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

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

Допустить к защите
И.о. заведующего кафедрой
канд. физ.-мат. наук, доц.
_____ О.В. Гаркуша
(подпись)
_____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

РАЗРАБОТКА ГИБРИДНОГО АЛГОРИТМА И ПРИЛОЖЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧИ УПАКОВКИ

Работу выполнил _____ Д.А. Плигин
(подпись)

Направление подготовки 01.03.02 «Прикладная математика и информатика»

Направленность (профиль) «Системное программирование и компьютерные технологии (Математическое и программное обеспечение вычислительных машин)»

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

Нормоконтролер
ст. преп. _____ А.В. Харченко
(подпись)

Краснодар
2019

РЕФЕРАТ

Выпускная квалификационная работа 59 с., 3 ч., 20 рис., 1 табл., 23 источника.

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

Объектом исследования являются методы и программные средства получения приближенного решения задачи двумерной упаковки прямоугольников на плоскости.

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

В результате проведенной работы была разработан гибридный алгоритм и прикладная программа для получения приближенного решения задачи двумерной упаковки прямоугольников на плоскости, удовлетворяющая поставленной задаче.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная работа посвящена исследованию и разработке алгоритмов, а также программных средств решения задачи двумерной упаковки прямоугольников с различной шириной и высотой в прямоугольник минимальной площади. 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 передача по сети множества небольших изображений существенно медленнее, чем одного большого, которое представляет собой эти изображения, соединенные вместе [2]. Распространенной прикладной проблемой является также задача раскроя материала, которая может возникать на швейном производстве или производстве мебели [3]. В подобных случаях требуется отыскать такой способ разделения цельного куска некоторого плоского материала на требуемые части (т. е. детали продукции), чтобы при этом потери ценного сырья были минимальны. В этих ситуациях применение оптимизационных решений, полученных с помощью современных информационных технологий и прикладной математики, позволять сократить издержки производства, что крайне полезно с экономической и экологической точек зрения.

Актуальность исследования данной темы также 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 [4]. 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 приближенного решения задачи двумерной упаковки прямоугольников. В рамках данной работы был сформулирован ряд задач, отвечающих поставленной цели. В частности, в качестве основных были приняты следующие задачи:

– изучить литературу по теме рассматриваемой задачи и о методах ее решения,

– сформулировать математическую модель данной задачи и возможные способы представления ее решений,

– провести анализ существующих программных средств решения задачи двумерной упаковки,

– разработать программу, предназначенную для решения рассматриваемой задачи.

В главах 1 и 2 настоящей работы рассматриваются существующие алгоритмические и программные наработки по данной проблеме. В главе 3 представлены результаты проведенной работы по созданию программы, предназначенной для решения задачи двумерной упаковки прямоугольников на плоскости.

1 Обзор 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 без взаимных пересечений друг с другом, 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. В таком случае ограничение, связанное с условием отсутствия у прямоугольников пересечений друг с другом, будет иметь вид выражения 1.

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

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 (этот принцип размещения фигур проиллюстрирован на рисунке 1). 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 (следующая итерация, действия, описанные в шаге а, изображены на рисунке 2),

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 загрузки веб-страницы предназначенной для этого программой, браузером, поскольку увеличение

числа изображений на странице существенно повышает время ее передачи по сети, 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. Такие объединенные изображения называются CSS-спрайтами (англ. CSS-sprites), поскольку браузер на клиентском компьютере может разъединять для последующего отображения полученную картинку на изначальные компоненты путем исполнения кода, написанного на языке CSS. Результат оптимизации компоновки CSS-спрайта проиллюстрирован на рисунке 3.

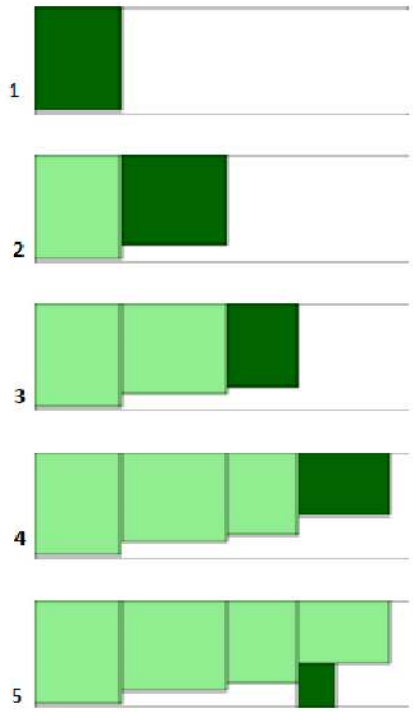


Рисунок 1 – Размещение фигур в порядке убывания площади

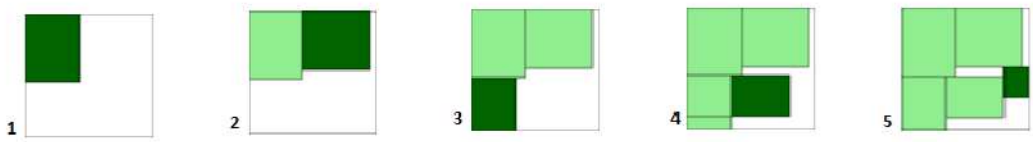


Рисунок 2 – Очередная попытка разместить фигуры после изменения размера вмещающего их прямоугольника

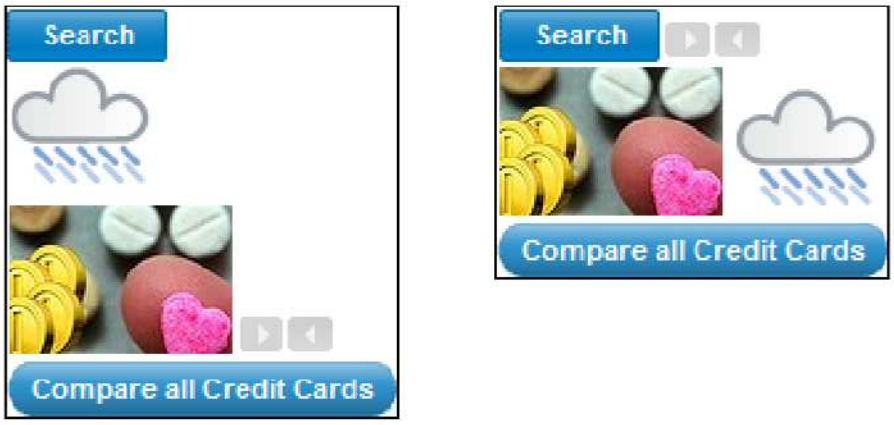


Рисунок 3 – Оптимизация CSS-спрайта

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 [5]. Им была предложена схема генетического алгоритма, впоследствии названная канонической [6].

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 выполняться 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, затем:

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 [6], 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].

Кроме того, оптимизационные методы, основанные на применении генетических алгоритмов, применяются для получения приближенного решения таких задач, как задача о ранце [7], задача коммивояжера [8], задача о назначениях [9], задача составления расписаний [10] и задача обучения и определения оптимальной структуры искусственных нейронных сетей [11-12]. Также проводятся разработки с целью найти эффективные численные методы решения нелинейных систем алгебраических уравнений [13], основанные на генетическом поиске.

1.2.3 Метод локального спуска

Существует ряд итерационных численных методов оптимизации, объединенных общей концепцией последовательного поиска по некоторому принципу очередного приближения к решению задачи в некоторой окрестности решения, полученного на предыдущем шаге (или заданного каким-либо образом на начальном шаге алгоритма). Для задач оптимизации числовых функций примерами таких методов могут служить метод градиентного спуска и другие методы, использующие для выбора следующего решения умноженный на минус единицу градиент оптимизируемой функции в точке, соответствующей решению, полученному на предшествующем шаге итерационного поиска [14].

Для задач дискретной оптимизации подобным методом является алгоритм локального спуска, который описан в [15]. Рассмотрим его общую схему. Допустим, есть некоторый способ формального представления элементов множества всевозможных решений некоторой задачи, а также на множестве таких представлений определена числовая функция, позволяющая

оценить качество всякого решения (то есть степень соответствия каждого решения требованиям к искомому элементу множества). Эта функция называется целевой функцией, и процесс поиска решения сводится к поиску её максимума. Предположим также, что на данном множестве представлений решений определена метрика, то есть, существует способ определения расстояния или степени схожести между двумя произвольно выбранными элементами этого множества. Необходимым также является наличие способа при заданном элементе множества построить все элементы, принадлежащие некоторой его окрестности, задаваемой фиксированным расстоянием до рассматриваемого элемента.

Пусть F – функция качества, определенная на множестве представлений решений задачи. Тогда алгоритм в общем случае будет иметь следующий вид:

а) Построить начальное решение и присвоить его представление переменной S,

б) Если в окрестности S нет нерассмотренных решений, перейти к шагу д,

в) Выбрать из окрестности S нерассмотренное ранее решение K,

г) Если F(K) больше, чем F(S), то присвоить переменной S значение K и перейти к шагу б,

д) Выдать найденное значение S.

Общая схема алгоритма локального поиска проиллюстрирована на рисунке 4.

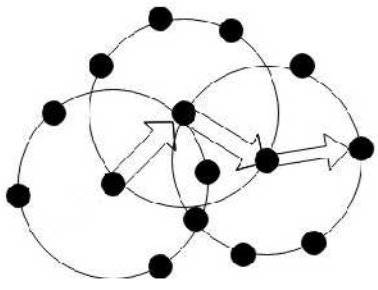


Рисунок 4 – Алгоритм локального спуска

На рисунке 4 черными кругами обозначаются решения задачи, а окружности, описанные вокруг некоторых из них, обозначают окрестности этих решений. Стрелки иллюстрируют процесс перехода к всё более и более качественным решениям.

Для рассматриваемой в данной работе задачи двумерной упаковки прямоугольников на плоскости можно конкретизировать данный алгоритм так, чтобы он подходил для ее решения. Например, в рассмотренном ранее методе решения, основанном на поочередном применении алгоритма последовательного размещения прямоугольников и генетического алгоритма, последний можно заменить на алгоритм локального спуска. Тогда в общую схему алгоритма локального спуска можно подставить в качестве решаемой задачи нахождение оптимального способа упорядочения фигур. В качестве представлений решений могут выступать перестановки номеров прямоугольников (предполагается, что множество размещаемых фигур упорядочено), то есть упорядоченные кортежи чисел от 1 до n (где n – число прямоугольников), в которых нет повторяющихся номеров. Тогда как функцию оценки качества решения F можно взять площадь наименьшего покрывающего фигуры прямоугольника, вычисляемую после их размещения на плоскости алгоритмом последовательного размещения в порядке, который однозначно задается используемым представлением – перестановкой номеров, а в качестве метрики на множестве этих перестановок можно взять, например, расстояние Хэмминга, то есть, для каждой пары перестановок расстоянием между ними будет число позиций, в которых они отличаются. Тогда из всякого произвольно взятого представления решения (перестановки чисел от 1 до n) можно получить наиболее близкие к нему в смысле расстояния Хэмминга вектора путем перестановки в нем двух произвольных различных чисел. Другими словами, для каждого представления решения можно построить его окрестность с расстоянием 2, так как при перестановке двух любых чисел в векторе-решении получается вектор, расстояние Хэмминга до которого равно двум.

Используя изложенное выше, можно получить алгоритм поиска решения задачи двумерной упаковки прямоугольников на плоскости, основанный на сочетании алгоритма последовательного размещения фигур, изложенного в [1] и алгоритма локального поиска, рассматриваемого в [15]. Получаемое с помощью этого метода решение является локально-оптимальным. Это значит, что в его окрестности нет решений с более высоким значением функции F. Очевидно, не всякое локально-оптимальное решение является в то же время глобальным оптимумом. В этом заключается главный недостаток алгоритма локального спуска. Попав в не очень хорошее с точки зрения значения функции F локально-оптимальное решение, алгоритм прекращает свой поиск. Чтобы покинуть точку локального оптимума, ему необходимо совершить шаг, ухудшающий значение целевой функции, но алгоритм локального спуска переходит только в решения лучшего качества.

2 Анализ некоторых существующих программных средств решения задачи двумерной упаковки

2.1 SVGnest

В настоящее время существуют программные 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.

Это веб-приложение, расположенное на сайте svgnest.com в сети Интернет. Главная страница приложения изображена на рисунке 5.

Рисунок 5 – Приложение SVGnest

Для начала работы с этим приложением необходимо ввести данные задачи в виде набора примитивов векторной графики, записанных в файл в формате SVG. Запись в файл является удачным решением ввиду того, что число размещаемых объектов может быть велико, и данные о них полезно сохранить в постоянную память компьютера. После нажатия на кнопку

Upload SVG и выбора файла с данными, все фигуры, информация о которых содержалась в файле, будут отображены на экране. На этом этапе необходимо выбрать из всех изображенных на экране фигур одну, которая будет являться контейнером для всех остальных, то есть, будет происходить поиск оптимального способа разместить все остальные фигуры внутри выбранной. После выбора приложение начинает поиск решения, выводя промежуточные результаты на экран. Визуализация промежуточных результатов является хорошим решением, так как позволяет в наглядной форме увидеть ход работы и оценить эффективность используемых алгоритмов. Этот этап работы программы изображен на рисунке 6.

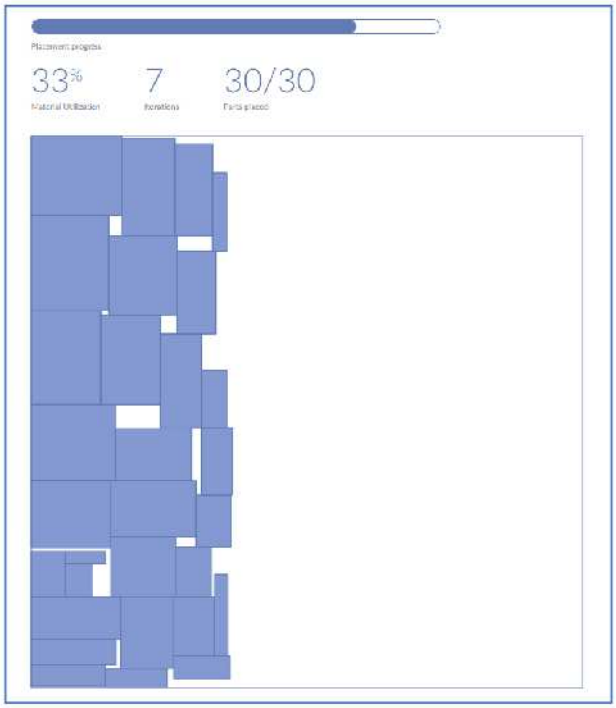


Рисунок 6 – Поиск решения приложением SVGnest

Процесс поиска решения можно прервать в любой момент, после чего становится активной кнопка Download SVG, нажав на которую можно загрузить полученный результат в виде файла с данными о фигурах (в т. ч. об их расположении) в формате SVG. Отсутствие возможности задания требуемого качества решения и автоматического завершения поиска в

случае, если было найдено достаточно хорошее размещение фигур, является недостатком данной программы, поскольку заставляет пользователя наблюдать за ходом решения и вручную прекращать поиск.

Исходный код данного приложения опубликован в сети Интернет под лицензией MIT, что позволяет свободно использовать данное ПО как в личных, так и в коммерческих целях, а также позволяет свободно изменять и распространять данный программный продукт.

Согласно документации [16], предложенной разработчиком программы, алгоритм поиска решения, используемый в данном веб-приложении основан на генетическом алгоритме, и в интерфейсе программы предусмотрена возможность менять некоторые параметры процедуры поиска решения (рисунок 7). Наличие такой возможности, несомненно, удачное решение, так как это позволяет сравнивать различные конфигурации алгоритмов с целью выбрать наиболее эффективные в общем случае, а также, возможно, подстраивать алгоритм под конкретные задачи.

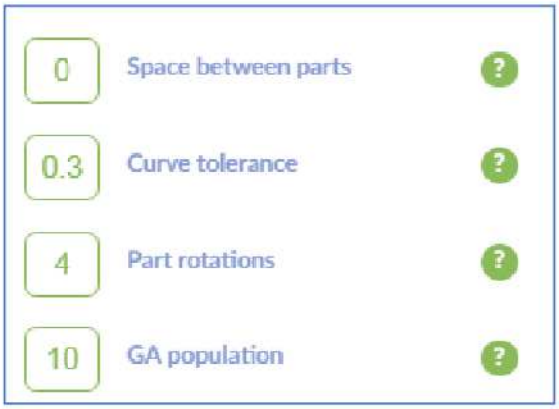


Рисунок 7 – Параметры поиска SVGnest

Несмотря на универсальность и производительность программы, предложенной проектом SVGnest 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.

2.2 Программные решения для генерации CSS-спрайтов

Поскольку одной из самых распространенных задач, связанных с упаковкой прямоугольников на плоскости, является задача объединения множества изображений (необходимость в этом возникает при разработке интернет-страниц), существуют программные решения, подобные рассмотренному в [2], предназначенные для использования в веб-разработке. Такие программы позволяют соединять множество различных изображений в одно. Это позволяет передавать по сети все присутствующие на странице картинки в одном файле, что уменьшает число запросов к серверу, а значит, и время загрузки веб-страницы. Исходные изображения извлекаются на компьютере-клиенте с помощью специальной разметки на языке CSS, которая передается отдельно от изображений. Несомненно, имеет смысл объединять изображения таким образом, чтобы размер полученного CSS-спрайта был минимальным, так как это позволит сократить объем передаваемого по сети файла, а значит, и уменьшить время загрузки страницы. Это позволяет отнести программное обеспечение, предназначенное для генерации CSS-спрайтов, к приложениям, решающим задачу двумерной упаковки прямоугольников на плоскости.

В сети Интернет можно найти множество сайтов с веб-приложениями, предназначенными для формирования CSS-спрайтов. В качестве примера такого приложения рассмотрим CSS Sprites Generator, размещенный в [17]. На главной странице приложения предлагается загрузить произвольное число файлов с изображениями, которые нужно объединить (рисунок 8), выбрать расстояние в пикселах, на которое эти изображения будут отстоять друг от друга в спрайте, а также выбрать алгоритм размещения картинок

(рисунок 9). На этой же странице будет представлен результат объединения, полученный CSS-спрайт (рисунок 10).

Рисунок 8 – Загрузка изображений

Рисунок 9 – Настройки CSS Sprites Generator

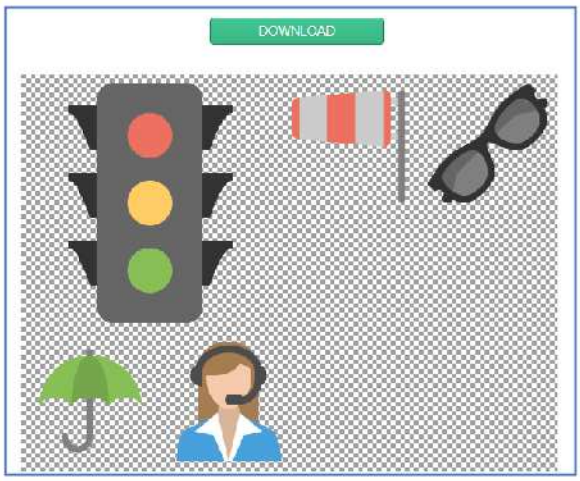


Рисунок 10 – Готовый CSS-спрайт

Стоит заметить, что в большинстве своем такие программные решения не предназначены для решения задачи упаковки большого числа

изображений (например, более 50), поскольку в веб-разработке задачи такой размерности возникают крайне редко. И ввиду небольшой размерности требования к качеству решения, то есть, размеру готового CSS-спрайта, обычно невысоки, что делает сомнительным использование этих наработок в решении задач, сводящихся к задаче двумерной упаковки прямоугольников на плоскости, и возникающих, скажем, на производстве мебели или швейном производстве, где возможная выгода от экономии ценного сырья при оптимальном раскрое листов материала существенна.

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 (в секундах или в количестве итераций), 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. Это также даст возможность в реальном времени получать сведения об эффективности применяемого метода поиска решения.

Для разработки заданной программы выбрана интегрированная среда разработки Microsoft Visual Studio. 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 [18-21]. Также крайне полезно будет использование системы контроля версий git в сочетании с системой управления репозиториями программного кода GitLab. Эта фактически общепринятая в сфере разработки программного обеспечения технология позволяет существенно упростить разработку и отладку сложных приложений. Также это принесет полезную возможность удобного доступа к различным конфигурациям и

состояниям программного проекта, что существенно облегчает анализ полученных результатов.

3.2 Алгоритм последовательного размещения прямоугольников

Рассмотрим алгоритм последовательного размещения фигур, который дан в [1]. 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 11. После нахождения точки, удовлетворяющей указанным условиям оптимальности и отсутствия пересечений, рассматриваемый прямоугольник размещается туда.

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. Это необходимо для достижения лучшего результата.

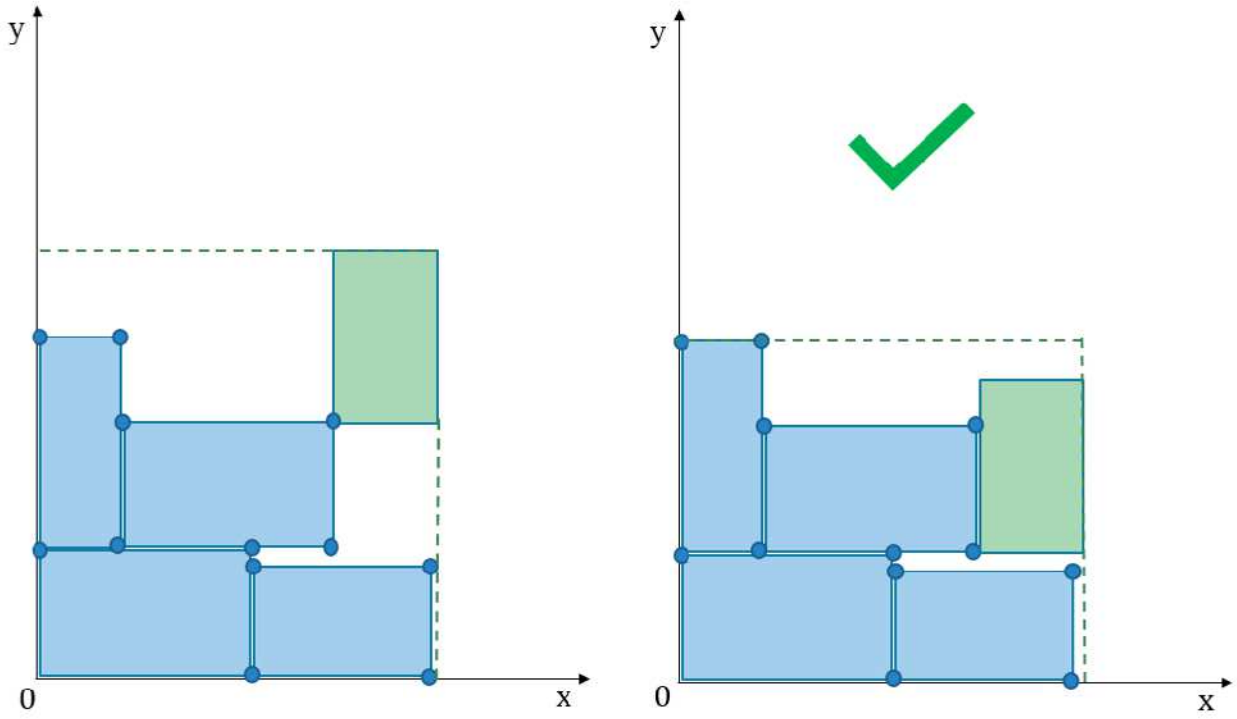


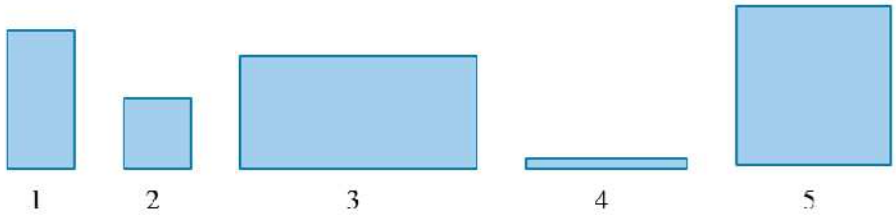
Рисунок 11 – Перебор узлов размещения

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. Этот алгоритм также рассмотрен в [1]. 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 12. 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 12 – 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 генетического алгоритма. 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 описанную в [22] реализацию оператора кроссинговера, то есть, скрещивания для порядкового представления решений, который имеет название OX кроссовера.

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 хорошо описан в [6].

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 [6].

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 [6].

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 13.

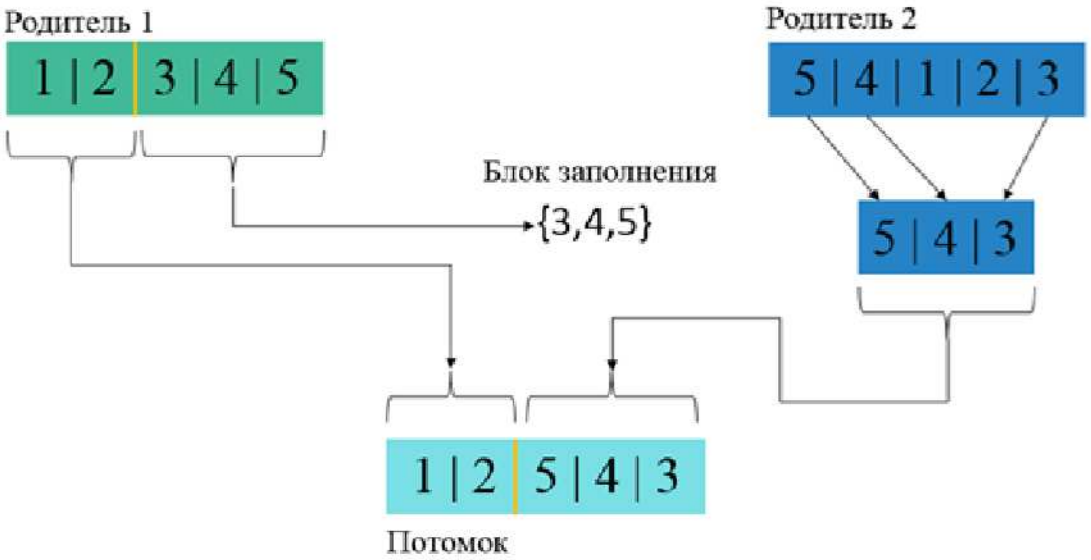


Рисунок 13 – Порядковый OX-кроссовер

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 [22].

Следующим шагом итерации цикла рассматриваемого генетического алгоритма будет выбор из полученной на этапе скрещивания популяции L решений, которые будут подвержены мутации и станут основой для получения новых, быть может, более оптимальных с точки зрения минимизации площади наименьшего покрывающего прямоугольника.

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 – это является разумным ограничением, диктуемым обоснованным требованием к большому разнообразию получаемых при рекомбинации решений на каждом произвольно взятом шаге эволюционного поиска. 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 14. 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.

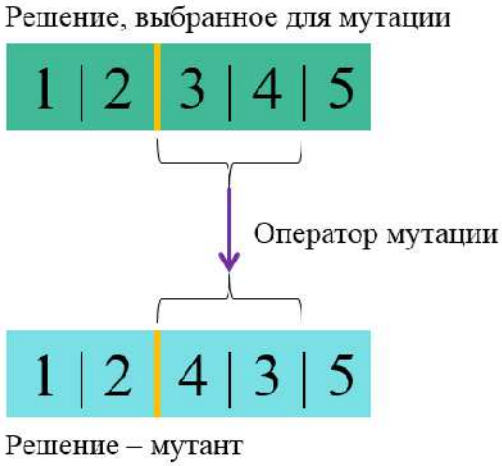


Рисунок 14 – Точечная мутация

Рассмотрим другой метод получения новых решений посредством мутации, метод инверсии. В методе инверсии, описанном в [6], 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 $k_1 \ldots k_n$ и $t_1 \ldots t_n$. 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 [22].

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 15. 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 ухудшаться. 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.

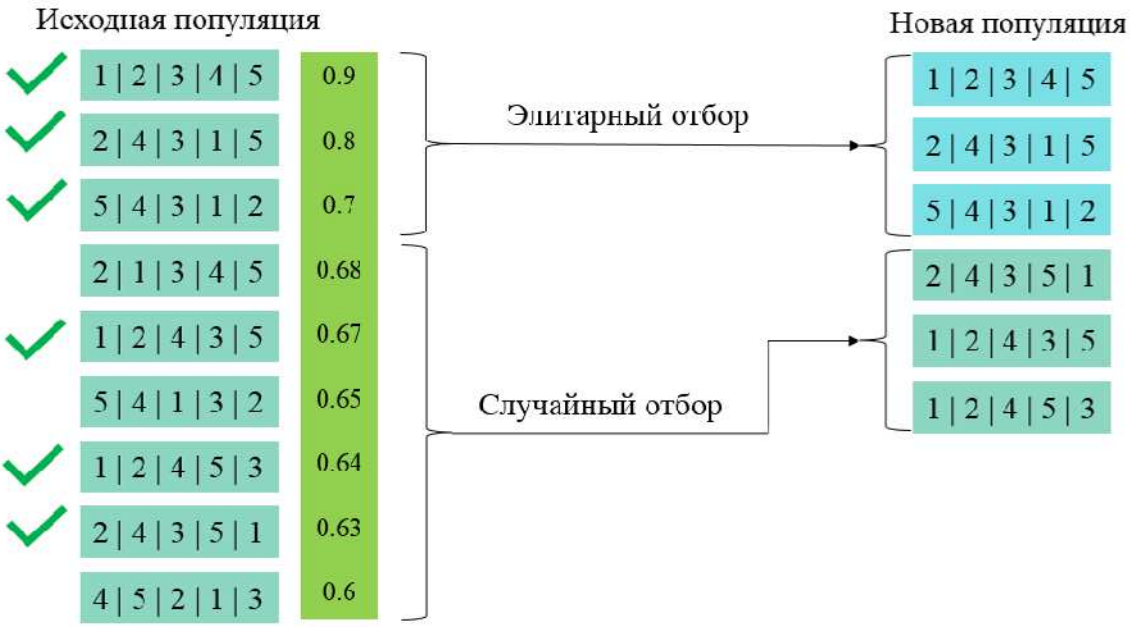


Рисунок 15 – Элитарный отбор

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 [6] 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 подпрограммами, реализующими основные решающие задачу алгоритмы, в приложении являются функции putRects и 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.

Также важной структурой данных, применяемой в приложении, является fitness_cache, объект шаблонного класса стандартной библиотеки языка программирования C++ unordered_map. Этот класс реализует такую структуру данных как хэш-таблица [23]. Эта структура данных реализует интерфейс ассоциативного массива и имеет сложность операций записи и поиска значений по ключу O(1) в среднем. Ее использование позволит кэшировать результаты работы функции putRects, то есть, значения функции приспособленности и конкретные схемы размещения фигур на плоскости. Через некоторое время после начала поиска, если в функцию putRects передается перестановка, то есть, код решения, который уже был обработан, и его коэффициент качества решения был найден, то вычисления не производятся заново, и нужные значения извлекаются из хэш-таблицы fitness_cache.

Функции evolve и putRects взаимодействуют в соответствии с описанным ранее выбранным алгоритмом поиска решения задачи двумерной упаковки прямоугольников на плоскости, основанном на сочетании генетического алгоритма и алгоритма последовательного размещения.

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 переменной, 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, определяющий количество особей, отбираемых по принципу элитарного отбора в процессе селекции решений в новую популяцию, для которой в дальнейшем будет вычислено значение фитнес-функции и получены новые решения, 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 генетического алгоритма выбранными алгоритмами скрещивания, мутации и селекции, при которых полученные результаты (время работы, наименьший, средний и наилучший найденные коэффициенты качества решения) сохраняются, а после того происходит вывод среднего времени, затраченного на поиск, наименьшего, среднего и наибольшего значения 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 текстовый 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 16. На оси абсцисс графика отмечено число итераций, прошедших с начала работы гибридного алгоритма с генетическим алгоритмом заданной конфигурации в составе, на оси ординат обозначена целевая функция, коэффициент качества решения задачи двумерной упаковки прямоугольников на плоскости.

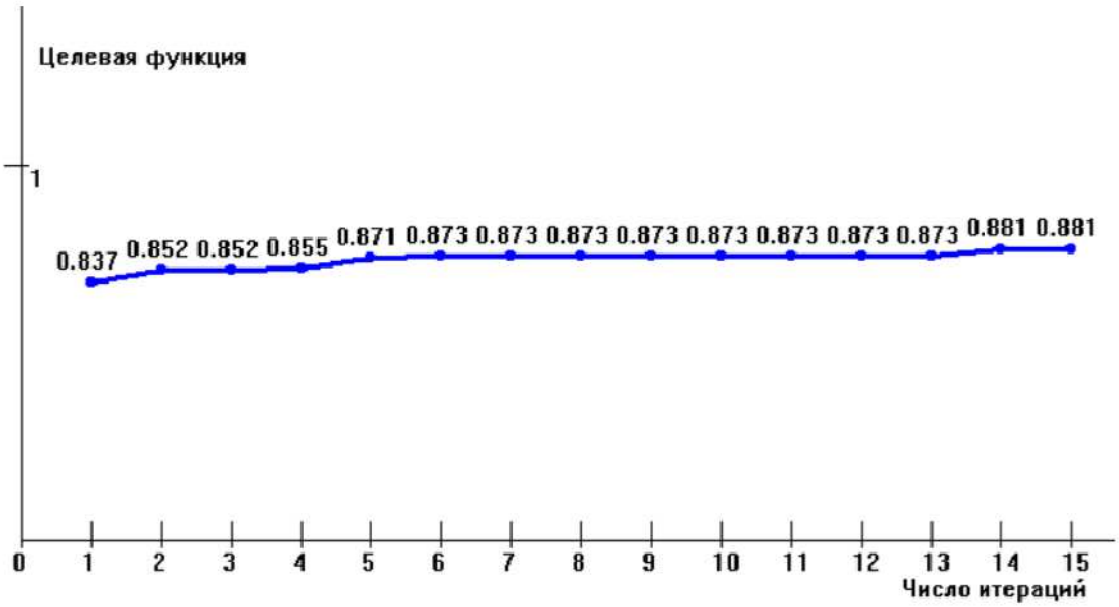


Рисунок 16 – Результат тестирования алгоритма

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”. 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. 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 или число итераций, по истечении которых 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. 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 17. Вид диалогового окна задания параметров программы представлен на рисунке 18.

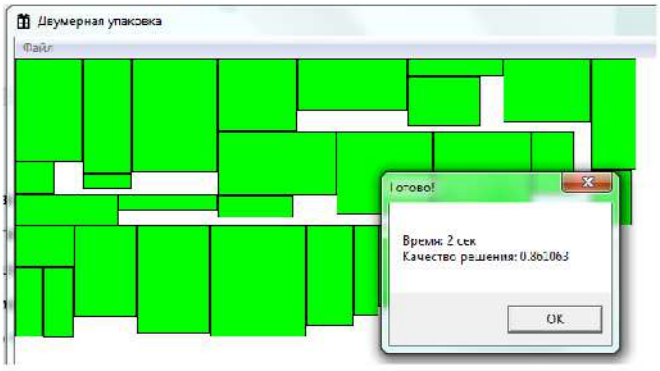


Рисунок 17 – Вывод полученного решения на экран

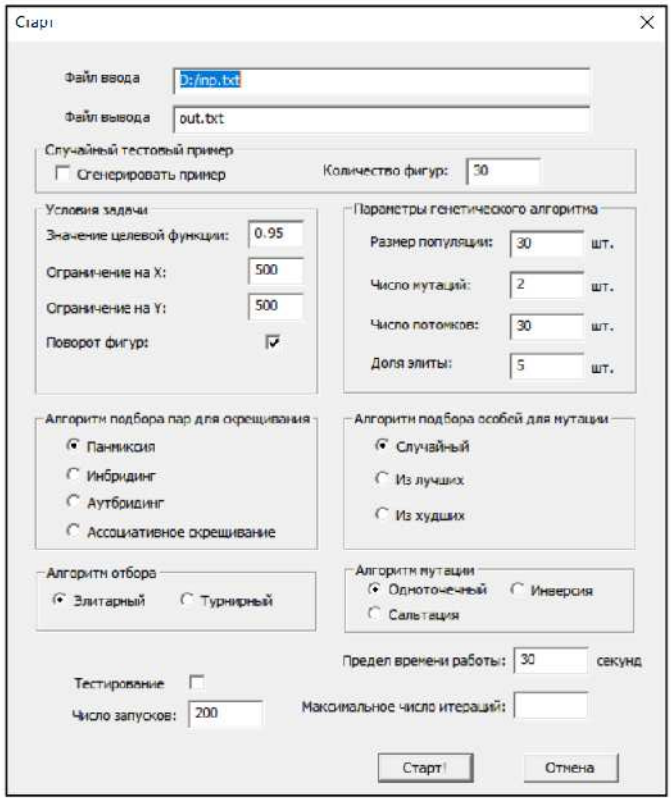


Рисунок 18 – Окно “Старт”

3.6 Проблема вырождения популяции

В процессе тестирования и отладки разрабатываемого приложения было обнаружено, что популяция особей, то есть, множество решений, рассматриваемых на определенном этапе поиска, теряла свое генетическое разнообразие после того, как с начала поиска решения проходило некоторое время. Иными словами, существенная часть популяции заполнялась абсолютно одинаковыми решениями, то есть имело место вырождение популяции или дрейф генов. Дрейф генов – это биологическое понятие, которое обозначает явление ненаправленного изменения частот аллельных вариантов генов в популяции, обусловленное случайными статистическими причинами. При работе генетических алгоритмов, как оказалось, возникает похожее явление.

В нашем случае утрата разнообразия популяции возникает вследствие того, что выбранные для использования при генетическом поиске решения

процедуры элитарного и турнирного отбора основаны на логике моделирования процесса естественного отбора, которая заключается в том, что выживают (то есть, оказывается выбранным для перехода в следующее поколение) особи с высокими показателями приспособленности (то есть, значениями целевой фитнес-функции). Это значит, что в ситуации, когда на определенном этапе поиска решения находится особь с относительно высоким коэффициентом качества решения, и после этого в течение некоторого времени программа не может отыскать решение лучше этого, то копии этого решения заполняют существенную часть популяции как самые приспособленные особи. В некоторых случаях популяция решений теряет свое генетическое разнообразие полностью, и в ней остаются только одинаковые особи. Пример вырожденной популяции можно видеть на рисунке 19, на котором изображены данные, полученные с помощью отладчика интегрированной среды разработки программ Microsoft Visual Studio. На этом рисунке представлено содержание структуры данных gen после вырождения популяции. Нетрудно видеть, что в популяции присутствует множество одинаковых особей с одинаково высоким значением фитнес-функции.

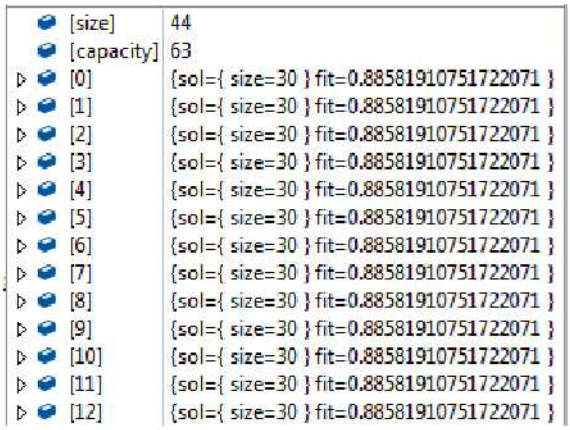


Рисунок 19 – Дрейф генов

Это явление, несомненно, является вредным, так как оно понижает эффективность генетического поиска, сильно замедляя или иногда даже

фактически останавливая процесс эволюции решений. Оно возникает вследствие того, что при скрещивании двух одинаковых особей единственным возможным результатом может быть только точно такая же особь, которая будет не лучше и не хуже своих родителей, а это значит, что решения в вырожденной популяции не могут эволюционировать, и, как следствие, прогресс в поиске решения будет крайне сильно замедлен. Также не исключено, что прогресса не будет вообще. На рисунке 20 можно видеть, как после того, как произошел дрейф генов, генетический поиск не достигает никакого прогресса.

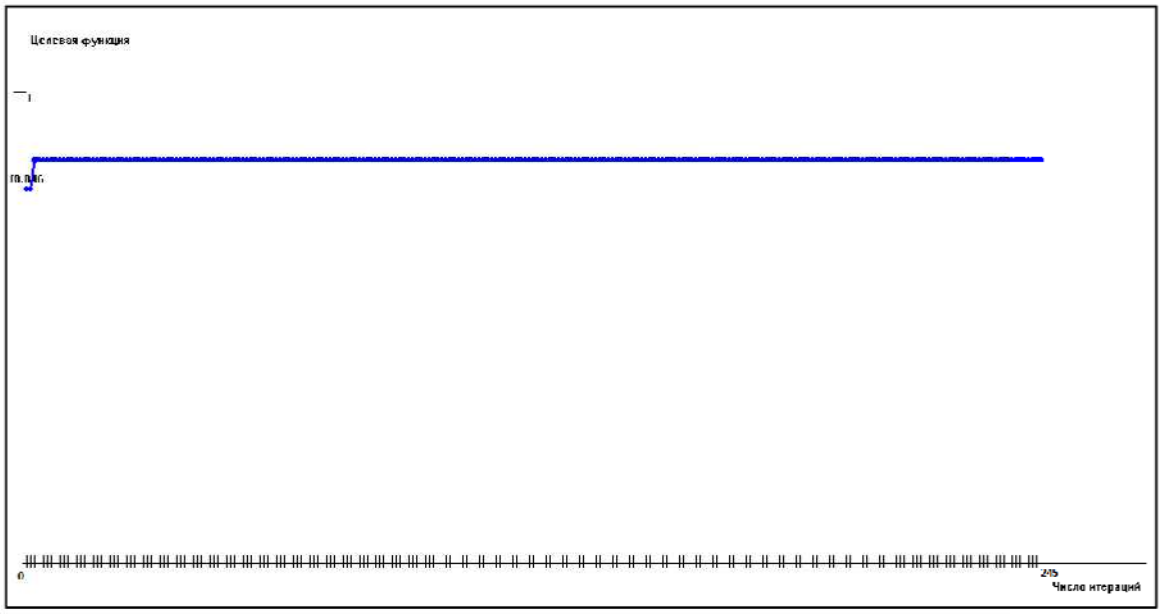


Рисунок 20 – Вырождение популяции

Возможные причины и некоторые способы борьбы с вырождением популяции рассмотрены в [6]. В частности, в число возможных способов исправления проблемы вырождения популяции генетического алгоритма входят:

– Изменение стратегии выбора родительских пар для скрещивания,

– Отслеживание появления в популяции идентичных особей и их удаление,

– Использование сильно разрушающего оператора кроссовера,

– Увеличение вероятности мутации.

В случае, если стратегия подбора родительских пар для скрещивания допускает образование пар из генетически идентичных особей, что является распространенной причиной вырождения популяции, то замена или модификация этой стратегии, позволяющая избежать скрещивания тождественных особей, может существенно снизить скорость потери генетического разнообразия популяции.

Проведение таких искусственных манипуляций как отслеживание появления в популяции идентичных особей и их удаление также может помочь в борьбе с дрейфом генов. В частности, при разработке рассматриваемого приложения для борьбы с вырождением популяции механизм формирования нового поколения был модифицирован. Новый алгоритм на каждой итерации поиска отслеживает степень генетического разнообразия текущей популяции путем подсчета особей, идентичных лучшему на данный момент найденному решению (так как вырождение генетического разнообразия популяции происходит из-за её заполнения копиями именно лучшего решения).

Увеличение вероятности мутации действительно помогает противодействовать дрейфу генов, так как рекомбинация, процесс получения новых решений, в ситуации с повышенной вероятностью мутации в большей степени зависит от мутации, чем от скрещивания. И так как скрещивание одинаковых решений является одной из основных причин вырождения популяции, то снижение его роли помогает решению проблемы.

Еще одним способом борьбы с вырождением популяции является инъекция случайных решений. Смысл этого приема состоит в том, что при достижении определенного достаточно высокого уровня однообразия решений в популяции (то есть, большого числа решений, совпадающих с наилучшим) происходит искусственное добавление в популяцию случайно сгенерированных решений задачи. Этот ход позволяет программе адаптивно сопротивляться дрейфу генов. Адаптивность заключается в том, что до

возникновения ситуации, близкой к вырождению популяции, алгоритм не меняет свой ход выполнения, а значит повышение вычислительной сложности сводится к минимуму. Данный прием был также применен в разработке рассматриваемого приложения и показал себя как крайне эффективное средство борьбы с вырождением популяции.

3.7 Применение алгоритма локального спуска для доводки решения

Одной из общих проблем генетического алгоритма, равно как и ряда других методов эвристических поиска (например, метода имитации отжига), является то, что в процессе отыскания оптимума из всего множества всевозможных решений задачи хорошие решения находятся достаточно быстро, но по мере достижения определенного уровня качества решения прогресс сильно замедляется – алгоритму трудно найти лучшие решения.

В качестве способа исправления такой проблемы можно использовать, например, рассмотренную ранее процедуру адаптивной инъекции случайных решений. В данном случае она заключается в том, что после того, как на протяжении некоторого наперед заданного количества шагов итерационного процесса не удается улучшить найденное решение, на очередном шаге генетического поиска производится добавление некоторого количества случайных решений задачи в популяцию с целью внести большее разнообразие в рассматриваемое множество решений, чтобы в процессе рекомбинации были получены существенно отличающиеся от имеющихся и, быть может, более качественные решения. Такой подход, несомненно, приносит свои результаты, но есть и другие способы преодоления стагнации процесса поиска на поздних этапах работы алгоритма, которые можно использовать совместно с указанным выше.

В частности, в случае, когда поиск решения заходит в тупик и в течение некоторого времени не получается улучшить найденный результат,

то возможно подключать к поиску решения другой алгоритм. Например, рассмотренный в главе 1 настоящей работы алгоритм локального спуска. Алгоритм локального спуска – схема оптимизации, которая предназначена для нахождения локальных оптимумов. Он требует задания перед началом работы некоторого начального решения, от которого сильно зависит качество полученного в итоге решения. Такое начальное решение может ему предоставлять генетический алгоритм. Результат, полученный генетическим алгоритмом, может быть улучшен алгоритмом локального спуска, и использование такого гибридного алгоритма может быть намного эффективнее чем применение алгоритма, основанного исключительно на генетическом поиске или алгоритма локального спуска со случайно задаваемым начальным решением. Такое улучшение полученного решения другим алгоритмом будем называть доводкой решения. Применение такого рода гибридизации также может быть адаптивным, то есть, алгоритм локального спуска может подключаться лишь тогда, когда на протяжении некоторого числа итераций генетический алгоритм не может улучшить найденное решение.

Указанные выше способы улучшения генетического алгоритма для повышения его способности преодолевать стагнацию поиска решения на поздних этапах работы программы были применены в разработке рассматриваемого в данной главе приложения. Полученный алгоритм является гибридным алгоритмом, так как сочетает в себе два различных метода поиска: генетический алгоритм и алгоритм локального спуска, используя преимущества каждого из них. Первый здесь служит средством глобальной оптимизации и находит хорошие начальные приближения для второго, который выступает средством уже локальной оптимизации и служит для доводки решения, полученного первым алгоритмом. Такое сочетание позволяет повысить эффективность поиска разработанной программой решения задачи двумерной упаковки прямоугольников на плоскости. Результаты тестовых запусков двух различных вариантов алгоритма:

генетического и гибридного представлены в таблице 1. Тестовые запуски в количестве 10 штук для каждого алгоритма производились на одном и том же наборе из 30 прямоугольников с требуемым значением целевой функции равным 0,9 и ограничением по времени равным 15 секундам.

Таблица 1 – Результаты тестовых запусков

Вид алгоритма	Среднее время работы, с	Минимальный коэффициент качества	Средний коэффициент качества	Максимальный коэффициент качества
Генетический	14,8	0,856415	0,884194	0,902146
Гибридный	14,9	0,877513	0,892443	0.913248

ЗАКЛЮЧЕНИЕ

В рамках данной работы были изучены методы получения приближенного решения NP-трудной задачи двумерной упаковки прямоугольников на плоскости и существующие программные разработки, направленные на решение данной проблемы. 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 гибридный алгоритм, основанный на сочетании генетического алгоритма и алгоритма локального спуска. 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 [6], а применение алгоритмов локального поиска для доводки решения позволяет улучшить результат.

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 минимального, среднего и максимального значения коэффициента качества решения, а также 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 M. 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: 17.05.2019)

3 Якушевич А.Е. Задача раскроя материала // Актуальные проблемы авиации и космонавтики. 2017. Т. 2. № 13. С. 88-90.

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

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

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

7 Королёва А.С., Юровская М.Б. Генетический алгоритм решения задачи о ранце // Студенческая наука XXI века : материалы VI Междунар. студенч. науч.-практ. конф. (Чебоксары, 1 окт. 2015 г.) Чебоксары: ЦНС «Интерактив плюс», 2015. № 3 (6). 196 с.

8 Семенов С.С., Ткачев Д.Ф., Педан А.В., Алисевич Е.А., Попов А.В., Воронцов О.С., Киселев Д.В., Климов И.С. Программа для решения задачи коммивояжёра с помощью генетического алгоритма // Навигатор в мире науки и образования. 2017 №1 С. 600-606.

9 Каширина И.Л., Семенов Б.А. Генетический алгоритм решения многокритериальной задачи о назначениях // Информационные технологии. 2007. №5. С. 62-67.

10 Еремеев А.В., Коваленко Ю.В. О задаче составления расписаний с группировкой машин по технологиям // Дискретный анализ и исследование операций. 2011. №5. С. 54-79.

11 Зернов Г.А., Семенкин Е.С. О выборе структуры нейронных сетей генетическими алгоритмами // Актуальные проблемы авиации и космонавтики. 2011. №7. С. 318.

12 Тененёв В.А., Тененёва А.В. Обучение нечетких нейронных сетей генетическим алгоритмом // Интеллектуальные системы в производстве. 2010. №1. С. 76-85.

13 Попова Е.И. Решение систем нелинейных алгебраических уравнений гибридными генетическими алгоритмами // Интеллектуальные системы в производстве. 2011. №2. С. 33-40.

14 Пантлеев А. В. Методы оптимизации в примерах и задачах: Учебное пособие/ А.В. Пантелеев, Т.А. Летова – 2-е изд. исправл. – М.: Высш. шк., 2005 – 544 с.

15 Руднев А. С. Алгоритмы локального поиска для решения задачи упаковки: дис. на соискание уч. степени канд. физ.-мат. наук. 2010.

16 Qiao J. SVGNest: A browser-based vector nesting tool [Электронный ресурс]. – режим доступа: https://github.com/Jack000/SVGnest (Дата обращения: 17.05.2019)

17 CSS Sprites Generator [Электронный ресурс]. – режим доступа: https://toptal.com/developers/css/sprite-generator (Дата обращения: 17.05.2019)

18 C++ reference [электронный ресурс] URL: http://en.cppreference.com/w/cpp (Дата обращения: 17.05.2019)

19 Каталог API (Microsoft) и справочных материалов [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/library/ (Дата обращения: 17.05.2019)

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

21 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.

22 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.

23 Кормен Т. Алгоритмы: построение и анализ : пер. с англ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест. - Москва: Изд-во МЦНМО, 1999. 955 с.