Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Российский экономический университет имени Г. В. Плеханова»
Институт математики, информационных систем и цифровой экономики
Базовая кафедра цифровой экономики института развития информационного общества

КУРСОВАЯ РАБОТА

по дисциплине «Технологии разработки современных программных комплексов»

на тему «Двумерная упаковка»

Выполнили
обучающиеся группы
15.11Д-МО11/19б
очной формы обучения института математики, информационных систем и цифровой экономики
Бабурян Армен Каренович,
Горшкова Анна Константиновна,
Кучерова Кристина Сергеевна,
Пушин Никита Андреевич,
Соколов Максим Владимирович
Научный руководитель:
старший преподаватель
Чапкин Н. С.

Москва – 2021

Введение

В двумерной задаче упаковки полос дается полоса конечной ширины n, но бесконечной высоты, и набор прямоугольных элементов, каждый из которых имеет ширину не более n. Цель состоит в том, чтобы упаковать все предметы в полоску, чтобы минимизировать используемую высоту. Элементы не могут перекрываться и поворачиваться.

Актуальность темы обусловлена тем, что задачи двумерной упаковки занимают важное место в современной комбинаторной оптимизации и привлекают внимание многих ученых как в России, так и за рубежом. Также задача раскроя-упаковки имеет большую практическую значимостью. Они относятся к задачам, решающим и оптимизирующим проблемы материалоемких производств, в которых одним из основных факторов является снижение себестоимости выпускаемой продукции - рациональное использование ресурсов. Подобные задачи, для решения которых необходимо применять алгоритмы комбинаторной оптимизации, встречаются все чаще в современном мире к ним относятся: резка деревянных досок, стальных пластин или рулонов бумаги, многомерное планирование ресурсов, проблем: рюкзака, упаковки контейнеров, загрузки контейнеров, складских запасов и загрузки поддонов. Чтобы решить эти проблемы исследователями Л. В. Канторовичем и В. А. Залгаллером было предложено использовать линейное программирование с неявно заданной матрицей ограничений. На сегодняшний день для решения задач упаковки используются различные оптимизационные алгоритмы. В. М. Картаком и Э. А. Мухачевой разработан метод ветвей и границ для решения одномерной задачи упаковки в контейнеры. В работах Э. Х. Гимади и В. В. Залюбовского рассматриваются асимптотически точные алгоритмы. Для решения задачи гильотинной прямоугольной упаковки в контейнеры М. И. Свириденко была предложена асимптотическая полиномиальная аппроксимационная схема. Задачи упаковки относятся к классу NP–трудных задач комбинаторной

оптимизации. Многие из них являются NP–трудными в сильном смысле. В связи с этим большое значение приобретает разработка и исследование различных алгоритмов упаковки.

Целью данной курсовой работы является исследование, описание и реализации алгоритмов двумерной упаковки, и визуальное представление методов, а также расширение теоретических знаний в области алгоритмов, реализации приложений, интерфейсов и их практическое применение в процессе разработки.

Для достижения поставленной цели необходимо решить следующие **задачи**:

- рассмотреть понятия, сущность и необходимость комбинаторной оптимизации.
- изучить и проанализировать алгоритмы двумерной упаковки.
- разработать трехуровневое приложение.

Объектом исследования являются алгоритмы и методы двумерной упаковки.

Предметом исследования является трехуровневое приложение с алгоритмами полного перебора и оптимальными алгоритмами.

Методы исследования. В данной курсовой работе применяются такие общенаучные методы исследования, как анализ, эксперимент и моделирование.

Глава 1. Методы двумерной упаковки.

1.1 Offline алгоритмы двумерной упаковки.

Так же, как и во всех алгоритмах двумерной упаковки необходимо разместить конечное число объектов прямоугольной формы в контейнер с заданной шириной и бесконечной высотой. Необходимо минимизировать высоту заполненного контейнера и максимизировать плотность упаковки. При распределении объектов необходимо учитывать, что их нельзя переворачивать.

В offline алгоритмах заранее известен размер всех упаковываемых прямоугольников, поэтому сначала все объекты сортируются, по выбранному критерию, группируются и размещаются в подходящие по размеры места. На таком принципе основываются алгоритмы:

- Level (уровневые);
- Plane (плоские);
- Shelf (шельфовые).

Алгоритмы уровня — это автономные алгоритмы. Их первый шаг включает предварительную сортировку списка прямоугольников в порядке уменьшения высоты, т. е. Первый прямоугольник, который будет упакован, будет самым высоким, а последний-самым коротким. Затем упаковка состоит из ряда уровней. Каждый прямоугольник последовательно упаковывается в контейнер, помещая его нижний край так, чтобы он лежал на одном из уровней. Первый уровень — это дно корзины, и каждый новый уровень определяется путем проведения горизонтальной линии через корзину через верхнюю часть самого высокого (т. е. первого) прямоугольника на предыдущем уровне.

1.1.1 Алгоритм NFDP

Алгоритм NFDP (Next Fit Decreasing High) — это алгоритм уровня, который использует подход следующего соответствия для упаковки отсортированного списка прямоугольников. Прямоугольники упакованы, выровнены по левому

краю на уровне, пока следующий прямоугольник не подойдет. Этот прямоугольник используется для определения нового уровня, и упаковка продолжается на этом уровне. Более ранние уровни не пересматриваются. На Рисунке 1 изображены два контейнера, упакованные по алгоритму NFDH в первом распределены вытянутые прямоугольники, а во втором объекты имеют более квадратную форму.

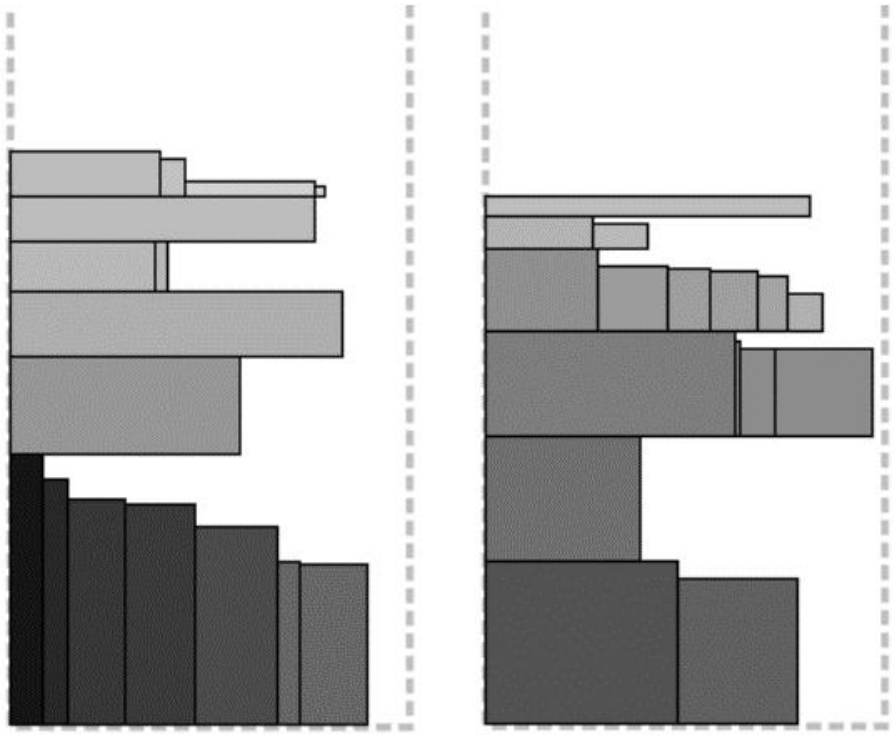

Рисунок 1. Иллюстрация алгоритм NFDH.

1.1.2 Алгоритм FFDH

Следующий алгоритм похож на описываемый ранее с отличием, что для каждого следующего прямоугольника ищется место не только на последнем уровне, а начиная с самого нижнего. Алгоритм FFDH (First Fit Decreasing High) использует подход первого соответствия. Каждый прямоугольник помещается на первый (т. е. самый низкий) уровень, на который он поместится. Если ни на одном из текущих уровней нет места, запускается новый уровень.

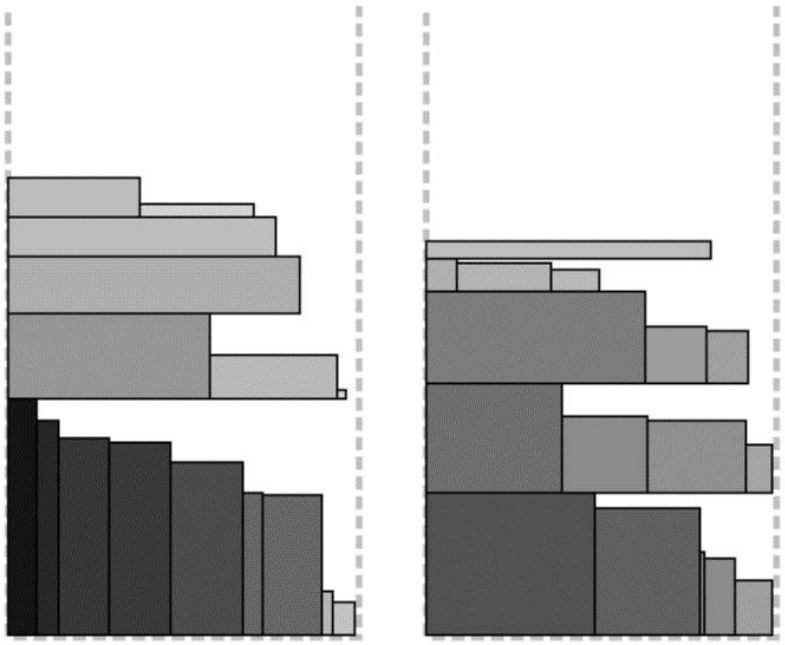

Рисунок 2. Иллюстрация алгоритма FFDH

1.1.3 Алгоритм BFDH

Алгоритм Best Fit Decreasing Hihg – модификация FFDH. Его идея состоит в том, что из уровней, подходящих для упаковки очередного прямоугольника, выбирается не первый, а такой, на котором останется минимум места после упаковки текущего прямоугольника, т. е. выбирается минимальное подходящее пространство, что способствует лучшему заполнению уровней.

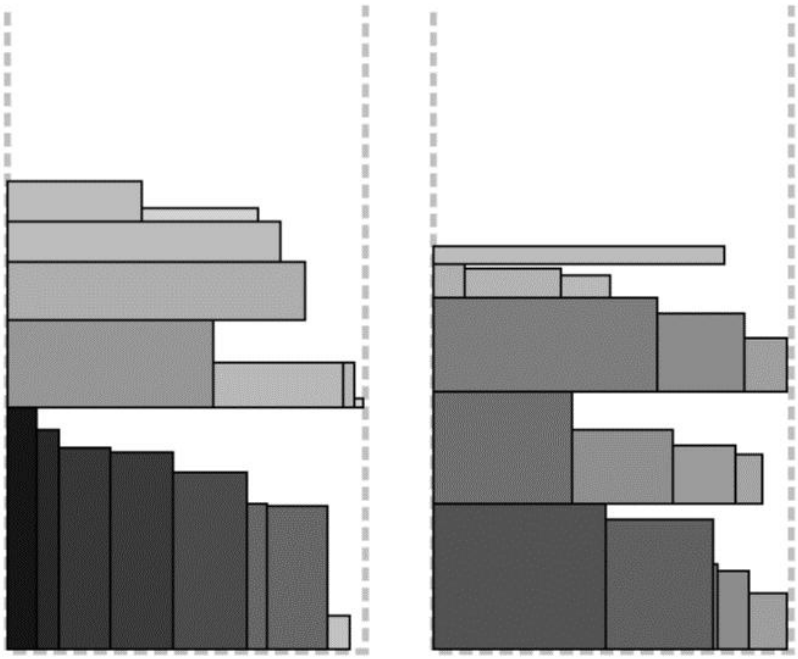

Рисунок 3. Иллюстрация алгоритма BFDH

1.1.4 Алгоритм Split Fit

Алгоритм Split Fit, является улучшенной версией FFDH алгоритма. Решение реализуется по принципу «разделяй и властвуй». Первым действием отбираются прямоугольники шире 1/2 ширины контейнера. Такие объяты упаковываются в первую очередь. Из отобранных прямоугольников выбираются превышающие ширину 2/3 полосы. Выбранные элементы упаковываются алгоритмом FFDH. Над ними упаковываются оставшиеся, которые превышают 1/2 длины, но меньше 2/3 полосы таким же алгоритмом что и предыдущие. Все оставшиеся прямоугольники, которые уже, чем 1/2 полосы, упаковываются с помощью того же FFDH в первую очередь в образовавшийся регион, а если не помещаются — то сверху

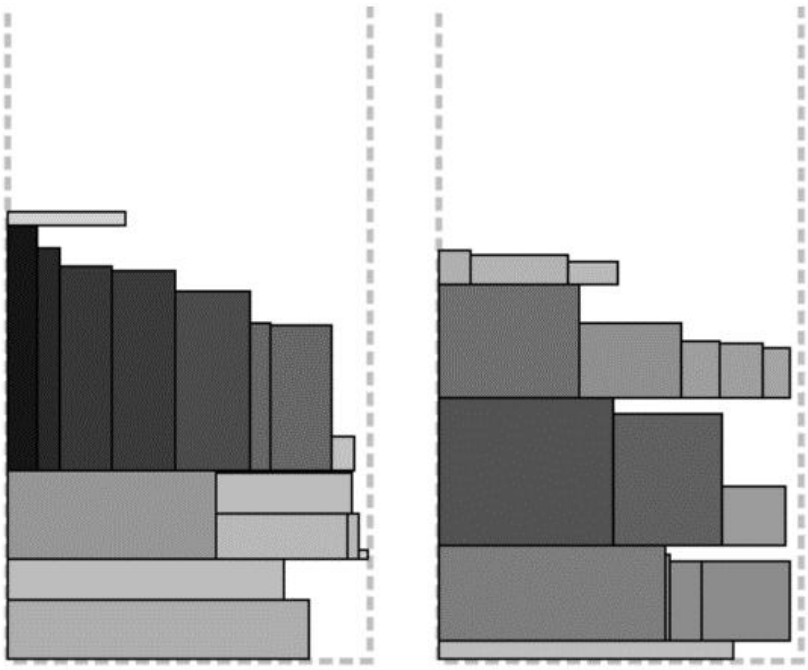

Рисунок 4. Иллюстрация алгоритма Split Fit

1.1.5 Алгоритм Floor Ceiling No Rotation

Принцип работы алгоритма FCNR: прямоугольники сортируются по не-возрастанию высоты и применяется алгоритм BFDH с некоторыми модификациями. У каждого уровня есть «пол» (floor) и «потолок» (ceiling). Пока есть возможность, прямоугольники пакуются на «пол» слева направо. Когда место заканчивается, предпринимается попытка упаковать на «потолок» справа налево; если нет места на потолке, то только тогда начинается новый уровень. В лучших традициях BFDH, на каждом шаге просматриваются все уровни — сначала «пол», затем «потолок» — на наличие наиболее подходящего места. Метод удачно подходит для упаковки по «потолкам» самые мелкие прямоугольники.

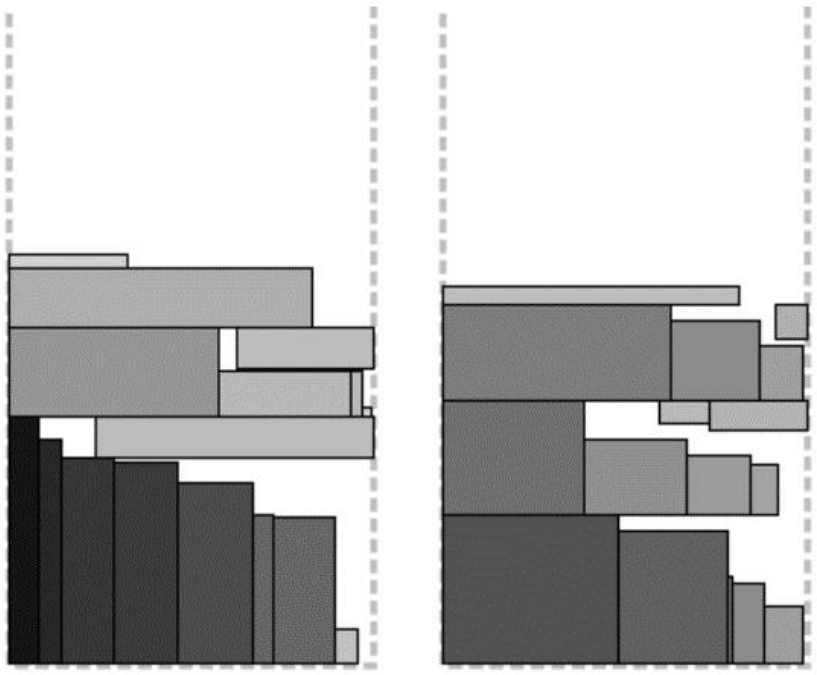

Рисунок 5. Иллюстрация алгоритма Floor Ceiling No Rotation

Подробно рассмотрев основные алгоритмы уровней, прейдем к плоским. Их особенность заключается в том, что они не используют принцип распределения блоков(прямоугольников) по уровням. Поэтому такие алгоритмы также называют «без уровневые».

1.1.6 Алгоритм Sleator

Первым алгоритмом для изучения стал Sleator. За основу взят принцип упаковки рюкзака: поместить на дно контейнера самый крупный элемент, затем сверху заполнить меньшего размера объектами (Рис. 5). Заполнение меньшими объектами выполняется по определенной последовательности: из прямоугольников выбирается те, что шире половины полосы и укладываются хаотично с выравниванием по левому краю. Оставшиеся сортируются по убыванию высоты и заполняются друг на друга в разных направлениях (слева-справа), в соответствии с принципом, где меньше всего элементов на текущий момент.

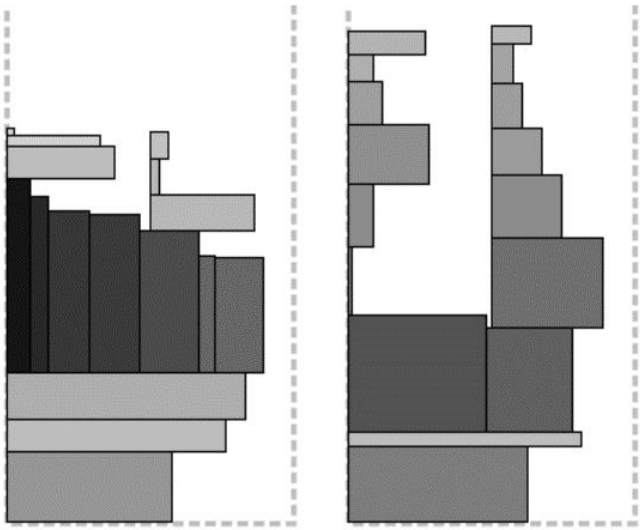

Рисунок 6. Иллюстрация алгоритма Sleator

1.1.6 Алгоритм Burke

Алгоритм Burke так же является без уровневым в нем вводится дополнительный массив в начале, заполненный нулями, по мере выполнения алгоритма можно отслеживать наименее заполненные области и их ширину(рис.7). Все имеющиеся объекты сортируются по убыванию ширины, затем на каждом шаге выполняется следующая последовательность действий:

- Вычисляется позиция самой низкой области по индексу минимального значения в массиве.
- Выбирается наиболее подходящий по параметрам прямоугольник. Он должен поместиться в выбранное пространство и максимально заполнять его по ширине.
- Если такой прямоугольник существует – размещается в эту область. Способов выполнения данного пункта 3. Первый - расположить по левому краю. Второй – поместить ближе к более высокому соседу, если он расположен на краю полосы, тогда ближе к ней. И последний - ближе к менее высокому соседу, если один из соседей — край полосы, то дальше от края. К значениям массива, соответствующим ширине прямоугольника, прибавляется его высота.

- Если подходящего прямоугольника нет, область «заполняется» путем выравнивания ее высоты до высоты ближайшего края.

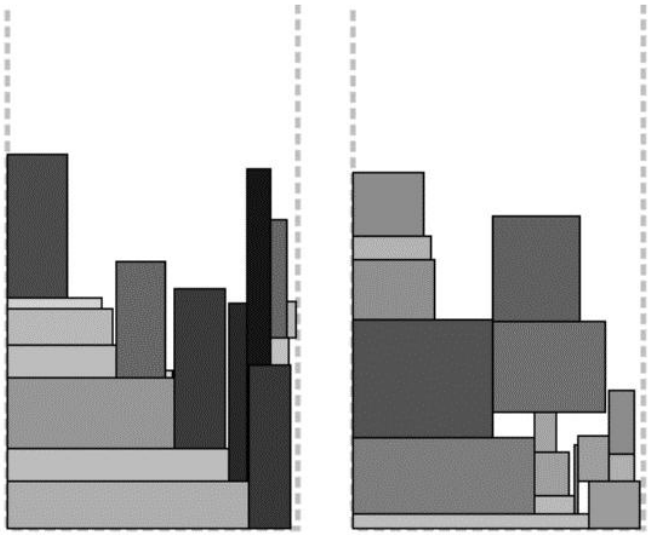

Рисунок 7. Иллюстрация алгоритма Burke

1.1.7 Shelf algorithms

Шельфовые алгоритмы – алгоритмы, для реализации которых необходимо знать точную высоту прямоугольников. Они распределяются по уровням в зависимости от высоты. Первый предмет помещается на первый шельф затем сравниваются высоты остальных прямоугольников, если они выше, то помещаются на следующий шельф, в противном случае ставятся рядом с первым предметом. Высота шельфа определяется как rk, где $r_{k+1} < h(L_i) \leq r_k$ для некоторого целого k и высоты первого упаковываемого прямоугольника $h(L_i)$.

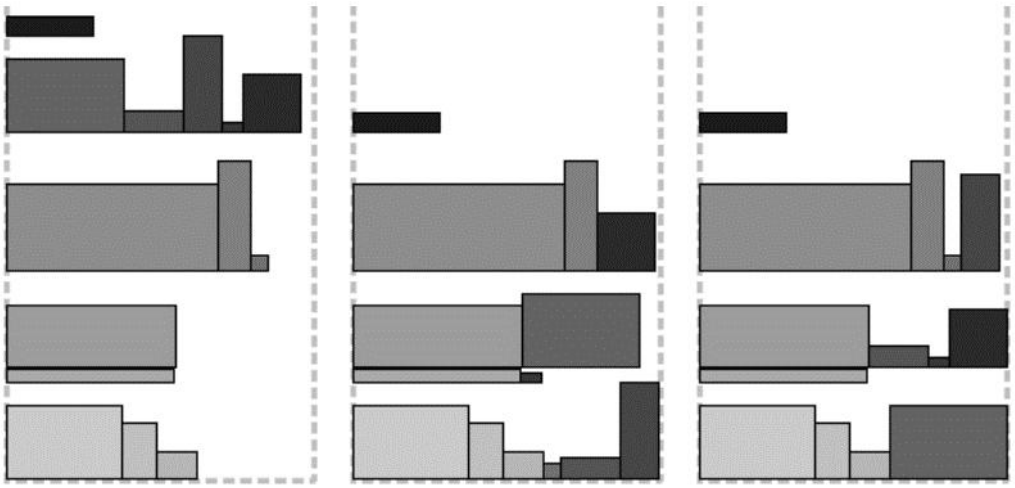

Рисунок 8. Иллюстрация shelf algorithms

1.1.8 Harmonic Shelf

Harmonic Shelf пакует на один уровень прямоугольники со схожей высотой и шириной. В результате получится большее количество плотно заполненных шлейфов. Вводится дополнительный параметр *M* для расчета допустимой ширины.

Общей ширине полосы присвоим значение единицы, делится на *M* интервалов $I_1..I_M$. Значение *M* должно находится в пределах [3; 12]. Ширина интервалов рассчитывается по формуле:

$$I_p = \left(\frac{1}{p+1}, \frac{1}{p}\right], 1 \le p < M \quad ; \quad I_M = \left(0, \frac{1}{M}\right]$$

Для каждого прямоугольника рассчитывается значение *p*. Параметр *k* для высоты рассчитывается аналогично шельфовым алгоритмам. Проверяются два условия: есть ли для прямоугольника место на каком-нибудь шельфе высотой r^k и соответствует ли его *p* уже упакованным. Если хотя бы одно условие не выполнено, для него создается новый шельф.

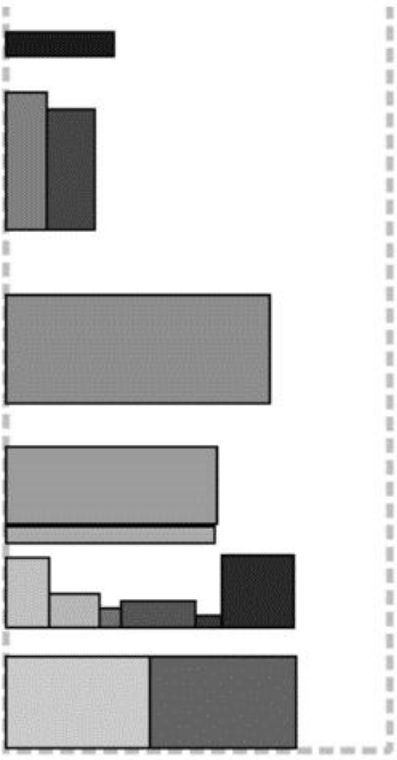

Рисунок 9.Иллюстрация Harmonic Shelf

1.2 Online алгоритмы двумерной упаковки.

В онлайн алгоритмах условия задачи видоизменяется. В offline алгоритмах были заранее известны размеры прямоугольников, теперь же параметры объектов узнаются в режиме реального времени. Задача заключается в быстром распределении предметов без возможности дальнейшего передвижения. Цель остается такой же – минимизировать общую высоту упакованных прямоугольников. Рассмотрим сами алгоритмы.

1.2.1 Level algorithms

Подход всех уровневых алгоритмов в том, что полоса разделяется на уровни, исходя из высоты имеющихся на данном этапе прямоугольников. Прямоугольники размещаются вдоль основания текущего уровня слева направо, пока это возможно; прямоугольник, который не поместился, упаковывается на следующий уровень. Высота каждого уровня определяется по самому высокому прямоугольнику в нем. Существует три варианта

упаковки:

Next Fit Level — когда открывается следующий уровень, предыдущий «закрывается» и в него больше вносить объекты;

First Fit Level — на каждом шаге алгоритма просматривается каждый уровень, начиная с самого нижнего, и прямоугольник упаковывается на первый подходящий, на котором достаточно места;

Best Fit Level — на каждом шаге алгоритма просматриваются все уровни, и определяется наиболее подходящий — тот, на котором после упаковки останется минимум места.

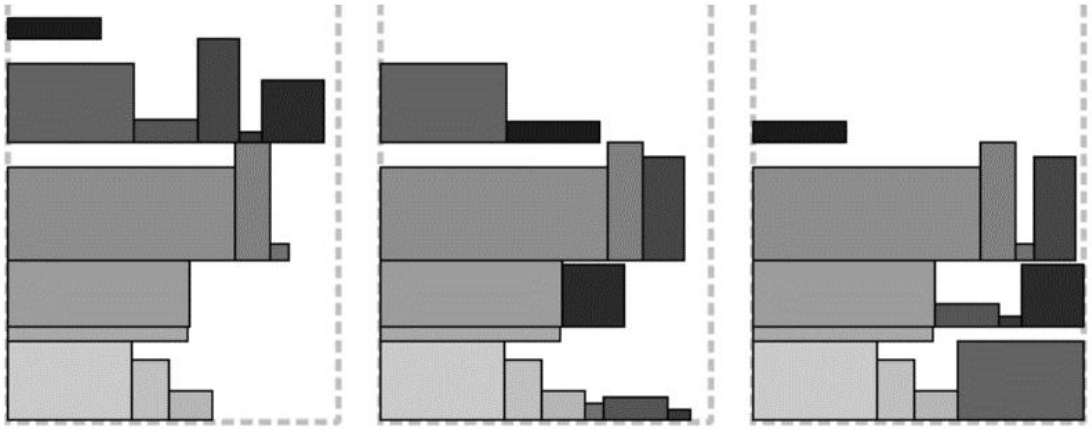

Рисунок 10. Иллюстрация Level алгоритмов

1.2.2 Bi – Level Next Fit

Bi – Level Next Fit – модификация Next Fit Level. В данном алгоритме создаются по два уровня на которых действует своя последовательность действий.

Все нечетные уровни работают по принципу: первый пришедший прямоугольник упаковывается по левому краю, остальные — справа налево, пока достаточно места. Затем уровень закрывается, и его высота определяется по самому высокому прямоугольнику в нем.

во всех четных уровнях: если на нижнем уровне всего один прямоугольник, уровень упаковывается аналогично нижнему. Если два и больше, то первый прямоугольник упаковывается над более низким из двух прижатых к краям полосы, и также прижимается к краю полосы; все последующие

упаковываются справа налево, независимо от того, где был упакован первый — слева или справа.

Звучит путано и не сразу понятно, к чему такие танцы с бубном. Единственное, что очевидно обеспечивает этот алгоритм — это более равномерное распределение прямоугольников по объему полосы, без перекоса к левому краю. Но мы еще вернемся к этой стратегии, когда будем рассматривать алгоритмы компрессии.

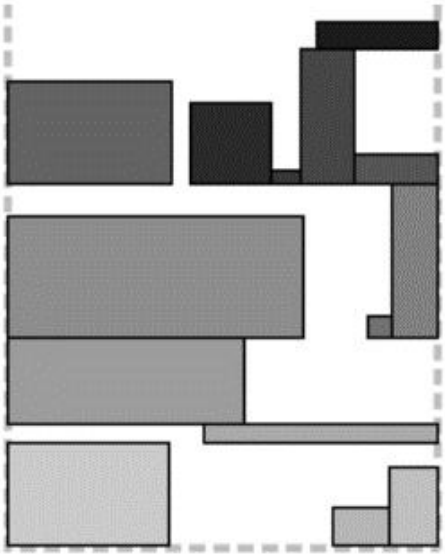

Рисунок 11. Иллюстрация Bi - Level Next Fit алгоритма

1.2.3 $Azar_y$

Для деления на уровни вводится некое пороговое значение $0 < Y < 1/2$. Ширина прямоугольников масштабируется в соответствии с шириной полосы, которая принимается за единицу. Прямоугольники высотой $2^{j-1} < h(L_i) \leq 2^j$ и шириной $2^{-x-1} < h(L_i) \leq 2^{-x}$ упаковываются на один уровень. То есть уровень характеризуется парой (x, j) для некоторого целого j и натурального x.

Прямоугольники шириной не менее Y нарекаются *буферными* и пакуются каждый на отдельный уровень. То есть, если нашелся такой прямоугольник, необходимо создать новый уровень, высотой равный ему. Все остальные (не-

буферные) пакуются на первый уровень с такой же парой (x, j). Если места не хватило, не-буферный прямоугольник может стать первым в новом уровне, тогда высота этого уровня будет 2^j.

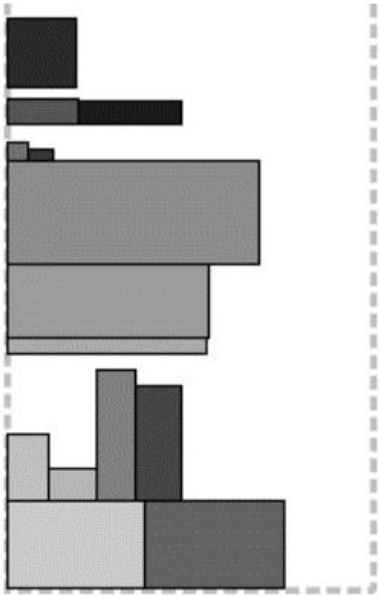

Рисунок 12. Иллюстрация алгоритма $Azar_y$

1.2.4 Compression algorithms

Компрессионные алгоритмы основаны на BiNFL.

Общее отличие — верхний уровень всегда упаковывается слева направо.

Compression Part Fit: Для каждого прямоугольника, упакованного на верхний уровень, проверяется, есть ли под ним место на нижнем уровне. Если можно сдвинуть его на нижний уровень так, что часть его останется на верхнем, он сдвигается.

Compression Full Fit: Для каждого прямоугольника, упакованного на верхний уровень, проверяется, может ли он полностью сместиться вниз на предыдущий уровень. Если может - смещается. Освободившееся таким образом место на уровне может быть использовано для следующего прямоугольника.

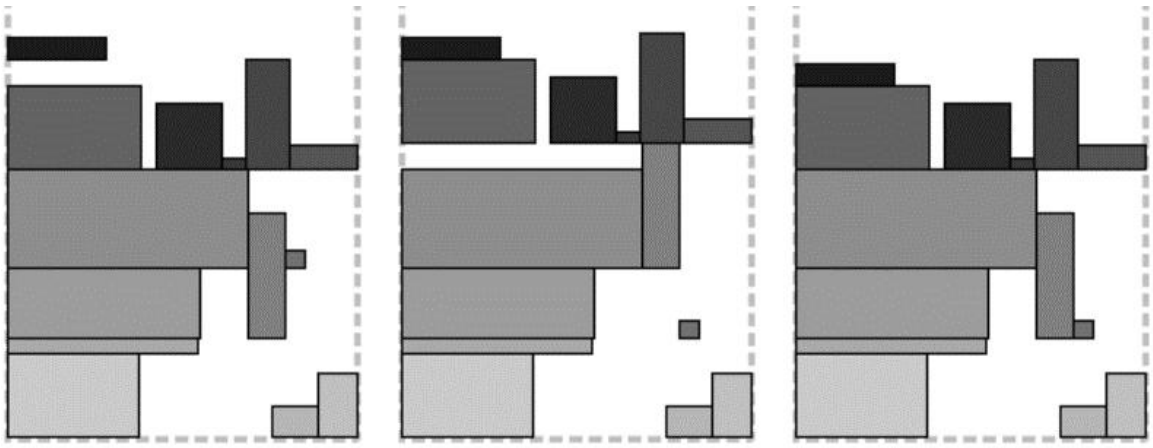

Рисунок 13. Иллюстация Compression algorithms

Глава 2. Проектирование и разработка программы

Заключение

Исследование и программная реализация задачи (комбинаторной оптимизации) позволили систематизировать и расширить теоретические знания в данной области и применить их на практике.

Работа с данной курсовой работы посредством решения задач позволила систематизировать знания о принципах и алгоритмах, применить их на практике.

Как пример практической работы с двумерной упаковкой была разработана программа с реализованными алгоритмами: FFDH, BFDH, FCNR, NFDH, также был разработан интерфейс, на котором наглядно показывается результат выбранных алгоритмов. По итоговым данным каждой реализации был произведен сравнительный анализ и сформирован вывод об эффективности каждого алгоритма.

При работе над данной курсовой работы были раскрыты основные понятия и алгоритмы двумерной упаковки.