МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Кафедра информационных технологий

КУРСОВАЯ РАБОТА

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДВУМЕРНОЙ УПАКОВКИ

Работу выполнил _____ Д.А.Плигин
(подпись, дата)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление 01.03.02 – «Прикладная математика и информатика»

Научный руководитель доц.,
канд. техн. наук, доц. _____ А.А.Полупанов
(подпись, дата)

Нормоконтролер ст. преп. _____А.В.Харченко
(подпись, дата)

Краснодар 2018

РЕФЕРАТ

Курсовая работа 38 с., 3 ч., 8 рис., 10 источников.

ЗАДАЧА РАЗМЕЩЕНИЯ, ГЕНЕТИЧЕСИКИЙ АЛГОРИТМ, ЗАДАЧА ОПТИМИЗАЦИИ, NP-ТРУДНАЯ ЗАДАЧА, ПРИБЛИЖЕННОЕ РЕШЕНИЕ

Объектом исследования являются методы приближенного решения задачи двумерной упаковки прямоугольников на плоскости.

Цель работы – разработка программы, предназначенной для получения такого решения.

В результате проведенной работы была разработана прикладная программа для автоматизации получения приближенного решения задачи двумерной упаковки прямоугольников на плоскости, удовлетворяющая поставленной задаче.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная работа посвящена исследованию и разработке программных средств решения задачи двумерной упаковки прямоугольников с различной шириной и высотой в прямоугольник минимальной площади. В частности, рассматривается алгоритм генетического поиска. Необходимость решения данной задачи возникает в различных сферах деятельности человека. Так при проектировании интегральных схем требуется поместить компоненты разных размеров в прямоугольной области кристалла без взаимных пересечений таким образом, чтобы наименьшая возможная площадь покрывающего их прямоугольника была минимальной [1]. Также при разработке веб-сайтов зачастую возникает необходимость соединять множество изображений в одно ввиду того, что браузер осуществляет обработку множества небольших изображений на веб-странице существенно медленнее, чем одного большого, которое представляет собой эти изображения, соединенные вместе [2].

Актуальность исследования данной темы обуславливается тем, что задача двумерной упаковки прямоугольников на плоскости является NP-трудной, и не представляется возможным получить для нее детерминированный алгоритм, который бы позволил находить точное решение этой задачи за приемлемое время [3]. Этим объясняется повышенный в настоящее время интерес к приближенным методам решения этой и подобных задач, подлежащим реализации на компьютере, которые можно модифицировать для существенного улучшения получаемого результата.

Цель данной работы заключается в разработке прикладной компьютерной программы, предназначенной для получения такого решения.

1 Анализ существующих методов решения задачи двумерной упаковки прямоугольников

1.1 Математическая постановка задачи

Рассмотрим математическую постановку задачи двумерной упаковки прямоугольников на плоскости. Пусть задано некоторое множество P, состоящее из n прямоугольников. Всякий прямоугольник описывается парой вещественных положительных чисел (w_i, h_i), которые представляют собой ширину и высоту прямоугольника соответственно. Требуется найти такой способ размещения прямоугольников множества на двумерной координатной плоскости, чтобы наименьшая возможная площадь покрывающего их прямоугольника была минимальна.

Решение будет представлять собой вектор $\bar{x}$, состоящий из точек расположения прямоугольников p_i для всех i от единицы до n, то есть, если p_i размещен в x_i, то вершина левого нижнего угла прямоугольника p_i находится в точке x_i. Будем помещать все прямоугольники в первой четверти двумерной декартовой системы координат. То есть, ось абсцисс и ось ординат будут прямыми, на которых расположены смежные стороны покрывающего прямоугольника. Зададим ограничения, связанные с условием размещать прямоугольники без взаимного пересечения друг с другом. Для этого введем функцию $g(p_i, x_i, p_j, x_j)$ равную площади пересечения прямоугольника p_i, размещенного в точку x_i, с прямоугольником p_j, размещенным в точку x_j. Тогда ограничение, связанное с условием отсутствия у прямоугольников пересечений друг с другом, будет иметь вид 1.

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} g\left(p_i, x_i, p_j, x_j\right) = 0, \tag{1}$$

Введем обозначение 2.

$$X^1(\bar{x}) = \max_{1 \le i \le n}(x_i^1 + w_i) \quad (2)$$

Также обозначим 3.

$$X^2(\bar{x}) = \max_{1 \le i \le n}(x_i^2 + h_i) \quad (3)$$

Эти два числа определяют соответственно высоту и ширину наименьшего покрывающего прямоугольника. Тогда отношение суммы площадей всех заданных прямоугольников к площади покрывающего прямоугольника будем называть коэффициентом эффективности решения $\bar{x}$. Тогда функция цели будет иметь вид 4.

$$\frac{\sum_{i=1}^{n} w_i \cdot h_i}{X^1(\bar{x}) \cdot X^2(\bar{x})} \to \max \quad (4)$$

Стоит заметить, что задача двумерной упаковки прямоугольников на плоскости принадлежит к классу NP-трудных задач [1]. Из этого следует, что применение на практике детерминированных методов решения данной задачи не представляется возможным, ввиду высоких временных затрат на поиск решения в случае, если число прямоугольников велико.

1.2 Обзор методов решения задачи

1.2.1 Метод, основанный на сортировке прямоугольников по высоте

Данный способ указан в [2]. Пусть дано некоторое конечное множество прямоугольников, а наилучший способ их размещения не задан. Отсортируем

их по убыванию высоты. Будем считать, что высота покрывающего прямоугольника равна высоте первого прямоугольника, а ширина равна бесконечности. Рассмотрим основные шаги этого алгоритма.

а) Попытаемся последовательно в заданном порядке поместить все неразмещенные прямоугольники так, чтобы каждый из них был расположен как можно левее и выше, но при этом не пересекался с уже размещенными прямоугольниками и областью вне покрывающего прямоугольника. Если не удается поместить так какой-либо прямоугольник, переходим к действию е;

б) Если ширина покрывающего прямоугольника равна бесконечности, найдем наименьшую допустимую ширину покрывающего прямоугольника для уже помещенных на плоскость фигур, вычислим его площадь и будем считать ее наилучшим найденным до сих пор результатом, иначе переходим к действию г;

в) Все прямоугольники размещены внутри покрывающего. Вычислим его площадь, и если она меньше до этого найденного минимума, то принимаем ее за минимум и запоминаем текущее расположение прямоугольников;

г) Уменьшим ширину покрывающего прямоугольника на единицу;

д) Удалим все прямоугольники из покрывающего и перейдем к шагу а;

е) Если ширина покрывающего прямоугольника равна наибольшей из ширин размещаемых прямоугольников, то переходим к шагу ж, иначе увеличим высоту покрывающего прямоугольника на единицу и перейдем к шагу д;

ж) Конец.

Данный способ предлагается автором [2] к использованию в разработке интерактивных веб-ресурсов для ускорения обработки веб-страницы предназначенной для этого программой, браузером, поскольку увеличение числа изображений на странице существенно увеличивает время ее обработки, а этот алгоритм позволяет соединить множество изображений воедино и передавать уже как одно целое, сокращая число отдельных изображений на

странице, что может существенно повысить скорость обработки сайта, размещенного в сети интернет.

1.2.2 Метод, основанный на применении генетического алгоритма

Генетический алгоритм – это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе. При использовании генетических алгоритмов решения задачи кодируются подобно признакам живых организмов в генетическом коде. В начале работы алгоритма случайным образом формируется множество решений задачи, называемое популяцией. Затем происходит поэтапное видоизменение решений путем применения к ним операторов кроссинговера и мутации с целью получить более подходящие решения задачи. На каждом этапе происходит отбор (селекция) наилучших с точки зрения значения целевой функции решений, которые становятся новой популяцией, и на их основе далее будет происходить поиск решения. Таким образом, с каждым шагом могут быть найдены лучшие решения рассматриваемой задачи. Идея генетических алгоритмов была впервые предложена американским ученым Джоном Генри Холландом (англ. John Henry Holland) в его работе [4]. Им была предложена схема генетического алгоритма, впоследствии названная канонической [5].

Рассмотрим метод, предложенный в [1]. Разобьем задачу на две подзадачи:

а) Разместить прямоугольники множества P в первой четверти двумерной декартовой системы координат в заданном порядке так, чтобы минимальная возможная площадь покрывающего их прямоугольника была как можно меньше;

б) Найти такой порядок очередности размещения прямоугольников, чтобы при решении задачи, а минимальная площадь покрывающего прямоугольника была как можно меньше.

Рассмотрим алгоритм решения задачи а из [1]. Имеется некоторое упорядоченное множество прямоугольников P. Будем по очереди размещать прямоугольники следующим образом: для каждого очередного прямоугольника формируется конечное множество узлов размещения, состоящее из точки начала координат и вершин углов всех размещенных до этого прямоугольников. Допустим, размещено k прямоугольников, тогда в множество узлов размещения для k+1-го прямоугольника попадут точки (x_i^1, x_i^2), $(x_i^1 + w_i, x_i^2)$, $(x_i^1, x_i^2 + h_i)$, $(x_i^1 + w_i, x_i^2 + h_i)$ для всех i от единицы до n. Затем из этого множества точек выбирается такая, что:

- при размещении k-го прямоугольника в эту точку будет выполнятся условие отсутствия пересечений прямоугольников друг с другом для всех k+1 прямоугольников;

- при размещении прямоугольника p_{k+1} в эту точку минимальная площадь покрывающего прямоугольника будет наименьшей из подобных площадей для всех точек из множества узлов размещения прямоугольника p_{k+1}, удовлетворяющих предыдущему условию. После того, как точка, удовлетворяющая указанным условиям найдена, прямоугольник p_{k+1} размещается в нее.

Указанная последовательность действий повторяется до тех пор, пока в множестве P есть неразмещенные прямоугольники.

Таким образом, с помощью данного алгоритма можно найти некоторое приближенное решение задачи двумерной упаковки прямоугольников на плоскости, но на качество решения в существенной степени влияет порядок, в котором прямоугольники множества P пронумерованы и, соответственно, размещаются на плоскость. Поэтому для поиска наилучшего способа упорядочивания прямоугольников решается задача б.

Представим решение как некоторую перестановку чисел от единицы до n. Найдем перестановку, отражающую наилучший порядок размещения прямоугольников. Задачу б будем решать с помощью генетического алгоритма. Рассмотрим его схему, указанную в [1].

Задаются натуральные числа N, K, L и некоторое условие останова.

а) начало;

б) инициализация начальной популяции из n решений;

в) начало цикла генетического алгоритма;

г) выбор K пар решений для осуществления кроссинговера;

д) кроссинговер с получением K пар потомков;

е) выбор L кандидатов на мутацию;

ж) мутация с получением L новых решений;

и) селекция (отбор) в новое поколение n решений из N+L+2K полученных решений;

к) если условие останова выполнено, то перейти к шагу л, иначе перейти к шагу г;

л) конец.

Автор предлагает использовать одноточечный OX-порядковый кроссинговер, описанный в [9], и одноточечную мутацию, а в качестве схемы скрещивания предлагается равновероятный выбор. В качестве метода отбора решений рекомендуется элитная селекция с отсечением решений с наихудшими значениями целевой функции.

Данный способ решения предлагается автором [1] для применения к решению задач с большим числом прямоугольников, что, очевидно, отличает практическую направленность метода использования генетических алгоритмов от направленности метода, изложенного в [2].

2 Анализ существующих программных средств решения задачи

В настоящее время существуют компьютерные разработки, направленные на решение задачи двумерной упаковки прямоугольников на плоскости. Прежде всего стоит отметить проект SVGnest, в рамках которого создано основанное на генетическом алгоритме программное средство для решения задачи упаковки произвольных фигур на плоскости, которое может быть применено для решения рассматриваемой в данной работе задачи. Поскольку одной из самых распространенных задач, связанных с упаковкой прямоугольников на плоскости, является задача объединения множества изображений в одно (необходимость в этом возникает при разработке интернет-страниц), существуют программные решения, подобные указанным в [2], предназначенные для использования в веб-разработке. Стоит также отметить, что в сети интернет можно найти веб-страницы, предназначенные для визуальной демонстрации принципов и результатов работы некоторых алгоритмов решения задачи двумерной упаковки, что показывает актуальность рассматриваемой проблемы.

Несмотря на универсальность и производительность программы, предложенной проектом SVGnest (которая является наилучшей из найденных) она разработана на языке программирования JavaScript, что подразумевает возможность ее использования только с помощью программы-браузера. Решения, которое бы представляло собой не зависящую от веб-технологий программу, не существует, и это, в свою очередь, обуславливает актуальность разработки такого приложения.

3 Разработка приложения, предназначенного для решения задачи двумерной упаковки прямоугольников на плоскости

3.1 Постановка задачи и выбор средств разработки программы

Необходимо спроектировать и реализовать приложение, позволяющее получать приближенное решение задачи двумерной упаковки прямоугольных объектов для произвольного количества фигур. Программа должна быть такой, чтобы пользователь мог ввести условия задачи, в частности, параметры размещаемых прямоугольников, задать требуемый коэффициент эффективности решения и получить ответ.

Для расширения потенциальных возможностей использования программы следует также дать пользователю возможность ограничить область размещения фигур по x и по y и указать, допустимо ли поворачивать прямоугольники при размещении. Ограниченность области размещения фигур может быть обусловлена условиями конкретной прикладной задачи. Возможность поворота прямоугольников при размещении позволяет расширить множество рассматриваемых способов расположения фигур, что увеличит время работы программы, но, возможно, приведет к более качественному решению. Возможно, что условия решаемой прикладной задачи не позволяют поворачивать прямоугольники, и применение такой стратегии поиска ответа может привести к получению решения, не адекватного ограничениям реальной проблемы, поэтому следует предоставить пользователю возможность выбора между этими двумя стратегиями поиска наилучшего способа размещения. После введения условий задачи, указания значений всех необходимых параметров и ограничений на время работы, программа должна начать поиск решения, и после того, как будет найдено решение со значением коэффициента эффективности не меньше заданного, предоставить пользователю результат работы, либо если по окончании заданного времени работы программы

решение с необходимым коэффициентом качества не будет найдено, то пользователю будет выдан наилучший достигнутый результат.

Для удобства работы с программой она должна быть выполнена в виде приложения с оконным интерфейсом. Ввиду, возможно, большого количества входных данных и, соответственно, большого размера решения, самым оптимальным способом ввода-вывода данных условий задачи и решения будет использование текстовых файлов, имена которых задает пользователь. Также, ввиду геометрического характера объектов, рассматриваемых в задаче, необходима визуализация полученного результата в виде изображаемых на экране прямоугольников, которые расположены относительно начала координат в соответствии с полученным способом их размещения.

Для разработки заданной программы выбрана интегрированная среда разработки Microsoft Visual Studio 2013. В качестве используемого языка программирования выбран C++. Выбор обоснован тем, что указанная среда разработки предоставляет большие возможности для написания и отладки приложений для операционной системы Windows, которая, в свою очередь, является самой распространенной операционной системой для персональных компьютеров. Язык программирования C++ является компилируемым языком и предоставляет возможность создания высокопроизводительных программ, что актуально для решения математических задач программными средствами. В качестве источников информации о средствах разработки используются [6-9].

3.2 Алгоритм последовательного размещения прямоугольников

Рассмотрим алгоритм последовательного размещения фигур. Пусть дано некоторое упорядоченное множество прямоугольников, которые необходимо разместить в заданном порядке на плоскости таким образом, чтобы минимальная площадь покрывающего их прямоугольника была как можно меньше. Кроме того, должны соблюдаться заданные пользователем условия ограничения размера покрывающего прямоугольника и возможности или

невозможности поворачивать фигуры при размещении. Также необходимо вычислить коэффициент эффективности полученного решения.

Поместим первый прямоугольник в начало координат. Затем для каждого очередного прямоугольника сформируем множество узлов размещения, состоящее из вершин углов всех размещенных до этого фигур. Переберем все узлы размещения для рассматриваемого прямоугольника и найдем такой, что если разместить в него рассматриваемую фигуру, то данное размещение не будет противоречить условиям отсутствия у прямоугольников пересечений друг с другом (сумма площадей всех возможных пересечений должна быть равна 0), невозможности пересечения любой фигуры с областью вне первой четверти декартовой системы координат, а также заданным пользователем ограничениям на область размещения, и, кроме того, наименьшая возможная площадь покрывающего прямоугольника будет минимальной среди аналогичных для всех других узлов размещения. Этот принцип размещения проиллюстрирован на рисунке 1. После нахождения точки, удовлетворяющей указанным условиям, рассматриваемый прямоугольник размещается туда.

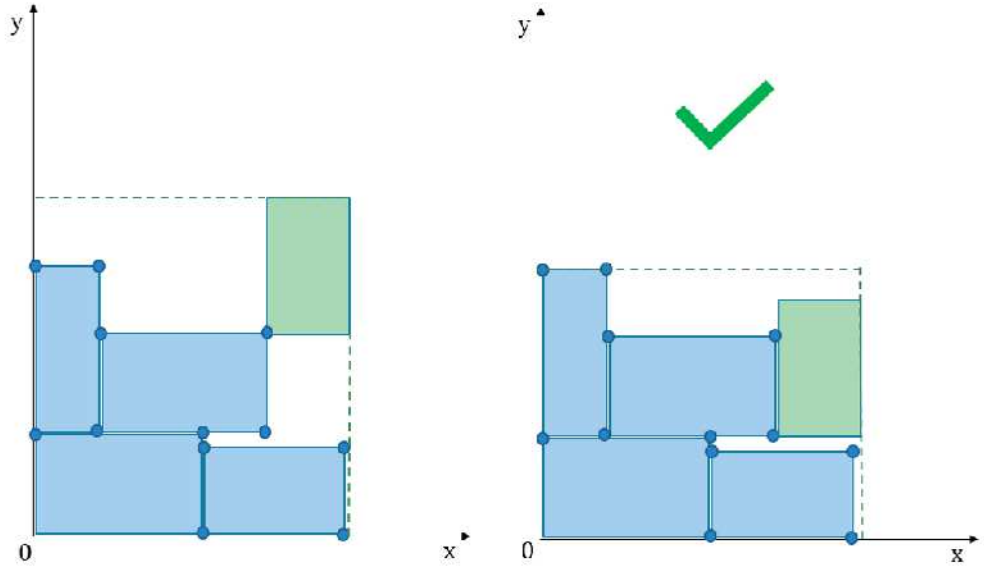


Рисунок 1 – Перебор узлов размещения

В случае, если пользователь указал, что возможен поворот фигур, то при поиске узла размещения текущего прямоугольника рассматривается вдвое больше вариантов: для неизмененной фигуры и для повернутой на 90 градусов. Если прямоугольник с точки зрения уменьшения минимизируемой площади лучше повернуть, то он должен быть размещен именно таким образом.

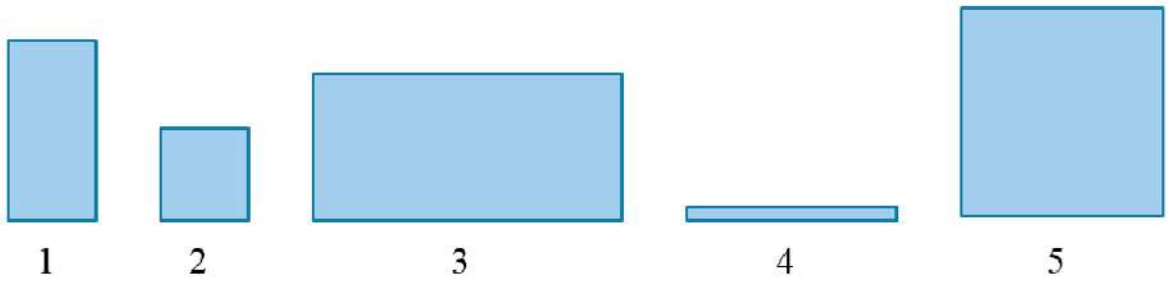
В том случае, если ни одна рассматриваемая точка не удовлетворяет указанным ограничениям, то весь данный в начале работы алгоритма порядок размещения прямоугольников помечается как некорректный, коэффициент эффективности для такого способа считается равным 0. Это делается для того, чтобы в ходе исполнения генетического алгоритма такие неадекватные решения были отсеяны. После успешного размещения всех данных объектов вычисляется наименьшая площадь покрывающего их прямоугольника и ее значение становится коэффициентом эффективности полученного решения.

3.3 Генетический алгоритм нахождения наилучшего порядка размещения прямоугольников

3.3.1 Начальный этап работы алгоритма

Рассмотрим генетический алгоритм поиска способа упорядочивания фигур для последовательного размещения, наилучшего с точки зрения минимизации площади наименьшего покрывающего прямоугольника. Решения задачи здесь будут кодироваться перестановками чисел от единицы до n, где n – число фигур, которые необходимо разместить на плоскости оптимальным образом. Этот принцип кодирования решений задачи поиска наилучшего порядка размещения проиллюстрирован на рисунке 2. Дан набор из пяти прямоугольников разного размера. Пронумеруем их числами от одного до пяти. Любой из 120 способов их упорядочения можно представить в виде последовательности номеров прямоугольников. Порядок вхождения номеров прямоугольников в последовательность соответствует порядку самих

прямоугольников. Закодированное так решение можно представлять в памяти компьютера в массиве целых чисел.



1 | 2 | 3 | 4 | 5	2 | 1 | 3 | 4 | 5	1 | 2 | 4 | 5 | 3
2 | 4 | 3 | 1 | 5	1 | 2 | 4 | 3 | 5	2 | 4 | 3 | 5 | 1
5 | 4 | 3 | 1 | 2	5 | 4 | 1 | 3 | 2	4 | 5 | 2 | 1 | 3

Рисунок 2 – Кодирование решений

Фитнес-функцией (целевой функцией, функцией приспособленности) будет единственный для каждой перестановки чисел коэффициент эффективности решения, полученный при размещении фигур на плоскость изложенным выше алгоритмом в порядке, описываемом этой перестановкой.

Заданы некоторые натуральные числа N (размер популяции, число решений, переходящих каждый раз в следующее поколение), K (число пар, выбираемых для проведения кроссинговера), L (число решений, к которым будет применяться оператор мутации). Сначала происходит заполнение начальной популяции решений случайными перестановками. И по алгоритму последовательного размещения фигур вычисляется их значения функции приспособленности. Затем запускается цикл, выполняемый до тех пор, пока наибольшее (то есть, наилучшее) найденное значение фитнес-функции меньше

заданного пользователем числа или время, отведенное на работу программы, не закончилось.

Рассмотрим состав одной итерации цикла. В начале итерации происходит случайный выбор из текущей популяции K пар решений для проведения кроссинговера, затем происходит собственно скрещивание. Поскольку решения здесь представляют собой не произвольные последовательности чисел, а ограниченные тем условием, что в одной последовательности-решении каждое число должно встречаться не менее и не более одного раза, то применение обычных методов реализации кроссинговера здесь невозможно, ввиду того, что с достаточно высокой вероятностью после применения оператора кроссинговера будут получены совершенно неадекватные поставленной задаче решения: в них какие-то прямоугольники будут размещены на плоскость более одного раза, а другие не попадут в рассмотрение алгоритмом последовательного размещения вообще. Поэтому разумным решением здесь будет применять описанный в [10] оператор кроссинговера для порядкового представления решений (OX кроссовер).

Будет реализовано несколько алгоритмов выбора особей для скрещивания, выбора особей для мутации, собственно мутации и селекции решений, которые пользователь может выбирать и по-разному организовывать вычисления. Так, перед началом работы программы пользователь наряду с заданием условий задачи, ограничений на время поиска и числовых параметров вычислений должен выбрать для использования алгоритмы, которые будут применены в цикле генетического поиска.

3.3.2 Оператор кроссовера

Рассмотрим предоставленные в приложении алгоритмы подбора пар перестановок для получения новых решений путем их скрещивания: панмиксию, инбридинг, аутбридинг и ассоциативное скрещивание. Результатом выполнения любого из этих алгоритмов является массив целых чисел,

именуемый в программном коде couples. Всякий элемент массива соответствует какому-либо решению в текущей популяции, и в котором хранится номер решения, с которым оно образует пару для скрещивания. Рассмотрим простейший способ получения таких пар: панмиксию. Панмиксия – биологическое понятие, обозначающее теоретическое представление, согласно которому все варианты опыления и скрещивания в популяции реализуются с равной вероятностью. Тот же принцип может быть реализован и при подборе пар решений в генетическом алгоритме. Принцип заполнения массива couples в этом случае крайне прост: сначала всякий его элемент содержит свой номер (на этом этапе все особи составляют пару сами с собой), затем происходит череда случайных обменов содержимым элементов массива couples, после чего полученная структура данных будет задавать пары решений для скрещивания, которые составлены по принципу случайного подбора. Данный способ описан в [5].

Также для составления пар особей для проведения кроссинговера может использоваться принцип инбридинга. В биологии инбридинг – это скрещивание близкородственных форм в пределах одной популяции организмов. В контексте теории генетических алгоритмов это осуществляется по следующему принципу: пары подбираются такими, чтобы перестановки, их составляющие, мало отличались друг от друга. Здесь в качестве способа измерения степени различия решений в популяции используется расстояние Хэмминга, то есть, число позиций, в которых соответствующие элементы двух последовательностей чисел одинаковой длины различны. Таким образом, будет производиться скрещивание перестановок, которым соответствуют мало отличающиеся способы упорядочения размещаемых фигур, то есть расстояние Хэмминга между этими особями будет мало. Каждому решению алгоритм будет стремиться поставить в соответствие другое решение, Хэммингово расстояние до которого будет минимально. Крайне важно в этом случае снизить вероятность массового образования пар из одинаковых решений, так как расстояние Хэмминга между совпадающими последовательностями равно

нулю, что, очевидно, является минимумом всех таких возможных расстояний. Этот способ составления пар для проведения кроссинговера описан в [5].

Рассмотрим другой принцип подбора пар особей для скрещивания, инбридинг. Аутбридинг – это понятие, противоположное инбридингу, оно означает один из методов разведения, представляющий собой неродственное скрещивание. В соответствии с этим принципом каждой перестановке будет подобрана в пару другая такая перестановка, которая будет сильно отличаться от нее. В качестве меры сходства здесь снова разумно использовать расстояние Хэмминга. Каждому решению в популяции будет подобрано другое решение, Хэммингово расстояние до которого будет максимальным среди прочих. Данный способ также представлен в [5].

Также в приложении пользователь может выбрать для использования в поиске решения принцип ассоциативного скрещивания. Этот принцип составления пар для проведения кроссинговера похож на принцип инбридинга с той лишь разницей, что в качестве меры степени различия решений здесь используется модуль разности значения фитнес-функции от этих решений. То есть, особи могут мало отличаться друг от друга как перестановки чисел, но притом иметь сильно различающиеся значения функции приспособленности, и таким образом эти два решения вряд ли образуют пару в соответствии с принципом ассоциативного скрещивания. В то же время, у двух сильно различающихся способов упорядочивания размещаемых фигур могут быть одинаково высокие значения фитнес-функции, в этом случае они могут образовать пару для скрещивания. Здесь так же, как и в инбридинге, важно соблюсти правило маловероятности образования пар из одинаковых особей, поскольку алгоритм будет так же стремиться поставить каждому решению в соответствие решение, мало отличающееся в смысле величины модуля разности значений фитнес-функции от этих перестановок, а для одинаковых решений такая разность, несомненно, равна нулю, что является абсолютным минимумом модуля вещественного числа.

Перейдем к алгоритму, по которому в приложении будет проходить кроссинговер особей в генетическом поиске решения. Рассмотрим принцип реализации представленного в программе алгоритма скрещивания, алгоритма одноточечного кроссовера OX. Пусть выбраны два решения-родителя. Одно из них назовем разрезаемой строкой, а другое заполняемой строкой. Случайным образом выберем натуральное число T – точку кроссовера. Затем сформируем решение-потомка: первые T его элементов будут определяться секцией кроссовера, то есть, первыми T элементами разрезаемой строки, а все прочие элементы разрезаемой строки войдут в блок заполнения. Затем строка-потомок дополняется до требуемой длины числами из блока заполнения, причем эти числа записываются в новое решение в том порядке, в котором они встречаются в заполняемой строке. Этот принцип показан на рисунке 3.

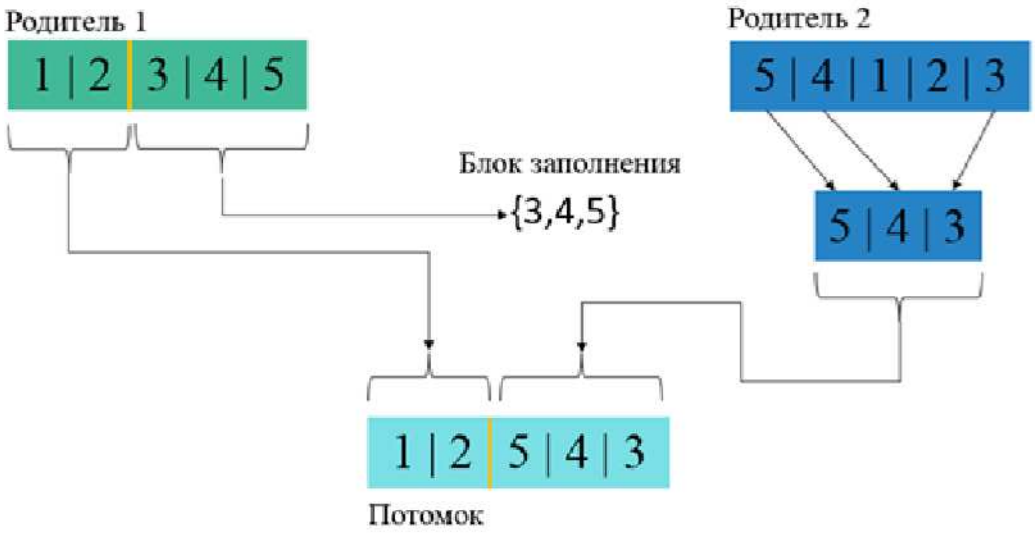


Рисунок 3 – Порядковый OX-кроссовер

После этого аналогичным образом формируется второе решение-потомок, при этом разрезаемая строка и заполняемая строка меняются этими ролями. Таким образом, с помощью OX-кроссовера возможно получить гарантированно допустимые решения (то есть, не произвольные последовательности, а именно перестановки), которые будут иметь структурные сходства с каждым из решений-родителей, но притом будут отличаться от каждого из них. Для

полученных в процессе кроссинговера новых решений будут также вычислены их значения функции приспособленности, и затем сами решения будут добавлены в популяцию. Кроссовер OX изложен в [10].

Следующим шагом итерации цикла рассматриваемого генетического алгоритма будет выбор из полученной на этапе скрещивания популяции L решений, которые будут подвержены мутации.

3.3.3 Оператор мутации

Вначале рассмотрим принципы выбора особей для получения новых решений на основе текущей популяции путем случайных изменений рассматриваемых решений. В приложении реализовано три способа отбора особей для мутации: случайный подбор, выбор из лучших, выбор из худших. Метод случайного подбора заключается в случайном выборе номеров особей, из которых в процессе мутации будут получены новые решения. Особей выбирается не больше, чем L. Метод выбора из лучших заключается в выборе для осуществления мутации L лучших по качеству решения особей. Метод выбора из худших, соответственно, заключается в выборе для мутации L худших по качеству решения особей.

Рассмотрим принцип реализации оператора точечной мутации для перестановочного кодирования решений. Пусть выбрано некоторое решение. Выберем натуральное число T – точку мутации. Сформируем новое решение-мутанта таким, чтобы оно полностью совпадало с выбранным для мутации решением за исключением элементов с номерами T и T+1. Их значения меняются местами. Принцип работы этого алгоритма показан на рисунке 4. Для полученного таким образом решения по изложенному выше алгоритму последовательного размещения вычисляется его значение функции приспособленности, а само решение добавляется в текущую популяцию. Таким образом, посредством применения оператора точечной мутации для перестановочного кодирования решений возможно получать новые решения,

основанные на полученных ранее, но отличающихся от них. Отличие зависит от случайного события, и это дает возможность в случае сходимости алгоритма к локальному оптимуму выйти из его окрестности и попасть в окрестность другого, более качественного решения.

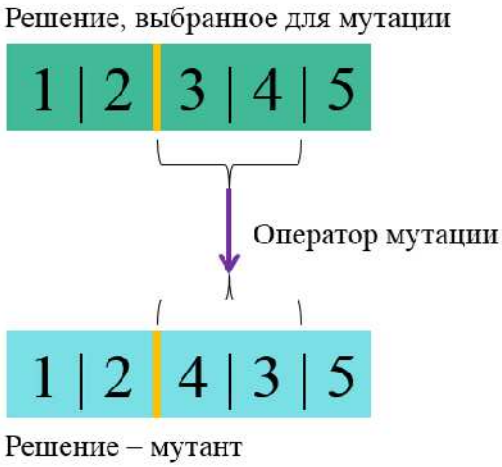


Рисунок 4 – Точечная мутация

Рассмотрим другой метод получения новых решений посредством мутации, метод инверсии. В методе инверсии, описанном в [5] выбирается два различных целых числа k_1, k_2, которые не меньше единицы и не больше количества прямоугольников (то есть, длины перестановки-решения), причем k_1 меньше, чем k_2. Затем последовательность чисел от k_1 до k_2 в рассматриваемом решении переписывается в обратном порядке. То есть, число, стоявшее в решении под номером k_1, меняется местами с числом на позиции k_2, число под номером k_1+1 меняется со значением номер k_2-1 и так далее. Данный способ мутации позволяет получить новое решение, сильно отличающееся от исходного.

Рассмотрим алгоритм мутации, основанный на множественных случайных перестановках чисел в решении, сальтацию. Пусть выбрано решение для проведения мутации. Выберем фиксированное количество случайных целых чисел от единицы до количества прямоугольников, данного в задаче. Эти числа должны образовывать две упорядоченных последовательности

порядковых номеров чисел в перестановке, кодирующей решение задачи. Назовем их k_1... k_n и t_1…t_n. Затем меняются местами числа с номерами k_1 и t_1, затем k_2 и t_2 и так далее. В итоге будет получена новая перестановка чисел, задающая способ упорядочивания данных в задаче прямоугольников. Все эти способы подбора особей для получения новых решений посредством мутации изложены в [10].

3.3.4 Оператор селекции

После проведения скрещивания и мутации решений, полученная популяция из N+2K+L решений сортируется по убыванию значения фитнес-функции, и производится отбор решений в новую популяцию, на основе которой будут происходить все действия в следующей итерации генетического алгоритма. Отбор может осуществляется посредством использования гибкого сочетания элитарного селекции и случайного выбора решений. Способ называется гибким, поскольку соотношение числа особей, отбираемых случайно, и число особей, которые проходят элитарный отбор можно изменять перед началом поиска решения.

Рассмотрим данный алгоритм селекции. Пусть выбрано натуральное число E – число наилучших решений в текущей популяции, гарантированно проходящих отбор в новое поколение. Поместим в новое поколение первые E наилучших решений из текущей популяции. Затем, если число решений в новой популяции меньше, чем N, поместим туда случайно выбранное решение из текущей популяции. Будем повторять предыдущий шаг до тех пор, пока число решений в новой популяции не станет равно N. Получена новая популяция, которая в следующей итерации цикла генетического алгоритма будет рассмотрена как текущая. Данный принцип отбора проиллюстрирован на рисунке 5. Поскольку до начала селекции текущая популяция состояла из популяции, полученной во время предыдущей итерации цикла, и решений, полученных посредством применения операторов кроссинговера и мутации к

старым решениям, результат работы алгоритма со временем не будет ухудшатся. В случае, если среди новых решений окажется перестановка, дающая большее значение коэффициента эффективности решения, то она попадет в новое поколение вместе с другими последовательностями, прошедшими элитарный отбор.

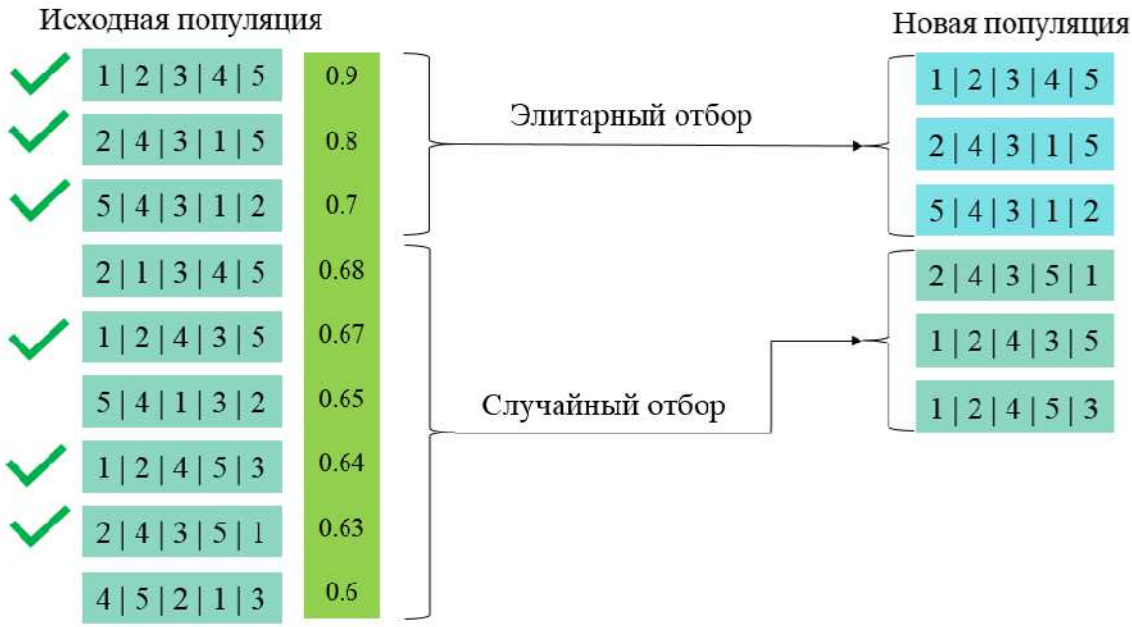


Рисунок 5 – Элитарный отбор

Если же новые решения не будут превосходить старые по значению коэффициента эффективности, то они не займут место лучшего найденного на данный момент времени решения в новом поколении, ввиду того же принципа элитарной селекции. В то же время следующий за этим случайный отбор препятствует преждевременной сходимости алгоритма и позволяет не самым лучшим решениям попасть в новое поколение, чтобы они, в последствии, могли принять участие в скрещивании и мутации и, возможно, стали причиной появления более качественных решений в будущем.

Процедура отбора решений в новое поколение, которое будет рассматриваться в следующей итерации цикла, по выбору пользователя может быть осуществлена по описанному в [5] принципу турнирной селекции.

Рассмотрим алгоритм турнирной селекции. Пусть есть некоторое множество решений задачи упорядочения прямоугольников для оптимального размещения на плоскости, полученное как сумма множества решений, отобранных на предыдущей итерации, и множеств новых решений, полученных в процессе проведения кроссинговера и мутации. Случайным образом выберем из этого множества два решения, сравним их значения функции приспособленности и поместим в новое поколение то, которое будет иметь большее значение. Таким образом, между случайно выбранными решениями происходит так называемый парный турнир, а решение с большим значением функции приспособленности называется победителем этого турнира и считается прошедшим отбор. Повторяя указанную процедуру еще N-1 раз получим новую популяцию решений.

3.3.5 Оценка качества полученного решения

После получения новой популяции необходимо провести оценку качества результата, полученного на данном этапе работы генетического алгоритма, найдя наилучшее значение коэффициента эффективности решения среди всех таковых в популяции. После нахождения такого числа, оно сохраняется, а также сохраняется соответствующее ему решение. В том случае, если найденное лучшее решение является некорректным (то есть, найденное наибольшее значение коэффициента эффективности решения равно 0), то считается, что решение поставленной задачи найти не удается. Последнее может произойти, например в случае, если пользователь поставил такие ограничения на область размещения фигур, что данные прямоугольники в указанную область разместить нельзя никаким образом.

Затем, если условие останова не выполнено (найденное наивысшее значение коэффициента эффективности решения пока еще меньше заданного пользователем или не истекло время, отведенное пользователем на поиск решения), происходит следующая итерация цикла генетического алгоритма,

иначе цикл прерывается, а найденное решение считается наилучшим, и на его основе с помощью алгоритма последовательного размещения прямоугольников находится вектор $\bar{x}$ – искомое решение задачи двумерной упаковки прямоугольных объектов на плоскости.

Для расширения возможностей использования приложения следует предлагать пользователю явно задавать фигурировавшие в описании алгоритма числовые параметры N, K, L, E и принципы проведения операции кроссинговера, мутации и селекции, тем самым давая возможность для проведения экспериментов и нахождения значений параметров и составляющих алгоритма, способствующих сходимости к наилучшему решению, а также наиболее подходящих для решения прикладных задач, связанных с нахождением такого способа размещения некоторого множества прямоугольников на плоскости, чтобы наименьшая возможная площадь покрывающего из прямоугольника была минимальной.

3.4 Архитектура приложения

3.4.1 Общий принцип работы приложения

Приложение может одновременно находиться в одном из четырех состояний, которое определяется значением, записанным в целочисленной глобальной переменной appMode. Рассмотрим состояния приложения.

а) appMode равно 0 - состояние ожидания ввода;

б) appMode равно единице - состояние считывания данных из файла;

в) appMode равно двум - состояние поиска решения;

г) appMode равно трем - состояние демонстрации и вывода найденного решения в файл.

В момент запуска приложение находится в состоянии а. Успешное задание пользователем имен файлов ввода и вывода, а также параметров задачи переводит приложение в состояние б. В случае, если указанный пользователем

файл ввода существует и доступен для чтения, происходит считывание данных о прямоугольниках, после чего приложение переходит в состояние в и начинается процесс поиска решения. Если решение, удовлетворяющее заданным условиям, найдено или закончилось время, отведенное на поиск решения, то приложение переходит в состояние г, создается или открывается файл вывода, в который затем осуществляется вывод решения, а на экране появляется визуализация полученного способа размещения прямоугольников на плоскости.

3.4.2 Описание основных структур данных

Для решения указанной задачи определяются структуры данных

- point;
- rectangle;
- solution.

Структура point предназначена для хранения координат одной точки двумерной декартовой системы координат, для чего она имеет два поля: x и y типа double. Все координаты точек, а также параметры прямоугольников в процессе вычислений в программе будут храниться в переменных и полях структур типа double, что обусловлено постановкой математической задачи. Структура rectangle предназначена для хранения параметров прямоугольника: его высоты, ширины и точки размещения. Для этого определены поля w и h типа double и поле p определенного выше типа point. Структура solution предназначена для хранения данных об одном конкретном решении задачи двумерной упаковки прямоугольников на плоскости. В ней определены поле fit типа double, предназначенное для хранения значения коэффициента эффективности решения, и поле sol, представляющее собой динамический массив целочисленных элементов, хранящий в себе решение, закодированное в соответствии с принципом работы используемого генетического алгоритма, то есть, упорядоченную последовательность чисел от одного до n, где каждое

число встречается ровно один раз. Динамические массивы реализуются в программе посредством использования шаблона vector стандартной библиотеки шаблонов языка программирования C++.

3.4.3 Описание основных переменных

Ключевыми переменными, используемыми в программе, являются

- rects;
- gen.

Динамический массив rects содержит структуры типа rectangle и предназначен для хранения данных о всех размещаемых прямоугольниках и используется в реализации алгоритма последовательного размещения в заданном порядке. Динамический массив gen состоит из структур типа solution и предназначен для хранения популяции решений, используется в реализации генетического алгоритма поиска порядка размещения прямоугольников на плоскости, наилучшего с точки зрения минимизации площади наименьшего возможного покрывающего прямоугольника.

3.4.4 Описание основных функций

Ключевыми функциями, реализующими основные решающие задачу алгоритмы, являются

- putRects;
- evolve.

Функция putRects реализует алгоритм последовательного размещения прямоугольников на плоскости. Она принимает в качестве единственного аргумента динамический массив, содержащий решение, закодированное в форме, соответствующей принципу работы используемого генетического алгоритма, и возвращает значение коэффициента эффективности этого решения. Кроме того, поскольку, вычисляя этот коэффициент, функция

изменяет значения полей структур rectangle, хранящихся в массиве rects (последовательно размещает прямоугольники наилучшим образом), после окончания ее выполнения из массива rects можно будет извлечь данные об оптимальном размещении фигур, соответствующем обработанному функцией закодированному решению.

Функция evolve не принимает и не возвращает никаких значений, она содержит в себе реализацию генетического алгоритма поиска наилучшего порядка размещения фигур на плоскости функцией putRects. Основная структура данных, с которой работает evolve – массив gen, которые содержит данные о текущей популяции решений.

Функции evolve и putRects взаимодействуют в соответствии с описанным ранее выбранным алгоритмом поиска решения задачи двумерной упаковки прямоугольников на плоскости.

3.4.5 Правила ввода-вывода

Вспомогательными функциями, реализующими процедуры файлового ввода-вывода, являются readRects и writeRects. Функция readRects считывает из указанного пользователем файла данные о размещаемых прямоугольниках в определенном формате. Данные о прямоугольниках должны быть записаны в файле построчно, на каждой строке расположены параметры одного прямоугольника. Каждая строка должна представлять собой запись двух вещественных чисел, разделенных пробелом – ширины и высоты прямоугольника соответственно. Прочие записи в файле приведут к неверному считыванию приложением данных решаемой задачи.

Вызов функции writeRects происходит тогда, когда искомое приближенное решение найдено, и данные о соответствующем способе размещения прямоугольников на плоскости хранятся в массиве rects. Эта функция создает файл с указанным пользователем именем и записывает туда полученное решение задачи в определенном формате, а также коэффициент

качества этого решения и время, затраченное на его поиск. В записи решения так же, как и в записи условий задачи, данные о прямоугольниках расположены построчно, на каждой строке находятся данные о размещении одного прямоугольника. Каждая строка записанного решения является записью четырех действительных чисел, разделенных пробелами – координаты x точки размещения прямоугольника, координаты y точки размещения прямоугольника, его ширины и высоты соответственно.

3.4.6 Обработка ошибок

Приложение спроектировано так, что оно может различать возникающие случаи некорректной работы и сообщать пользователю о произошедшей ошибке. Для этого в программе существует глобальная целочисленная переменная errCode, которой в случае ошибки, произошедшей при выполнений какого-либо действия, присваивается код ошибки. Кроме того, при возникновении такой ситуации приложение переводится в начальное состояние a. Сообщения об ошибках выводятся на экран только, если программа находится в состоянии a.

Рассмотрим возможные значения переменной errCode.

а) errCode равно 0 - начальное значение, ошибок нет, приложение работает нормально, никакие сообщения не выводятся;

б) errCode равно единице - указанный пользователем файл для ввода не найден;

в) errCode равно двум - заданы некорректные ограничения на область размещения фигур: хотя бы одно из ограничений на область размещения фигур, по x или по y, принимает значение меньшее либо равное 0;

г) errCode равно трем - в файле ввода найдены некорректные данные: прочитанная ширина или высота какого-либо прямоугольника, меньше либо равна 0;

д) errCode равно четырем - задано некорректное значение желаемого коэффициента эффективности решения: коэффициент не принадлежит интервалу от 0 до единицы;

е) errCode равно пяти - задан некорректный размер популяции, меньший либо равный 0;

ж) errCode равно шести - задано некорректное число мутаций, меньшее либо равное 0;

и) errCode равно семи - не удается найти решение задачи;

к) errCode равно восьми - ошибка ввода параметров задачи. В поле для ввода одного из условий задачи или одного из параметров генетического алгоритма введена строка, которую невозможно интерпретировать как значение нужного числового типа;

л) errCode равно девяти - задано некорректное число скрещиваний, не большее 0 или большее размера популяции;

м) errCode равно 10 - задан некорректный параметр элитарного отбора, не больший 0 или больший размера популяции.

3.4.7 Режим тестирования

Поскольку для выполнения вычислений, необходимых для поиска решения, могут быть использованы разные алгоритмы и заданы различные значения параметров, то разумным решением будет реализовать в приложении возможность его запуска в режиме тестирования. Работа в режиме тестирования будет представлять собой некоторое число последовательных запусков поиска решения с выбранными числовыми параметрами и алгоритмами скрещивания, мутации и селекции, при которых полученные результаты (время работы и наилучший найденный коэффициент качества решения) сохраняются, а после того происходит вывод среднего времени, затраченного на поиск, среднего коэффициента качества полученных при этом решений, а также графика функции от числа проведенных итераций, которая

равна лучшему найденному значению целевой функции за это число итераций. В окне ввода параметров поиска решения, предусмотрена возможность указать приложению необходимость запуска вычислений в тестовом режиме и задать число последовательных запусков для проведения тестирования. Вывод средних значений времени работы алгоритма и целевой функции для заданного способа поиска решения задачи осуществляется как в указанный пользователем файл, так и на экран по окончании тестирования. График функции от числа проведенных итераций, которая равна лучшему найденному значению целевой функции за это число итераций выводится только на экран. Вид этого графика дан на рисунке 6.

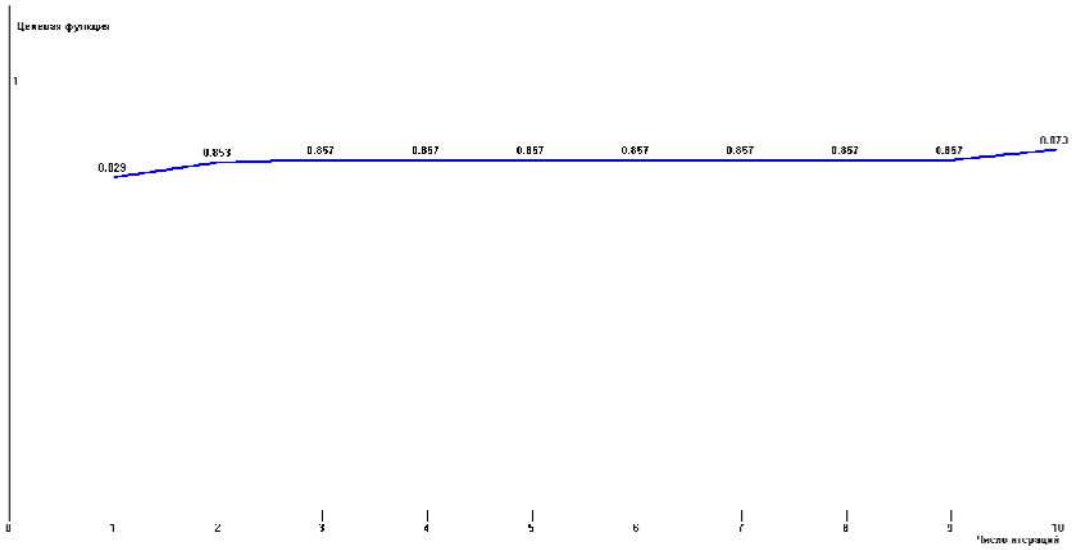


Рисунок 6 – Результат тестирования алгоритма

С помощью тестирования возможно выявить более эффективные алгоритмы. Например, в ходе проведенных пятидесяти тестовых запусков было выяснено, что алгоритм элитарной селекции дает результат лучше, чем алгоритм турнирной селекции, поскольку при приблизительно одинаковых полученных средних коэффициентах качества решения среднее время работы алгоритма с отбором по первому способу существенно меньше, чем среднее время работы алгоритма с турнирным отбором.

По аналогичному принципу было выяснено, что выбор особей для мутации из лучших решений дает более качественный результат, чем случайный выбор: средний коэффициент качества решения, полученный с его помощью за пятьдесят запусков алгоритма, не ниже, а среднее время работы программы существенно меньше, чем при том же числе запусков алгоритма со случайным выбором.

3.5 Описание интерфейса приложения

Программа выполнена в виде приложения с оконным интерфейсом. Он реализован с помощью стандартных средств интерфейса прикладного программирования Win32 API. Здесь будем рассматривать случай корректной работы программы.

Главное окно программы имеет меню, в котором есть единственный пункт “Открыть” (подразумевается открытие файла для считывания данных задачи). В этом пункте меню есть единственный пункт “Старт”, при выборе которого открывается диалоговое окно для ввода данных задачи, выбора алгоритмов для генетического поиска решения и задания его числовых параметров. Вид этого окна представлен на рисунке 7. В этом окне пользователь задет имена файлов для ввода и вывода, вводит числовые параметры задачи, выбирает алгоритмы для проведения генетического поиска, а также определяет время работы, по истечении которого программа прекратит вычисления, даже если не найдет решение требуемого качества. После этого по нажатию на кнопку “Старт!” приложение начнет поиск решения. В окне “Старт” возможно также задать запуск вычислений в режиме тестирования и указать нужно число запусков алгоритма. После того, как приложение найдет решение задачи с нужным коэффициентом качества решения, завершит все тестовые запуски или истечет заданное пользователем время работы программы, произойдет вывод полученного решения или данных тестирования в файл и на экран.

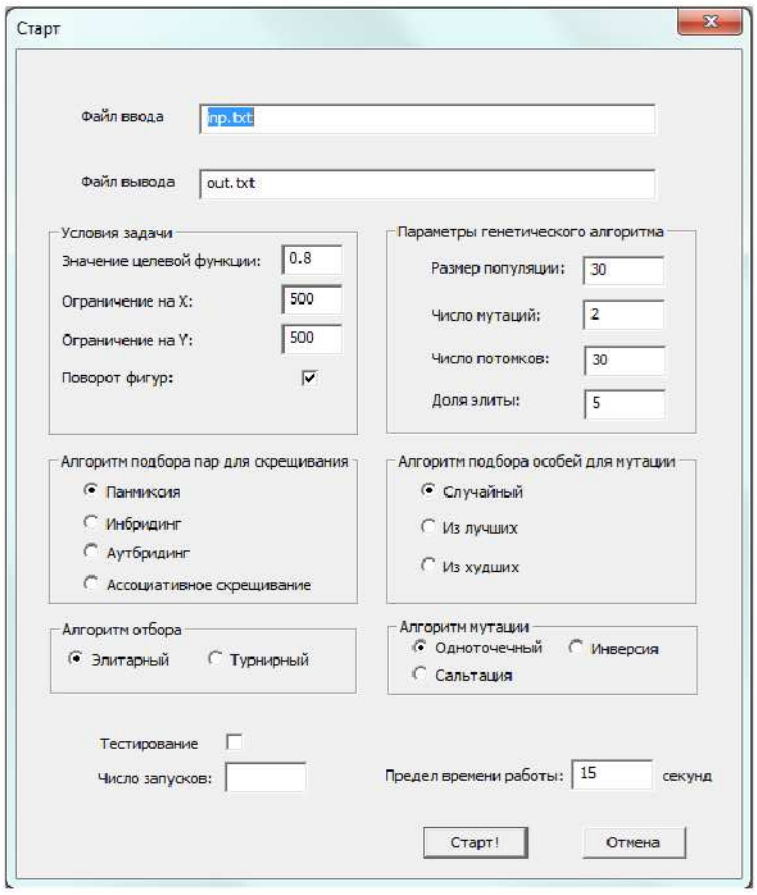


Рисунок 7 – Окно “Старт”

Как по окончании поиска решения, так и во время него, на экране будет показана графическая интерпретация полученного решения: набор прямоугольников, размещенный в соответствии с найденным решением. Это показано на рисунке 8.

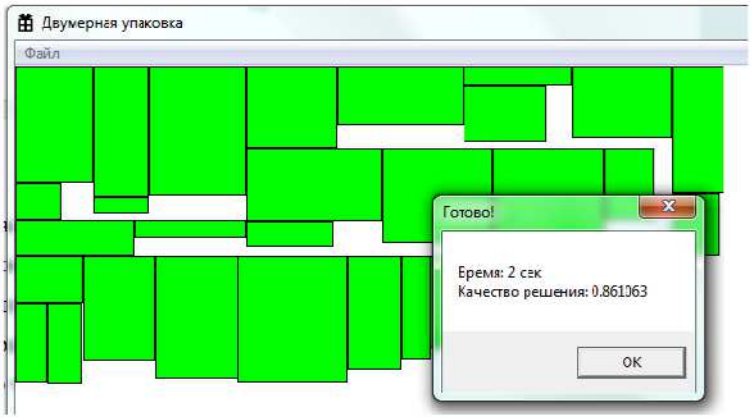


Рисунок 8 – Вывод полученного решения на экран

ЗАКЛЮЧЕНИЕ

В рамках данной работы были изучены методы приближенного решения NP-трудной задачи двумерной упаковки прямоугольников на плоскости и существующие программные разработки, направленные на решение данной проблемы, и впоследствии была спроектирована и реализована прикладная программа для получения приближенного решения этой задачи.

Поскольку рассматриваемая задача является NP-трудной, исследования в области приближенных методов ее решения не теряют своей актуальности. На основании анализа существующих компьютерных программ, реализующих методы решения этой задачи, был сделан вывод о целесообразности разработки полноценного приложения для нахождения наилучшего способа размещения прямоугольников на плоскости.

В качестве реализуемого в программе способа поиска приближенного решения был использован метод, основанный на сочетании алгоритма последовательного размещения прямоугольников и генетического алгоритма поиска наилучшего порядка размещения прямоугольников на плоскость. Выбор обоснован тем, что генетические алгоритмы дают хорошие результаты при решении оптимизационных задач с большим количеством входных данных и, в частности, стойки к сходимости к локальному оптимуму целевой функции [5].

В ходе работы было разработано приложение, соответствующее поставленной задаче. Оно предоставляет пользователю возможность ввести заранее подготовленные данные, задать различные условия задачи, а также требуемый желаемое качество решения. Кроме того, пользователь может изменять параметры генетического алгоритма для получения наилучшего результата. Это, несомненно, расширяет возможности прикладного использования программы. После введения всех входных данных приложение осуществляет поиск решения, и в случае успеха записывает его в файл на диске, а также демонстрирует пользователю найденный наилучший способ размещения прямоугольников на плоскости, что позволяет быстро и наглядно

получить представление о качестве полученного решения. Также приложение может оценивать корректность введенных пользователем данных и выдавать рекомендации по устранению ошибок в работе алгоритма, что, несомненно, упрощает использование приложения при решении прикладных задач. В приложении предусмотрен режим тестирования, в котором можно организовать некоторое количество запусков поиска решения с подсчетом и выводом средних значений коэффициента качества решения и среднего времени работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Старостин Н.В Многоуровневый эволюционно-генетический метод размещения прямоугольников на плоскости / Старостин Н.В., Силаев А.Н., Седых И.О. // Вестник Нижегородского университета им. Н.И. Лобачевского. 2009. № 5. С. 163-168.

2 Perdeck Matt. Fast Optimizing Rectangle Packing Algorithm for Building CSS Sprites. [Электронный ресурс]. – режим доступа: https://www.codeproject.com/Articles/210979/Fast-optimizing-rectangle-packing-algorithm-for-bu (Дата обращения: 17.05.2018)

3 Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи: М.: Мир, 1982. 416 с.

4 Holland John H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. First MIT Press edition. 1992. 232 pp.

5 Панченко, Т.В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. – Астрахань : Издательский дом «Астраханский университет», 2007. — 87 [3] с.

6 Каталог API(Microsoft) и справочных материалов [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/library/ (Дата обращения: 17.05.2018)

7 Страуструп Б. Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2011 г. – 1136 с.

8 Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Пер. с англ. – 4-е изд. – Спб.: Питер; Издательство «Русская Редакция»; 2008. — 720 стр.: ил.

9 C++ reference [электронный ресурс] URL: http://en.cppreference.com/w/cpp (Дата обращения: 17.05.2018)

10 Батищев Д.И. Применение генетических алгоритмов к решению задач дискретной оптимизации. Учебно-методический материал по программе

повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике» / Батищев Д.И., Неймарк Е.А., Старостин Н.В. Нижний Новгород. 2007. 85 с.