18. *Patel M.K., Kabat M.R., Tripathy C.R.* A hybrid ACO/PSO based algorithm for QoS multicast routing problem, *Ain Shams Engineering Journal*, 2014, Vol. 5, Issue 1, pp. 113-120.
19. OR-Library is collection of test data for a variety of OR problem. Available at: http://mscmga.ms.ic.ac.uk.
20. *Warme D.M.* A new exact algorithm for rectilinear Steiner trees. INFORMS Conf., San-Diego, California, 1997.
21. *Andrew B.K., Mandoiu I.* RMST-Pack: Rectilinear minimum spanning tree algorithms. Available at: http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST/.
22. *Chen H., Qiao C., Zhou F., and Cheng C.-K.* Refined single trunk tree: A rectilinear Steiner tree generator for interconnect prediction, *In Proc. ACM Intl. Workshop on System Level Interconnect Prediction*, 2002, pp. 85-89.
23. *Hai Zhou.* Efficient Steiner tree construction based on spanning graphs, *In Proc. Intl. Symp. on Physical Design*, 2003, pp. 152-157.
24. *Griffith J., Robins G., Salowe J.S., and Zhang T.* Closing the gap: Near-optimal Steiner trees in polynomial time, *IEEE Trans. Computer-Aided Design*, 1994, No. 13 (11), pp. 1351-1365.
25. *Chris Chu.* FLUTE: Fast lookup table based wirelength estimation technique, *In Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design*, 2004, pp. 696-701.
26. GeoSteiner – software for computing Steiner trees http://www.diku.dk/geosteiner/.
27. *Chu C. and Wong Y.-C.* Fast and accurate rectilinear steiner minimal tree algorithm for VLSI design, *In Proc. International Symposium on Physical Design*. ACM Press, 2005, pp. 28-35.

Статью рекомендовала 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: lebedev.ob@mail.ru; тел.: 89289591426; кафедра систем автоматизированного проектирования; студентка.

Lebedev Boris Konstantinovich – Southern Federal University; e-mail: lebedev.b.k@gmail.com; 44, Nekrasovsky, Taganrog, 347928, Russia; phone: 89282897933; the department of computer aided design; professor.

Lebedev Oleg Borisovich – e-mail: lebedev.ob@mail.ru; phone: 89085135512; the department of computer aided design; associate professor.

Lebedeva Ekaterina Olegovna – e-mail: lebedev.ob@mail.ru; phone: 89289591426; the department of computer aided design; student.

УДК 519.712.2

Л.А. Гладков, Н.В. Гладкова, С.Н. Лейба

РАЗРАБОТКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ГИБРИДНОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧ РАЗМЕЩЕНИЯ И ТРАССИРОВКИ*

Рассматривается гибридный алгоритм решения задач размещения и трассировки элементов схем цифровой электронно-вычислительной аппаратуры. Отмечена актуальность и важность рассматриваемой задачи и многообразие существующих подходов к решению подобного рода задач. Приведена постановка задачи, выбраны ограничения области допустимых решений и сформулирован критерий оценки качества получаемых решений. Предложен новый гибридный подход к решению рассматриваемой задачи на основе

* Работа выполнена при финансовой поддержке РФФИ (проект № 17–01–00627).

сочетания эволюционных методов поиска, математического аппарата нечеткой логики и возможностей параллельной организации вычислительного процесса. В статье предложено производить обмен решениями между популяциями с использованием общего промежуточного буфера хромосом. Разработаны новые модификации основных генетических операторов. Предложен модифицированный оператор миграции для обмена информацией между популяциями решений в процессе выполнения параллельных вычислений. Разработана структура параллельного гибридного алгоритма. Предложена реализация модуля нечеткого управления на основе использования многослойной нейронной сети и функции Гаусса. Для повышения качества получаемых результатов в контур эволюции экспертной информации включен нечёткий логический контроллер, регулирующий значения параметров процесса эволюции. Сформулированы основные принципы работы блока нечеткого управления. Представлена структурная схема, разработанного гибридного алгоритма. Подробно рассмотрены особенности программной реализации предложенного гибридного алгоритма. Сформулированы требования к архитектуре разрабатываемой программы с учетом необходимости поддержки свойств модульности и расширяемости приложения. Приведены примеры описания элемента печатной платы на основе существующих спецификаций. Описана структура интерфейса, представлены основные элементы графического интерфейса разработанного приложения. Представлено краткое описание проведенных вычислительных экспериментов, подтверждающих эффективность предложенного метода.

Автоматизация проектирования; генетический алгоритм; эволюционные вычисления; нечеткая логика; параллельные вычисления; нечеткий логический контроллер.

L.A. Gladkov, N.V. Gladkova, S.N. Leiba

DEVELOPMENT AND PROGRAM IMPLEMENTATION OF THE HYBRID ALGORITHM SOLUTIONS OF PLACEMENT AND ROUTING PROBLEMS

The article considers a hybrid algorithm for solving the problems of placement and tracing elements of circuits of digital electronic computing equipment. The relevance and importance of the problem under consideration and the variety of existing approaches to the solution of such problems are noted. The formulation of the problem is given, the limitations of the domain of admissible solutions are chosen and the criterion for estimating the quality of the solutions obtained is formulated. A new hybrid approach to the solution of the problem under consideration is proposed on the basis of a combination of evolutionary search methods, a mathematical apparatus of fuzzy logic and the possibilities of parallel organization of the computational process. In the article, it was suggested to exchange solutions between populations using a common intermediate buffer of chromosomes. New modifications of the basic genetic operators have been developed. A modified migration operator is proposed to exchange information between solution populations in the process of performing parallel computations. The structure of the parallel hybrid algorithm is developed. The implementation of the fuzzy control module based on the use of a multilayer neural network and the Gaussian function is proposed. To improve the quality of the results obtained, a fuzzy logic controller that regulates the values of the parameters of the evolution process is included in the evolution of expert information. The basic principles of the fuzzy control unit are formulated. The structural scheme of the developed hybrid algorithm is presented. The features of the software implementation of the proposed hybrid algorithm are considered in detail. The requirements to the architecture of the developed program are formulated taking into account the need to support the modularity and extensibility of the application. Examples of the description of an element of a printed circuit board on the basis of existing specifications are given. The structure of the interface is described, the main elements of the graphic interface of the developed application are presented. A brief description of the computational experiments that confirm the effectiveness of the proposed method is presented.

Design automation; genetic algorithm; evolutionary computation; fuzzy logic; parallel computing; fuzzy logic controller.

Введение. Задачи конструкторского проектирования характеризуются большой вычислительной сложностью, обусловленной необходимостью перебора огромного числа различных вариантов решений [1]. Особую роль среди решаемых задач занимают задачи размещения и трассировки. Поэтому представляется целе-

сообразной разработка интегрированных методов решения задач размещения и трассировки, позволяющих выполнять эти задачи в одном цикле с взаимным учетом имеющихся ограничений и текущих результатов [2].

В настоящее время для решения задач с большой вычислительной сложностью активно используются гибридные системы. Они основаны на совмещении различных научных направлений, например, таких как генетические алгоритмы, нечеткие системы и нейронные сети [3–8]. Также, широкое распространение получили механизмы распараллеливания вычислений для эффективного освоения потенциала вычислительных средств, возможности которых возросли за счет параллельной организации вычислений [9–12].

Постановка задачи. Для краткости воспользуемся постановкой задачи, приведенной в работах [13, 14]. Пусть задано множество элементов E:

$$E = \{e_i \mid i = 1, \ldots, N\},$$

где e_i – размещаемый элемент, N – количество размещаемых элементов.

$$e_i = (l_i, h_i, T_i),$$

где l_i – длина элемента, h_i – высота элемента, T_i – список контактов элемента размещения.

$$T_i = \{t_j \mid j = 1, \ldots, K\},$$

где t_j – контакт, K – количество контактов элемента.

$$t_j = (x_j, y_j),$$

где x_j, y_j – координаты контакта относительно базовой точки элемента.

Множество соединяющих элементы цепей:

$$U = \{u_h \mid h = 1, \ldots, L\},$$

где u_h – цепь, L – количество цепей.

$$u_h = \{(N_{ek}, N_{ck}) \mid k = 1, \ldots, M\},$$

где N_{ek} – номер элемента, N_{ck} – номер контакта, M – количество контактов, соединяемых цепью.

Необходимо найти вариант размещения элементов на монтажном пространстве

$$V = \{(x_i, y_i) \mid i = 1, \ldots, N\},$$

где (x_i, y_i) – координаты верхнего левого угла установочной площади элемента размещения i, такие, чтобы суммарная площадь перекрытия размещенных элементов была равна нулю, а сумма значений остальных критериев минимальной.

Для каждой цепи необходимо найти список позиций коммутационного поля, через которые она проходит:

$$W_h = \{(x_q, y_q) \mid i = 1, \ldots, Q\},$$

где Q – количество позиций, через которые проходит h-я цепь.

Разработка алгоритма. Для совместного решения задач размещения и трассировки используется параллельный генетический алгоритм. Он предполагает параллельное выполнение эволюционных процессов на нескольких популяциях. Обмен особями между популяциями осуществляется через общий промежуточный буфер хромосом. Обмен выполняется при наступлении определённых асинхронных событий, которые задают точки миграции. При наступлении события в одном из процессов, к популяции применяется оператор миграции.

Оператор миграции применяется для перемещения хромосом между популяциями. Отбор особей для миграции выполняется из некоторого количества хромосом популяции, имеющих наилучшее значение ЦФ размещения. Отбор осуществляется на основе оценки количества непротрассированных соединений. Для каждого варианта размещения, описываемого хромосомой, выполняется трассировка с помощью волнового алгоритма. Затем, из популяции, к которой применяется оператор миграции, осуществляется копирование некоторого количество хромосом с наилучшим значением данного показателя в общий буфер хромосом. При этом из буфера

удаляется такое же количество хромосом с наихудшим значением ЦФ трассировки. Также осуществляется копирование наиболее приспособленных особей из буфера в текущую популяцию с вытеснением наименее приспособленных особей.

Начальная популяция задается методом дробовика. Используется селекция методом рулетки. Применяются одноточечный оператор кроссинговера и многоточечный оператор мутации, в котором количество подвергающихся мутации генов пропорционально длине хромосомы [13, 14]. На рис. 1 представлена структурная схема алгоритма, выполняемого на 2-х популяциях. На практике количество популяций может быть значительно больше.

Модуль нечеткого управления реализуется следующей функцией:

$$\overline{y} = \frac{\sum_{k=1}^{N} \overline{y}^{k} \left(\prod_{i=1}^{n} \exp\left(-\left(\frac{\overline{x}_i - \overline{x}_i^{k}}{\sigma_i^{k}} \right)^2 \right) \right)}{\sum_{k=1}^{N} \left(\prod_{i=1}^{n} \exp\left(-\left(\frac{\overline{x}_i - \overline{x}_i^{k}}{\sigma_i^{k}} \right)^2 \right) \right)},$$

где $\overline{x}_i^{k}$ – это центр, а σ_{ik} – ширина гауссовской кривой (функции принадлежности блока фаззификации), y_k – центры функций принадлежности нечетких множеств блока дефаззификации.

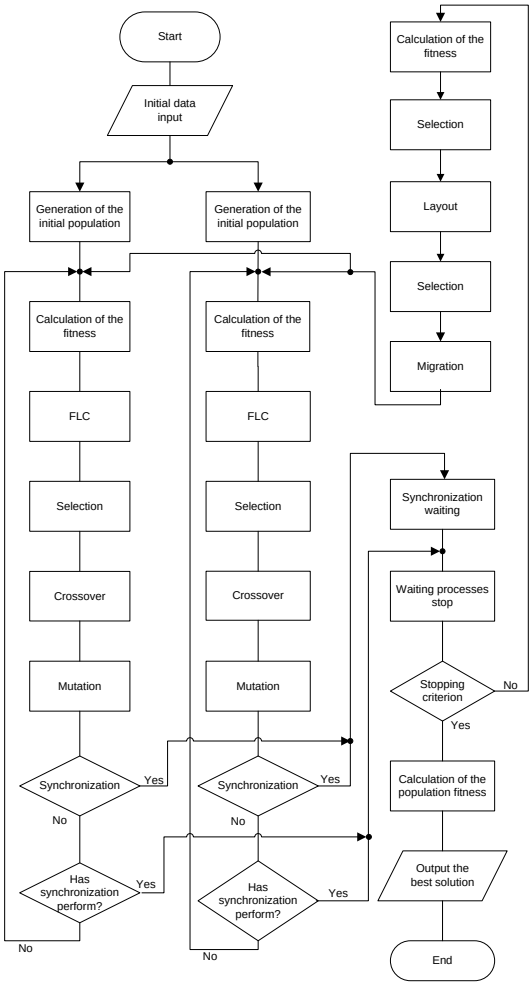

Рис. 1. Структурная схема алгоритма

Каждый элемент этой формулы можно задать в форме функционального блока (сумма, произведение, функция Гаусса), что после соответствующего объединения позволяет создать многослойную сеть. В нашем случае нейронная сеть будет содержать 4 слоя [13–15].

Для повышения качества результатов генетического поиска решается задача включения в контур эволюции экспертной информации путем построения нечёткого контроллера, регулирующего значения параметров процесса эволюции.

В качестве входных используются следующие параметры:

$$e_1(t) = \frac{f_{ave}(t) - f_{best}(t)}{f_{ave}(t)}; \quad e_2(t) = \frac{f_{ave}(t) - f_{best}(t)}{f_{worst}(t) - f_{best}(t)};$$

$$e_3(t) = \frac{f_{best}(t) - f_{best}(t-1)}{f_{best}(t)}; \qquad e_4(t) = \frac{f_{ave}(t) - f_{ave}(t-1)}{f_{ave}(t)},$$

где t – временной шаг; fbest(t) – лучшее значение ЦФ на итерации t; fbest(t-1) – лучшее значение ЦФ на итерации (t – 1); fworst(t) – худшее значение ЦФ на итерации t; fave(t) – среднее значение ЦФ на итерации t; fave(t – 1) – среднее значение ЦФ на итерации (t – 1) [13–15].

На выходе получаем вероятности применения операторов кроссинговера, мутации, а также оператора миграции.

Особенности программной реализации. Эффективная программная реализация разрабатываемых алгоритмов является важным аспектом при создании качественных, высокопрозводительных вычислительных систем. Современные языки программирования, такие, как C++, C#, Java позволяют максимально эффективно использовать потенциал аппаратных ресурсов.

Разработка архитектуры приложения велась с учетом необходимости поддержки свойств модульности и расширяемости приложения. При реализации компонентов системы применялся принцип уменьшения связности, что позволило разделить алгоритмы размещения и трассировки, синтаксический анализатор и графический интерфейс. Каждый компонент системы, при необходимости, может быть быстро заменен на более эффективный. Структура классов используемых алгоритмов позволяет дополнять систему новыми свойствами и моделями поведения. Например, наследование классов генетических операторов позволяет добавлять новые классы, каждый из которых, определяет алгоритм обработки данных [16, 17].

Для хранения данных о топологии печатной платы используется LEF/DEF спецификация. LEF (Library Exchange Format) – это спецификация для представления физической структуры интегральной схемы в формате ASCII. Она включает правила оформления и абстрактную информации об элементах. LEF используется в сочетании с DEF (Design Exchange Format) спецификацией, которая используется для представления полного размещения элементов интегральной схемы [17]. Приведем пример описания элемента печатной платы при помощи LEF спецификации.

```
MACRO ms00f80
        PROPERTY LEF58_EDGETYPE "
          EDGETYPE LEFT 2 ;
          EDGETYPE RIGHT 2 ;
        " ;
        CLASS CORE ;
        ORIGIN 0 0  ;
        SIZE 1.6 BY 2.0 ;
        SYMMETRY X Y R90 ;
```

```
    SITE  core ;
    PIN o DIRECTION OUTPUT ;
      PORT
        LAYER metal2 ;
        RECT 0.05 0.500 0.15 1.500 ;
      END
    END o
    PIN a DIRECTION INPUT ;
      PORT
        LAYER metal1 ;
        RECT 1.05 0.500 1.15 1.500 ;
      END
    END a
  END ms00f80
```

Также приведем пример описания размещения элементов на печатной плате, а также описания цепей при помощи DEF спецификации.

```
  COMPONENTS 6;
     - g2278701 ms00f80
        + PLACED (20, 10);
     - g2278702 ms00f80
        + PLACED (20, 40);
     - g2278703 ms00f80
        + PLACED (20, 70);
     - g2278704 ms00f80
        + PLACED (60, 10);
     - g2278705 ms00f80
        + PLACED (60, 40);
     - g2278706 ms00f80
        + PLACED (60, 70);
  END COMPONENTS

NETS 2;
   - ternarymux_ln49_unr9_z_9_ ( g2278701 a ) ( g2278705 o )
( g2278703 a );
   - ternarymux_ln49_unr9_z_10_ ( g2278704 o ) ( g2278702 a )
( g2278706 o );
```

Алгоритм размещения реализован классом CPlacingAlgorithm. Для выполнения алгоритма вызывается метод execute, который принимает указатель на класс CBoard, используемый для хранения топологии печатной платы (Рис. 2). Результатом работы алгоритма является обработанная топология печатной платы с заданными позициями элементов и протрассированными соединениями.

Параметры алгоритма устанавливаются с помощью метода setParams, который принимает структуру SParams, содержащую такие поля, как количество хромосом, количество итераций, вероятности кроссинговера и мутации, а также частота миграции. Генетические операторы устанавливаются при помощи указателей на абстрактные базовые классы, определяющие интерфейс операторов. Каждый указатель может указывать на конкретную реализацию оператора. Родительские и дочерние популяции хранятся в динамической памяти. Доступ к ним осуществляется с помощью векторов указателей. Промежуточный буфер хромосом также хранится в динамической памяти.

Алгоритм трассировки реализован классом CRoutingAlgorithm. Для выполнения алгоритма вызывается метод execute, который принимает указатель на объект класса CBoard, в котором хранится топология печатной платы с размещёнными элементами (рис. 3). Результатом работы алгоритма является обработанная топология печатной платы с протрассированными соединениями.

Алгоритм трассировки каждой цепи определяется конкретной реализацией базового класса CRoutingOperator. На текущем этапе разработки приложения используется только волновой алгоритм, реализованный в классе CWaveRoutingOperator.

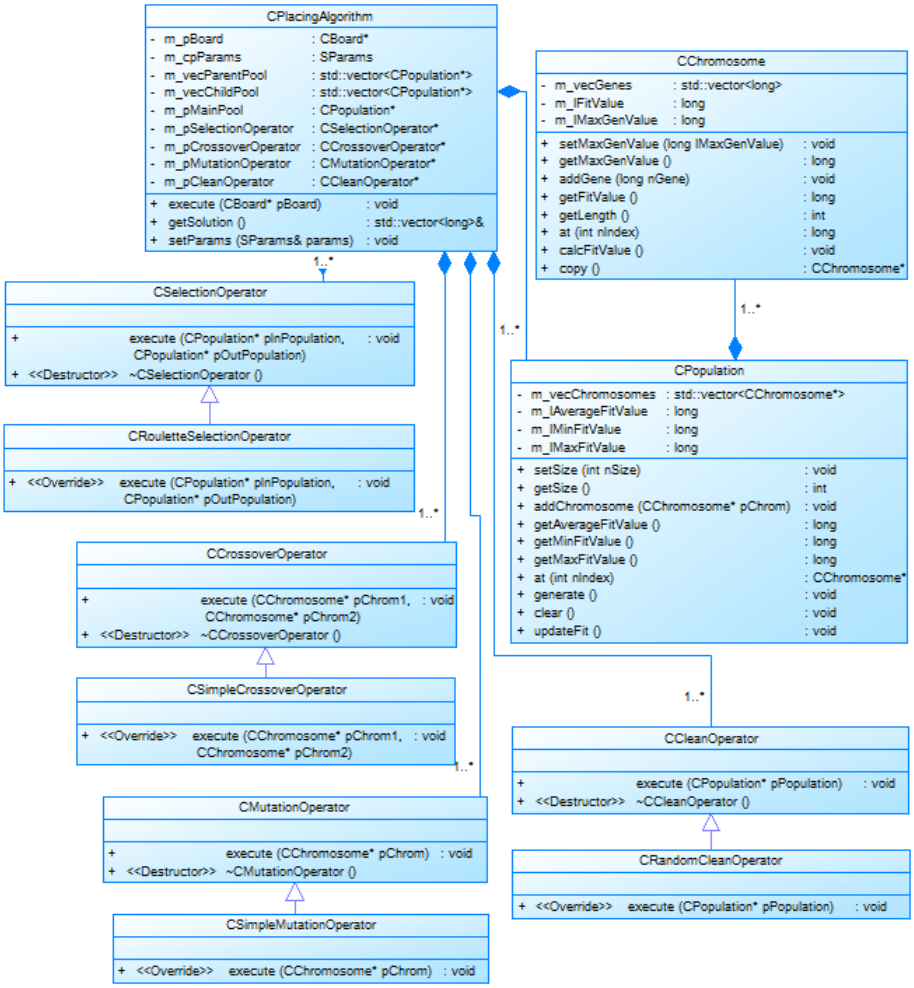

Рис. 2. Диаграмма классов алгоритма размещения

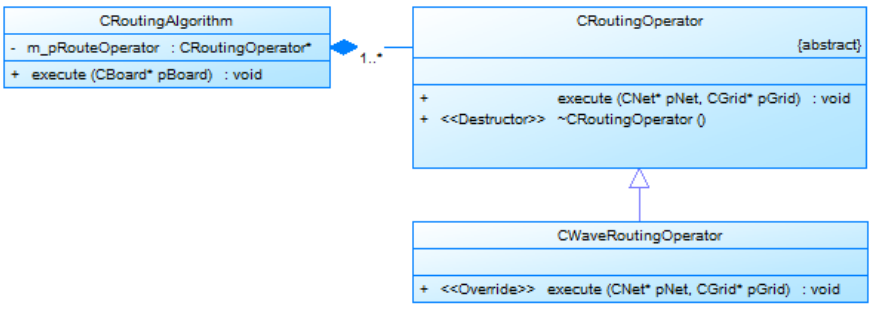

Рис. 3. Диаграмма классов алгоритма трассировки

Описание интерфейса. Для реализации графического интерфейса использовался фреймворк Qt 5.6 [18, 19]. Qt – представляет из себя кроссплатформенный инструментарий разработки прикладного программного обеспечения, широко используемый для создания графических интерфейсов. Он написан на C++ и предоставляет мощные расширения этого языка. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Рассмотрим основные элементы графического интерфейса разработанного приложения. Окно приложения состоит из меню, панели инструментов, рабочего пространства и текстового поля для вывода различной вспомогательной информации. Меню состоит из пунктов File и Help. Пункт меню File содержит подпункты Import и Exit. В пункте Import можно перейти к загрузке файлов содержащих LEF и DEF спецификацию. Пункт Help содержит подпункты About и AboutQt, по нажатию на которые открывается окно с информацией о приложении и окно с информацией о используемой версии библиотеки Qt соответственно.

Панель инструментов содержит кнопку для загрузки конфигурации печатной платы по умолчанию, кнопки для запуска алгоритмов размещения и трассировки, а также кнопку настроек. В окне настроек можно установить параметры размещения элементов на печатной плате такие, как минимальное расстояние между элементами и шаг сетки размещения. А также параметры алгоритма такие, как количество хромосом, количество итераций, вероятности кроссинговера и мутации, и частоту миграции (рис. 4).

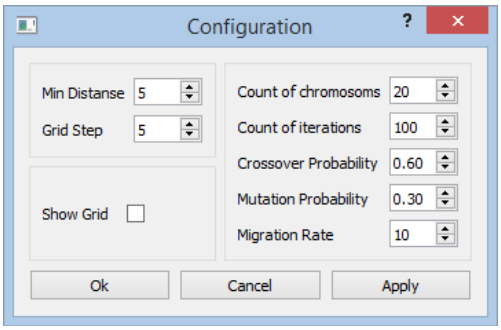

Рис. 4. Окно настроек

В рабочей области приложения осуществляется отрисовка текущего состояния печатной платы. Для отрисовки используется класс QGraphicsView. Для представления графических элементов используются классы, наследуемые от класса QGraphicsItem. Размещаемые элементы печатной платы представляются классом CGraphicComponent, соединения – классом CGraphicNet, сетка размещения – классом CGraphicGrid. Все графические элементы добавляются на сцену. Сцена является объектом класса QGraphicsScene. Сцена отрисовывается с помощью объекта класса QGraphicsView, которым можно манипулировать с помощью матрицы преобразований. Реализована возможность масштабирования и вращения графического отображения печатной платы (рис. 5). При увеличении графического отображения можно изменять видимую область путём перетаскивания. Размещённые графические элементы также можно перетаскивать, тем самым корректируя полученное размещение [18, 19].

Для отображения графиков используется расширение для фреймоворка QT – QCustomPlot. QCustomPlot представляет из себя виджет QT, который используется для построения графиков и визуализации данных. Он не имеет дополнительных зависимостей и хорошо документирован. Данная библиотека позволяет получать качественное визуальное отображение графиков и диаграмм, при этом обладает высокой производительностью, что позволяет использовать её в системах реального времени [20].

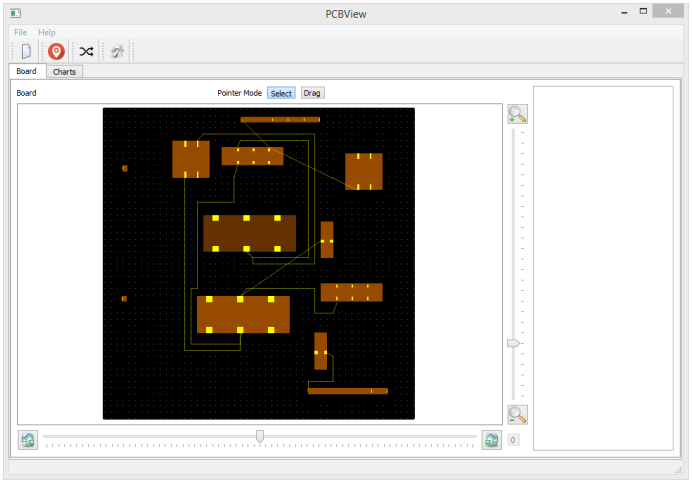

Рис. 5. Окно программы

Для анализа эффективности разрабатываемых алгоритмов используются графики изменения среднего и минимального значения целевой функции размещения (рис. 6). На каждой итерации рассчитывается средние значения целевой функции всех популяций, в которых запущен эволюционный процесс. Также используются графики среднего и минимального значения целевой функции трассировки, значения которой рассчитываются в точках миграции.

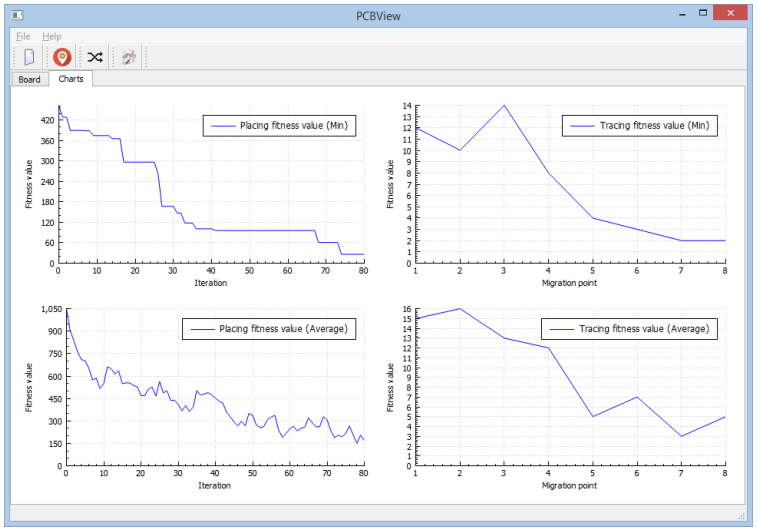

Рис. 6. Отображение графиков

Результаты экспериментов. Были проведены серии вычислительных экспериментов с целью исследования эффективности разработанного алгоритма и программы. В процессе исследования выполнялось решения задачи размещения и трассировки для 300 случайно сгенерированных элементов и 150 цепей, содержащих от 2 до 5 контактов. Ниже представлены результаты экспериментов для различного количества потоков параллельного алгоритма.

Таблица 1

Результаты экспериментальных исследований

Число потоков	% непроведенных соединений					% непроведенных соединений (в среднем)
	1	2	3	4	5	
1	23	21	24	24	22	22.8
2	16	13	14	14	15	14.4
3	13	14	12	13	11	12.6
4	12	10	13	10	11	11.2
5	14	13	13	14	12	13.2

Таблица 2

Сравнение эффективности работы алгоритмов

№	Без использования НЛК			С использованием НЛК		
	(N_{el}=50)	(N_{el}=100)	(N_{el}=150)	(N_{el}=50)	(N_{el}=100)	(N_{el}=150)
1	4585	29658	67953	3147	21296	48509
2	3870	31145	64311	3330	23582	51737
3	4245	28192	68989	2724	23145	50901
4	4056	31632	65576	3425	23481	50798
5	3774	29761	65184	2885	21844	48973
6	4896	28487	67925	2984	23148	49752
7	4129	31845	65427	2873	22946	52164
8	4812	29145	64964	3776	21941	48862
9	3981	29411	65817	3145	22157	50314
10	3876	30491	68482	3168	22981	50957
В среднем	4222,4	29976,7	66862,8	3145,7	22652,1	50296,7

Заключение. Анализ результатов проведенных вычислительных экспериментов позволяет сделать однозначный вывод о том, что использование предложенного гибридного подхода к решению поставленной задачи позволяет добиться существенного улучшения качества получаемых решений. Так, из приведенной таблицы (табл. 2) видно, что выигрыш в качестве получаемых решений при использовании нечеткого логического контроллера составляет порядка 25 % (25,6 % – для задач с 50 элементами; 24,44 % – для 100 элементов; 24,78 % – для 150 элементов) по сравнению с теми задачами, решаемыми без использования нечеткого логического контроллера.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Норенков И.П.* Основы автоматизированного проектирования. – М.: Изд-во МГТУ им. Баумана, 2010.
2. *Shervani, N.* Algorithms for VLSI physical design automation. – Kluwer Academy Publisher, USA, 1995. – 538 p.
3. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы. – М.: Физматлит, 2010.
4. *Курейчик В.М., Курейчик В.В., Родзин С.И.* Концепция эволюционных вычислений, инспирированных природными системами // Известия ЮФУ. Технические науки. – 2009. – № 4 (93). – С. 16-25.

5. *Cohoon J.P., Karro J., Lienig J.* Evolutionary Algorithms for the Physical Design of VLSI Circuits. Advances in Evolutionary Computing: Theory and Applications, Ghosh, A., Tsutsui, S. (eds.) Springer Verlag, London, 2003. – P. 683-712.
6. *Michael A., Takagi H.* Dynamic control of genetic algorithms using fuzzy logic techniques // Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann. – 1993. – P. 76-83.
7. *King R.T.F.A., Radha B., Rughooputh H.C.S.* A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration // Proceedings of 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan. – 2004. – P. 577-582.
8. *Im S.-M., Lee J-J.* Adaptive crossover, mutation and selection using fuzzy system for genetic algorithms // Artificial Life and Robotics. – 2008. – Vol. 13, No. 1. – P. 129-133.
9. *Rodriguez M.A., Escalante D.M., Peregrin A.* Efficient distributed genetic algorithm for rule extraction // Applied Soft Computing. – 2011. – Vol. 11. – P. 733-743.
10. *Alba E., Tomassini M.* Parallelism and evolutionary algorithms // IEEE T. Evolut. Comput. – 2002. – Vol. 6. – P. 443-461.
11. *Zhongyang X., Zhang Y., Zhang L., Niu S.* A parallel classification algorithm based on hybrid genetic algorithm // Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China, 2006. – P. 3237-3240.
12. *Кныш Д.С., Курейчик В.М.* Параллельные генетические алгоритмы: Проблемы, обзор и состояние // Известия РАН. Теория и системы управления. – 2010. – № 4. – С. 72-82.
13. *Гладков Л.А.* Интегрированный алгоритм решения задач размещения и трассировки на основе нечётких генетических методов // Известия ЮФУ. Технические науки. – 2011. – № 7 (120). – С. 22-29.
14. *Гладков Л.А. Гладкова Н.В., Лейба С.Н.* Размещение элементов схем ЭВА на основе гибридных интеллектуальных методов // Известия ЮФУ. Технические науки. – 2015. – № 4 (165). – С. 25-36.
15. *Gladkov L.A., Gladkova N.V., Leiba S.N.* Electronic Computing Equipment Schemes Elements Placement Based on Hybrid Intelligence Approach // Advanced in Intelligent Systems and Computing. Vol. 348: Intelligent Systems in Cybernetics and Automation Theory. – Springer International Publishing, Switzerland, 2015.– P. 35-45.
16. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.* Приёмы объектно-ориентированного программирования. Паттерны проектирования. – М.: Питер, 2010.
17. *Макконел С.* Совершенный код. – М.: Питер, 2005.
18. "Library Exchange Format". University of Maryland, Baltimore County, 2011.
19. Qt Documentation. – http://doc.qt.io/qt-5/reference-overview.html.
20. QCustomPlot. – http://qcustomplot.com/index.php/introduction.

REFERENCES

1. *Norenkov I.P.* Osnovy avtomatizirovannogo proektirovaniya [Fundamentals of CAD]. Moscow: Izd-vo MGTU im. Baumana, 2010.
2. *Shervani, N.* Algorithms for VLSI physical design automation. Kluwer Academy Publisher, USA, 1995, 538 p.
3. *Gladkov L.A., Kureychik V.V., Kureychik V.M.* Geneticheskie algoritmy [Genetic algorithms]. Moscow: Fizmatlit, 2010.
4. *Kureychik V.M., Kureychik V.V., Rodzin S.I.* Kontseptsiya evolyutsionnykh vychisleniy, inspirirovannykh prirodnymi sistemami [Concept evolutionary computation is inspired by natural systems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2009, No. 4 (93), pp. 16-25.
5. *Cohoon J.P., Karro J., Lienig J.* Evolutionary Algorithms for the Physical Design of VLSI Circuits. Advances in Evolutionary Computing: Theory and Applications, Ghosh, A., Tsutsui, S. (eds.) Springer Verlag, London, 2003. – P. 683-712.
6. *Michael A., Takagi H.* Dynamic control of genetic algorithms using fuzzy logic techniques, *Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann,* 1993, pp. 76-83.

7. *King R.T.F.A., Radha B., Rughooputh H.C.S.* A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration, *Proceedings of 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan*, 2004, pp. 577-582.
8. *Im S.-M., Lee J-J.* Adaptive crossover, mutation and selection using fuzzy system for genetic algorithms, *Artificial Life and Robotics*, 2008, Vol. 13, No. 1, pp. 129-133.
9. *Rodriguez M.A., Escalante D.M., Peregrin A.* Efficient distributed genetic algorithm for rule extraction, *Applied Soft Computing*, 2011, Vol. 11, pp. 733-743.
10. *Alba E., Tomassini M.* Parallelism and evolutionary algorithms, *IEEE T. Evolut. Comput*, 2002, Vol. 6, pp. 443-461.
11. *Zhongyang X., Zhang Y., Zhang L., Niu S.* A parallel classification algorithm based on hybrid genetic algorithm, *Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China. 2006*, pp. 3237-3240.
12. *Knysh D.S., Kureychik V.M.* Parallel'nye geneticheskie algoritmy: Problemy, obzor i sostoyanie [Parallel genetic algorithms: Problems, overview, and status], *Izvestiya RAN. Teoriya i sistemy upravleniya* [Journal of Computer and Systems Sciences International], 2010, No. 4, pp. 72-82.
13. *Gladkov L.A.* Integrirovannyy algoritm resheniya zadach razmeshcheniya i trassirovki na osnove nechetkikh geneticheskikh metodov [The integrated algorithm of the decision of problems of placement and routing on the basis of fuzzy genetic methods], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2011, No. 7 (120), pp. 22-29.
14. *Gladkov L.A. Gladkova N.V., Leyba S.N.* Razmeshchenie elementov skhem EVA na osnove gibridnykh intellektual'nykh metodov [Placement circuit elements of electronic computing devices based on hybrid intelligent methods], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2015, No. 4 (165), pp. 25-36.
15. *Gladkov L.A., Gladkova N.V., Leiba S.N.* Electronic Computing Equipment Schemes Elements Placement Based on Hybrid Intelligence Approach, *Advanced in Intelligent Systems and Computing. Vol. 348: Intelligent Systems in Cybernetics and Automation Theory*. Springer International Publishing, Switzerland, 2015, pp. 35-45.
16. *Gamma E., Khelm R., Dzhonson R., Vlissides D.* Priemy ob"ektno-orientirovannogo programmirovaniya. Patterny proektirovaniya [Techniques of object-oriented programming. Design patterns]. Moscow: Piter, 2010.
17. *Makkonel S.* Sovershennyy kod [Perfect code]. Moscow: Piter, 2005.
18. "Library Exchange Format". University of Maryland, Baltimore County, 2011.
19. Qt Documentation. Available at: http://doc.qt.io/qt-5/reference-overview.html.
20. QCustomPlot. Available at: http://qcustomplot.com/index.php/introduction.

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

Гладков Леонид Анатольевич – Южный федеральный университет; e-mail: leo@tgn.sfedu.ru; 347928, г. Таганрог, пер. Некрасовский, 44; тел.: 88634371625; кафедра систем автоматизированного проектирования; доцент.

Гладкова Надежда Викторовна – e-mail: leo_gladkov@mail.ru; тел.: 88634393260; кафедра систем автоматизированного проектирования; старший преподаватель.

Лейба Сергей Николаевич – e-mail: lejba.sergej@mail.ru; тел.: 88634371625; кафедра систем автоматизированного проектирования; аспирант.

Gladkov Leonid Anatol'evich – Southern Federal University; e-mail: leo@tgn.sfedu.ru; 44, Nekrasovskiy, Taganrog, 347928, Russia; phone: +78634371625; the department of computer aided design; associate professor.

Gladkova Nadezhda Viktorovna – e-mail: leo_gladkov@mail.ru; phone: +78634393260; the department of computer aided design; senior teacher.

Leiba Sergey Nikolaevich – e-mail: lejba.sergej@mail.ru; phone: +78634371625; the department of computer aided design; postgraduate student.