время во всем мире наблюдается резкое увеличение производства такой аппаратуры, повышение ее «интеллектуальности», быстродействия, объемов хранимой информации. В этих условиях неизбежно возрастает сложность создаваемой ЭВА [1-3]. В настоящее время при проектировании ЭВА до 70% усилий затрачивается на разработку СБИС. Проектирование СБИС сложный процесс, занимающий много времени. В последние время, в связи с постоянно растущей конкуренцией, необходимо сокращать время и затраты на проектирование. Одним из путей для достижения этих целей является применение САПР разного уровня, что позволит сократить временные и материальные затраты, а также повысить качество, проектируемых СБИС. Совершенно очевидно, что производство СБИС невозможно без использования этапа автоматизированного конструкторского проектирования. Конструкторское проектирование включает в себя много этапов. Все задачи, выполняемые на этих этапах, относятся к NP-трудным и NP-полным. Среди них размещение фрагментов СБИС является наиболее важным этапом, т.к. необходимо производить обработку огромных массивов информации [1-4]. В этой связи становится необходимым модернизация структуры и основных методов автоматизированного конструкторского проектирования. Одним из подходов является использование гибридных и комбинированных стратегий. В работе для сокращения размерности задачи предлагается гибридный подход на основе двух методов - пчелином и генетическом поиске.

Работа выполнена при финансовой поддержке гранта Президента Российской Федерации для государственной поддержки молодых российских ученых - кандидатов наук № МК-92.2017.8.

II. Постановка задачи

При решении задачи размещения фрагментов СБИС в общем случае исходными данными являются:

- монтажная плоскость;
- схема электрическая принципиальная;
- множество фрагментов СБИС;
- набор связей между фрагментами (список цепей).

Пусть $X_1, \ldots, X_n$ размещаемые на коммутационном поле элементы СБИС. Каждый элемент $X_i | 1 \le i \le N$, имеет следующие свойства: высота hi, ширина w_i. $N = \{n_i \mid i = 1,m\}$ – список цепей, соединяющих элементы. Задача размещения заключается в нахождении прямоугольных областей на коммутационном поле для каждого размещаемого элемента из X и определяется множеством непересекающихся областей $R = \{r_i \mid i = 1, n\}$ таких, что каждый блок располагается в соответствующей прямоугольной области r_i.

$$r_i =< u_i, t_i, w_i, h_i >,$$

Каждая цепь n_i – представляется в виде кортежа областей из множества R.

$$n_i = \langle R_{n_i} \rangle, R_{n_i} \subseteq R,$$

где R_{n_i} – область из R, входящая в состав цепи n_i.

В качестве критерия размещения фрагментов СБИС предлагается использовать значение полупериметра описывающего прямоугольника цепи.

$$F(R) = \sum_{i=1}^{m} \Big(\big(uMax(\langle R_{n_i} \rangle) - uMin(\langle R_{n_i} \rangle) \big) + \big(tMax(\langle R_{n_i} \rangle) - tMin(\langle R_{n_i} \rangle) \big) \Big),$$

где $uMax(\langle R_{n_i} \rangle)$ и $uMin(\langle R_{n_i} \rangle)$ координата по оси X крайне правого и крайне левого элементов, входящих в состав цепи n_i, $tMax(\langle R_{n_i} \rangle)$ и $tMin(\langle R_{n_i} \rangle)$ координата по оси Y крайне верхнего и крайне нижнего элементов, входящих в состав цепи n_i. Данный подход позволяет сократить время и ресурсы, требуемые на расчет целевой функции для каждого решения [5,6].

III. Архитектура гибридного поиска

Одним из основных путей уменьшения сложности задач размещения является сокращение их размерности. На рис. 1 представлена архитектура двухуровневого подхода на основе алгоритма пчелиной колонии и генетическом алгоритме [3,7].

В данной задаче набор параметров представляется в виде альтернативного решения (хромосомы), причем число возможных альтернатив размещения в общем случае составляет N!.

Рис. 1. Архитектура двухуровневого подхода

Опишем комбинированный подход более подробно. С помощью жадной эвристики получаем решение, на основе которого генерируем множество альтернативных решений (популяция) S и с заданной мощностью (размер популяции) |S|. Для каждого решения рассчитывается значение целевой функции (ЦФ) F(R). Производится сортировка популяции по возрастанию значения ЦФ. На основании эвристики поведения колонии пчел выбираем заданное количество M_{search} альтернативных решений таких, что разность значений ЦФ любых двух выбранных решений не менее заданного значения ΔF_{fixed}.

$$\forall i \text{ и } \forall j, \text{где } i \neq j, 0 < j < |S|,$$

$$0 < i < |S|, \exists s_i \text{ и } \exists s_j, \text{для которых}$$

$$|F(s_i) - F(s_j)| > \Delta F_{fixed} \quad (1)$$

где s_i, s_j альтернативные решения из популяции S, $F(s_i)$ и $F(s_j)$ – значения ЦФ соответствующих альтернативных решений.

Таким образом мы получаем множество решений, удаленных друг от друга не менее чем на заданную величину в пространстве поиска, что позволяет соблюдать принцип диверсификации. Далее для каждой из выбранных решений параллельно друг от друга инициализируются генетические алгоритмы с заданными параметрами размера популяции и количества итераций, с помощью которого осуществляется поиск в глубину в окрестностях предложенных точек. Данный подход позволяет соблюдать принцип интенсификацией поиска.

IV. «Жадный» алгоритм размещения для инициализации начальной популяции решений.

«Жадная» эвристика позволяет найти решение сопоставимое по качеству с оптимальным за линейное время. Её основные шаги состоят в следующем:

1. Для очередной цепи n_i выбираются все входящие в ее состав фрагменты СБИС, удаляются все уже размещённые фрагменты. Если в результате удаления получается пустой список, то процесс останавливается. Все элементы размещены.
2. Выбирается очередной фрагмент из списка и размещается в очередную свободную позицию. Размещение элементов осуществляется построчно, начиная с нижнего левого угла. Далее, после заполнения текущей строки, размещение продолжается с новой.
3. После того, как все элементы из списка размещены процесс переходит к п.1.

Очевидно, что если при использовании «жадной» эвристики изменить очередность рассмотрения цепей, а также очередность рассмотрения фрагментов внутри списков, то полученные решения будут иметь разные значения целевой функции. Это значит, что для получения популяции размером |S| необходимо |S| раз перемещать список цепей и после каждого «взбалтывания» запустить «жадный» алгоритм размещения [8].

V. Выбор окрестности поиска.

Как описывалось ранее, множество S сортируется по возрастанию и выбирается M_{search} решений в соответствии с неравенством (1). Если, в следствии вырождения популяции, достигнуть заданного количества M_{search} невозможно, то в качестве M_{search} принимается текущее максимально возможное количество найденных решений. В результате получаем множество S^0, где $|S^0| = M_{search}$ [9].

VI. Генетический поиск

На данном этапе в качестве оптимизационного алгоритма используется широко известный простой

генетический алгоритм (ПГА). Схема работы данного алгоритма представлена на рис. 2. Для каждого элемента $S^0{}_i$ из множества S^0 инициализируется ПГА [10, 11].

Рис. 2. Схема работы простого генетического алгоритма

Начальная популяция создается методом фокусировки, реализация которого заключается в том, что каждая хромосома начальной популяции отличается от другой не более чем на заданное количество парных перестановок Δ_{init}. Таким образом становится возможным контролировать размер окрестности вокруг решения $S^0{}_i$.

VII. Экспериментальные исследования

В рамках данной работы вышеописанный алгоритм был реализован на языке Java. В качестве тестовых схем использовались известные бенчмарки фирмы IBM [12]. Целью проведения экспериментов является исследования качества решений, полученных при использовании разработанного двухуровневого алгоритма. Сравнение будет проводиться с результатами работы известных алгоритмов: Capo 8.6 [13,14], Feng Shui 2.0 [14,15], Dragon 2.23 [15,16]. Исследования проводились на двух различных платформах с процессорами: Intel I7 – 4 ядра, 2.7 ГГц и AMD 6000+, - 2 ядра – 2.5 ГГц. Данные экспериментальных исследований сведены в таблицу 1. Результаты сравнения качества полученных решений представлены на рис. 3 и 4.

ТАБЛИЦА I Результаты экспериментальных исследований

Тестовые схемы		**Capo 8.6**	**Feng Shui 2.0**	**Dragon 2.23**	**2х-уровневый алгоритм**						
Название	***Число элементов***	***длина, м***	***длина, м***	***длина, м***	***2х-уровневый алгоритм***	***приращение длины (Capo 8.6), м***	***приращение длины (Capo 8.6), %***	***приращение длины (Feng Shui 2.0), м***	***приращение длины (Feng Shui 2.0), %***	***приращение длины (Dragon 2.23), м***	***приращение длины (Dragon 2.23), %***
ibm01	12752	4,97	4,87	4,42	4,2	-0,77	-18,33%	-0,67	-15,95%	-0,22	-5,24%
ibm02	19601	15,23	14,38	13,57	13,03	-2,2	-16,88%	-1,35	-10,36%	-0,54	-4,14%
ibm03	23136	14,06	12,84	12,33	11,57	-2,49	-21,52%	-1,27	-10,98%	-0,76	-6,57%
ibm04	27507	18,13	16,69	15,41	13,2	-4,93	-37,35%	-3,49	-26,44%	-2,21	-16,74%
ibm05	29347	44,73	37,3	36,38	35,98	-8,75	-24,32%	-1,32	-3,67%	-0,4	-1,11%
ibm06	32498	21,96	20,27	20,38	18,77	-3,19	-17,00%	-1,5	-7,99%	-1,61	-8,58%
ibm07	45926	36,06	31,5	29,97	27,98	-8,08	-28,88%	-3,52	-12,58%	-1,99	-7,11%
ibm08	51309	37,89	34,14	32,2	29,12	-8,77	-30,12%	-5,02	-17,24%	-3,08	-10,58%
ibm09	53395	30,28	29,86	28,1	25,96	-4,32	-16,64%	-3,9	-15,02%	-2,14	-8,24%
ibm10	69429	61,25	57,99	57,2	53,98	-7,27	-13,47%	-4,01	-7,43%	-3,22	-5,97%
ibm11	70558	46,45	43,28	40,77	36,4	-10,05	-27,61%	-6,88	-18,90%	-4,37	-12,01%
ibm12	71076	81,55	75,91	71,03	68,3	-13,25	-19,40%	-7,61	-11,14%	-2,73	-4,00%
ibm13	84199	56,47	54,09	50,57	45,12	-11,35	-25,16%	-8,97	-19,88%	-5,45	-12,08%

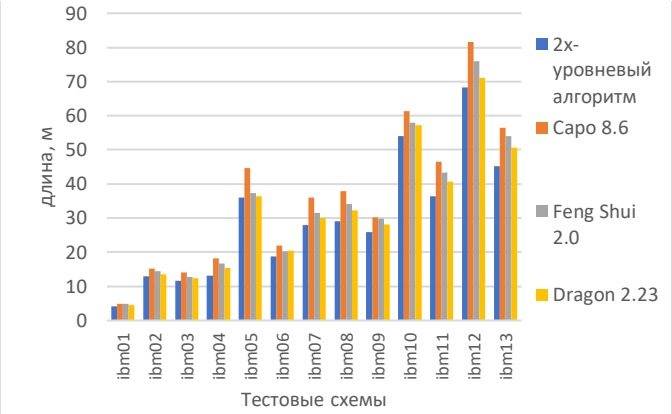

Рис. 3. График зависимости качества решений от используемого алгоритма

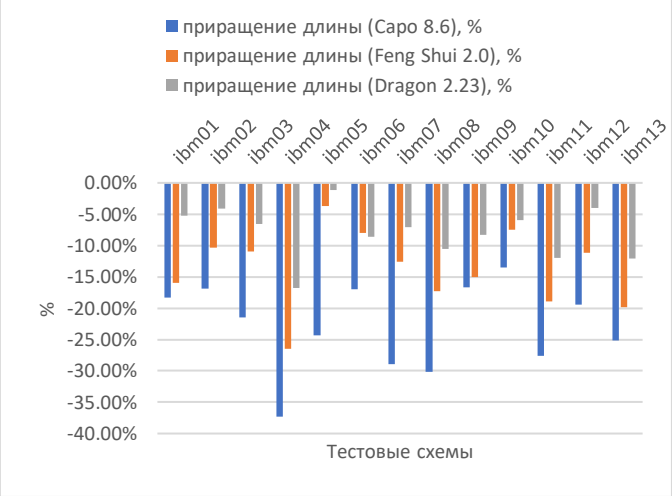

Рис. 4. Гистограмма зависимости приращения значения ЦФ 2x-уровневого алгоритма от используемого аналога

VIII. Заключение

В работе был разработан и исследован двухуровневый биоинспирированный алгоритм решения задачи размещения фрагментов СБИС. Данный подход использует парадигмы пчелиного алгоритма на первом уровне для диверсификации поиска и параллельного генетического алгоритма на втором уровне для обеспечения принципа интенсификации поиска. В ходе проведения серии вычислительных экспериментов эмпирически была подтверждена эффективность предложенного подхода. Для определения качества полученных решений проводилось сравнение с известными аналогичными алгоритмами размещения фрагментов СБИС: Capo 8.6, Feng Shui 2.0, Dragon 2.23. В качестве тестовых схем использовались бэнчмарки фирмы IBM. Прирост качества полученных решений в среднем составил 19%.

Список литературы

[1] Жиленков М.А., Курейчик В.В. 1. Модификация вероятностного генетического алгоритма решения задачи размещения элементов ЭВА с учетом электромагнитной совместимости // Известия ЮФУ. Технические науки. 2017. №7(192). С. 6-15.

[2] V. Kureichik, V.Kureichik Jr, V.Bova. Placement of VLSI fragments based on a multilayered approach // Advances in Intelligent Systems and Computing. 2016. №464. С. 181-190.

[3] D. Zaruba, D. Zaporozhets, V. Kureichik. Artificial bee colony algorithm—A novel tool for VLSI placement // Advances in Intelligent Systems and Computing. 2016. 450. С. 433-442.

[4] Заруба Д.В., Запорожец Д.Ю. Генерация биоинспирированных поисковых процедур для решения оптимизационных задач // Известия ЮФУ. Технические науки. 2016. № 6 (179). С. 13-24.

[5] Zaporozhets, D.U., Zaruba, D.V., Kureichik, V.V. Representation of solutions in genetic VLSI placement algorithms // Proceedings of IEEE East-West Design and Test Symposium, EWDTS 2014. (2014) art. no. 7027053.

[6] Курейчик В.В., Лещанов Д.В. Комбинированный поход для решения задачи размещения фрагментов СБИС // Информатика, вычислительная техника и инженерное образование. 2017. № 1 (29). С. 1-9.

[7] Zaporozhets, D., Zaruba, D.V., Kureichik, V.V. Hierarchical approach for VLSI components placement // Advances in Intelligent Systems and Computing. 2015. №347, pp. 79-87.

[8] Kureichik, L., Kureichik, V., Jr., Kureichik, V., Leschanov, D., Zaruba, D. Hybrid approach for VLSI fragments placement // Advances in Intelligent Systems and Computing. 2018. №679, pp. 349-358.

[9] Кулиев Э.В., Запорожец Д.Ю., Ксалов А.М., Кудаев А.Ю., Коков А.А., Ошхунов М.М. Биоинспирированный поиск при решении задачи размещения компонентов СБИС // Известия Кабардино-Балкарского научного центра РАН. 2014. № 6 (62). С. 58-65.

[10] Родзин С.И., Курейчик В.В. Состояние, проблемы и перспективы развития биоэвристик // Программные системы и вычислительные методы. 2016. № 2. С. 158-172.

[11] Kureichik, V., Zaporozhets, D., Zaruba, D. Generation of bioinspired search procedures for optimization problems // Application of Information and Communication Technologies, AICT 2016 - Conference Proceedings. 2017.

[12] IBM-PLACE 2.0 benchmark suits http://er.cs.ucla.edu/benchmarks/ibm-place2/bookshelf/ibm-place2-all-bookshelf-nopad.tar.gz

[13] Caldwell A. E., Kahng A. B., Markov I. L. Can Recursive Bisection Alone Produce Routable Placements? – DAC 2000, pp.477-82.

[14] Wang M., Yang X., Sarrafzadeh M. Dragon2000: Standard-cell Placement Tool for Large Industry Circuits – ICCAD2000, pp. 260-263.

[15] Yang X., Choi B.-K., Sarrafzadeh M. Timing-Driven Placement using Design Hierarchy Guided Constraint Generation – ICCAD 2002, pp. 177-184.

[16] Agnihotri A., Yildiz M. C., Khatkhate A., Mathur A., Ono S., Madden P.H. Fractional Cut: Improved Recursive Bisection Placement – ICCAD2003.