Содержание

[1. Формальная поставновка задачи 2](#_Toc389503574)

[2. Примеры постановочных задач 4](#_Toc389503575)

[3. Схемы решения задач 7](#_Toc389503576)

[3.1 Конструктивные стратегии испольнения решения 7](#_Toc389503577)

[3.1.1 Стратегия минимизации пересечений 7](#_Toc389503578)

[3.1.2 Стратегия размещения компонентов интегральной цепи по критериям. 11](#_Toc389503579)

[3.2 Итерационные стратегии испольнения решения 15](#_Toc389503580)

[3.2.1 Force Directed алгоритм размещения 18](#_Toc389503581)

[3.3 Гибридные схемы генераций решений 20](#_Toc389503582)

[4. Аспекты программной реализации 20](#_Toc389503583)

[4.1 Требования 20](#_Toc389503584)

[4.2 Архитектура 20](#_Toc389503585)

[4.3 Сущность схем 20](#_Toc389503586)

[4.4 Реализация стратегий 20](#_Toc389503587)

[4.4.1 Программная реализация стратегии минимизации пересечений 20](#_Toc389503588)

[4.4.2 Программная реализация стратегии размещения компонентов по критериям 21](#_Toc389503589)

[4.5 Визуализация решений 23](#_Toc389503590)

[4.6 Компонент сбора статистических данных 23](#_Toc389503591)

[5. Тестовые задачи и тестирование 24](#_Toc389503592)

[5.1 Описание тестовых задач 24](#_Toc389503593)

[5.2 Система тестирования 24](#_Toc389503594)

[5.3 Программная реализация системы тестирования 25](#_Toc389503595)

[6. Вычислительный эксперимент 27](#_Toc389503596)

[6.1 Результат экспериментов 27](#_Toc389503597)

[6.2 Выводы 27](#_Toc389503598)

[7. Заключение 27](#_Toc389503599)

[8. Список использованных источников 27](#_Toc389503600)

[9. Приложения 27](#_Toc389503601)

# Формальная поставновка задачи

*x*

(0;0)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | (N;M) |

*y*

**Исходные данные:**

Дано пространство, известны координаты его верхнего левого угла (0;0) и нижнего правого (N;M).

- множество размещаемых элементов

- множество периферийных элементов.

- размер элемента *i* по оси *y;*

- размер элемента *i* по оси *x;*

 - координата верхнего левого угла элемента *j* по оси *x*

 - координата верхнего левого угла элемента *j* по оси *y*

- множество цепей, 

**Варьируемые параметры** – координаты верхнего левого угла размещаемых элементов

 - координата верхнего левого угла элемента *i* по оси *x*

 - координата верхнего левого угла элемента *i* по оси *y*

**Ограничения:**

Элементы не выходят за пределы пространства:

 -



Элементы не пересекаются:

 если  или ,  -

 если  или , 

**Критерий:**

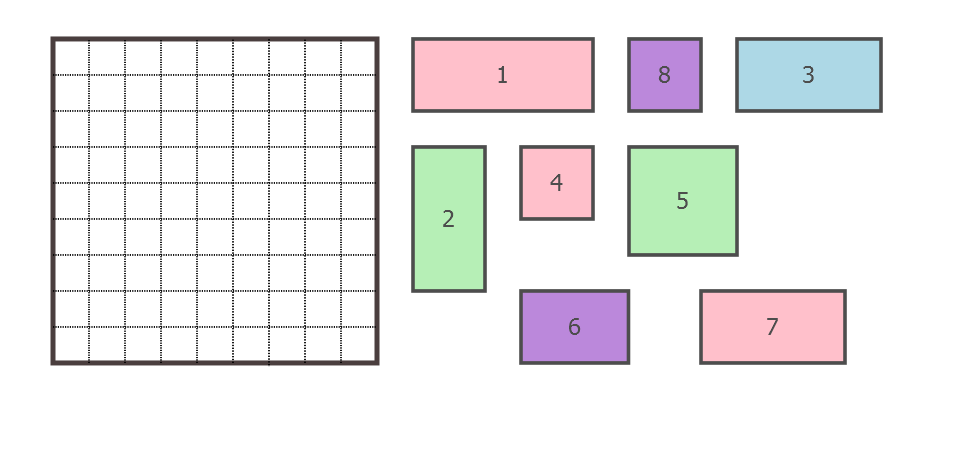
# Примеры постановочных задач

**Исходыне данные:**

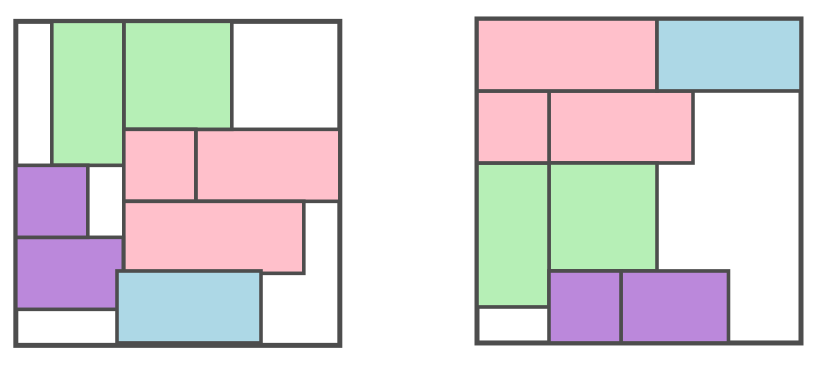
* Монтажное пространство 9х9
* Количество компонентов 8
* Количество цепей 3

|  |  |  |
| --- | --- | --- |
| **№ Элемента** | **Ширина** | **Высота** |
| 1 | 5 | 2 |
| 2 | 2 | 4 |
| 3 | 4 | 2 |
| 4 | 2 | 2 |
| 5 | 3 | 3 |
| 6 | 3 | 2 |
| 7 | 4 | 2 |
| 8 | 2 | 2 |

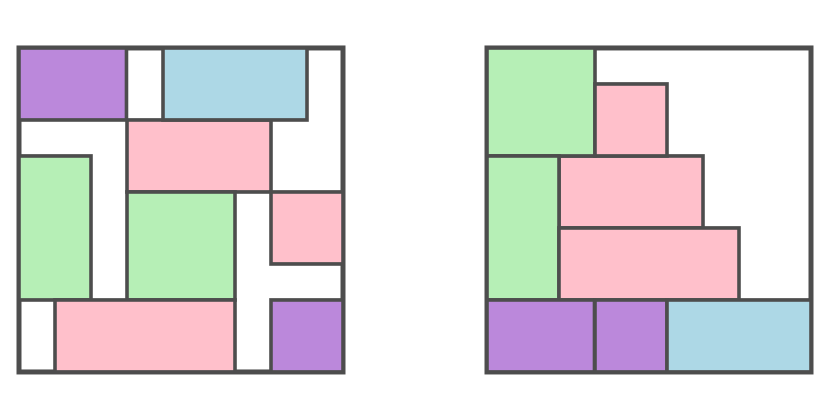
|  |  |  |
| --- | --- | --- |
| **№ Цепи** | **Элементы цепи** | **Цвет цепи** |
| 1 | 1,4,7 | розовый |
| 2 | 2,5 | зеленый |
| 3 | 6,8 | фиолетовый |



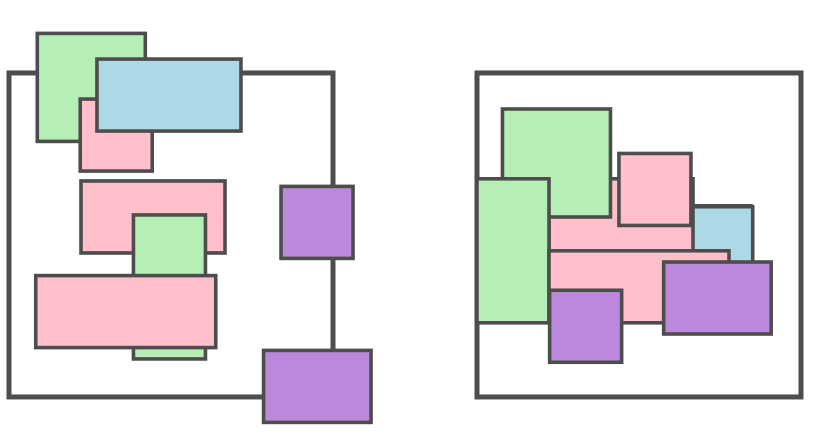
**Оптимальное решение:**

****

**Допустимое решение:**

****

**Недопустимое решение:**

****

# Схемы решения задач

# Конструктивные стратегии испольнения решения

# Стратегия минимизации пересечений

**Общая идея стратегии**

Путём построения сетки, проходящей через посадочные места, монтажное пространство разбивается на непересекающиеся области (регионы), которые представляют собой единичные квадраты, левые верхние углы которых совпадают с посадочными местами. Для каждой области рассчитывается коэффициент заполненности, который вычисляется как количество попадающих в неё компонент. Затем рассматривается множество элементов попадающих в регион с максимальным коэффициентом. Данное множество может содержать как ещё неразмещенные (незафиксированные), так и размещенные (зафиксированные) элементы. Если область не содержит последних, то один из элементов фиксируется. Незафиксированные компоненты последовательно размещаются и фиксируются вне рассматриваемой области, таким образом, чтобы их новые координаты были как можно ближе к исходным. При этом не допускаются пересечения с зафиксированными элементами. Процесс повторяется до тех пор, пока не будут размещены все элементы схемы.

**Алгоритм работы стратегии**

Параметры алгоритма:

– стратегия поиска посадочных мест элементов с учётом зафиксированных элементов

Обозначения:

номера элементов схемы

номера регионов, на которые разбивается пространство

– коэффициент заполненности региона с номером

множество рассмотренных областей

**Алгоритм**:

1. Разбить пространство на области

=∅

1. Если - ВЫХОД

Иначе перейти на шаг 3

1. Для каждой области ∉ посчитать коэффициент

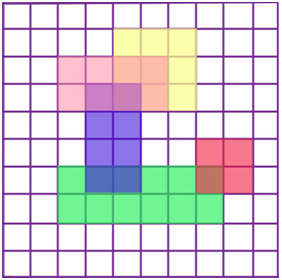
Если есть хотя бы одна область, для которой , перейти на шаг 4

Иначе - ВЫХОД

1. Выбрать область ∉ с максимальным значением
2. Если ∅ , выбрать из самый большой по площади элемент c и установить
3. Для каждого элемента ∈ повторить следующие шаги:
   1. Используя алгоритм найти допустимые позиции для элемента
   2. Среди позиций выбрать ту, при размещении в которой, текущий элемент будет иметь максимально близкое расположение к своим исходным координатам
4. Поместить в выбранную позицию, установить

Добавить в

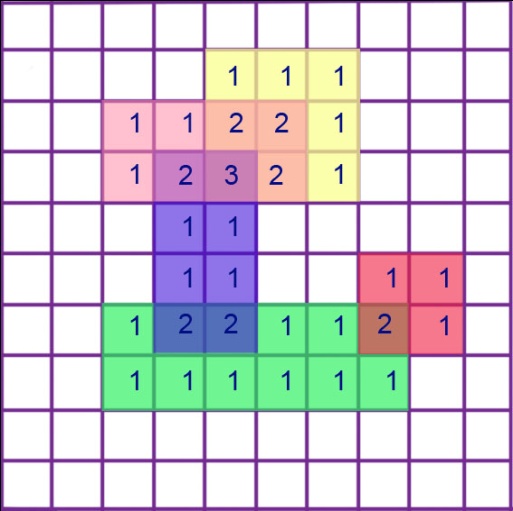
Перейти на шаг 2

**Пример работы стратегии**

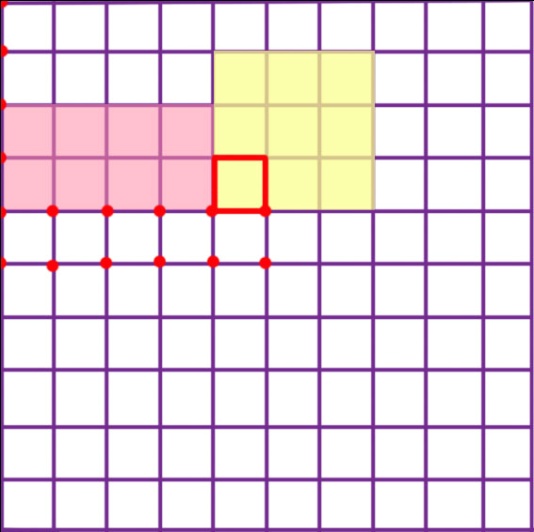
Исходные данные:

* Количество элементов 5
* Габариты элементов

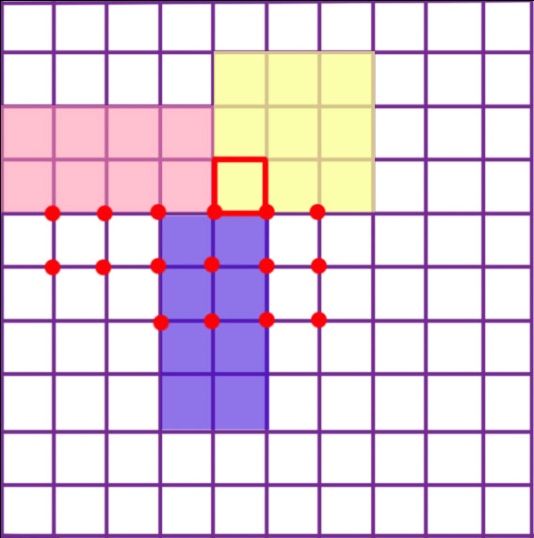
|  |  |  |
| --- | --- | --- |
| № Элемента | Длина | Ширина |
| 1 | 3 | 3 |
| 2 | 4 | 2 |
| 3 | 2 | 4 |
| 4 | 6 | 2 |
| 5 | 2 | 2 |

Шаг 1 (Выбор элементов для размещения)

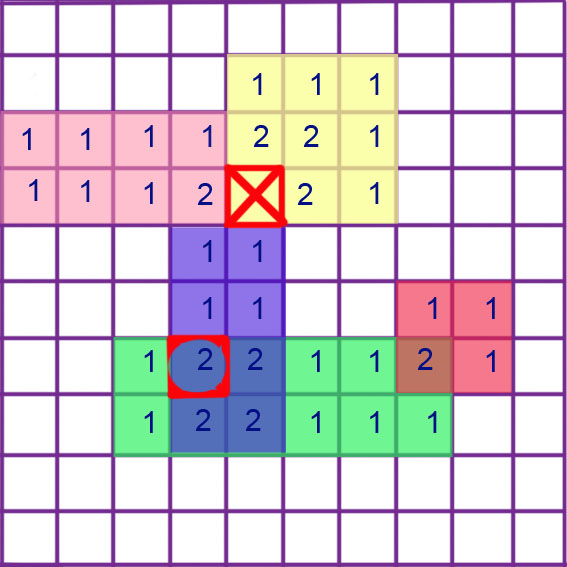
* Рассчитываем коэффициенты заполненности областей.
* Находим область с максимальным значением коэффициента
* Область не содержит зафиксированных элементов, следовательно, фиксируем самый большой по площади элемент – 1-ый.
* Необходимо разместить 2-ой и 3-ий элементы

Шаг 2 (Размещение 2-го элемента)

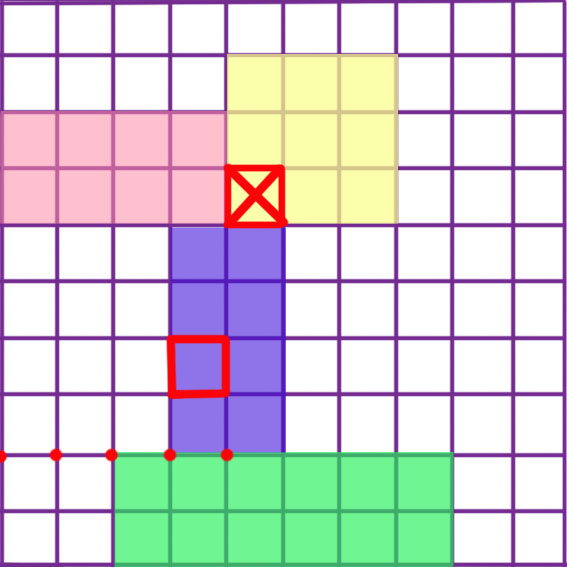
* Находим первые 16 позиции для размещения 2-го прямоугольника с учетом того, что зафиксирован 1-ый элемент
* Выбираем лучшую позицию.
* Фиксируем 2-ой элемент

Шаг 3 (Размещение 3-го элемента)

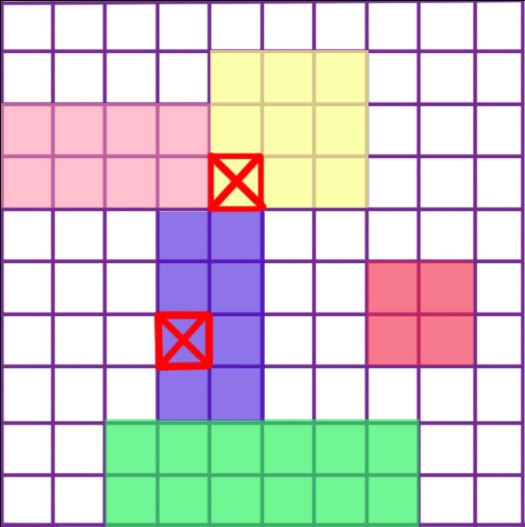
* Находим первые 16 позиции для размещения 3-го элемента
* Выбираем лучшую позицию
* Фиксируем синий 3-ий элемент

Шаг 4 (Выбор элементов для размещения)

* Помечаем область как рассмотренную
* Пересчитываем коэффициенты заполненности областей
* Выбираем область с максимальным значением.
* Внутри области уже зафиксированный элемент – 3ий.
* Необходимо разместить 4-ый элемент

Шаг 5 (Размещение 4-го элемента)

* Находим первые 16 позиции для размещения 4-го элемента
* Выбираем лучшую позицию. Фиксируем 4-ый элемент

Шаг 6 (Выбор элементов для размещения)

* Помечаем область как рассмотренную
* Пересчитываем коэффициенты заполненности областей
* Нет ни одной области с коэффициентом большим единицы

ВЫХОД

# Стратегия размещения компонентов интегральной цепи по критериям.

**Общая идея стратегии**

Первым шагом стратегии является построение сетки, которая представляет собой монтажное пространство. Это пространство разбивается на единичные квадраты (ячейки), которые не пересекаются между собой, у которых левые верхние углы совпадают с посадочными местами. Для каждой ячейки рассчитывается коэффициент заполненности, который равен количеству попадающих в нее компонент. Затем ищется максимальное значение среди всех коэффициентов. Среди множества компонентов, которые попяли в данную ячейку, могут быть размещенные(зафиксированные) и неразмещенные( незафиксированные) элементы. Если среди них, есть незафиксированные элементы, то выбирается компонент с максимальной площадью. Иначе продолжается поиск максимального коэффециента. После того, как был выбрали «лучший» элемент, выбирается наилучшая позиция в сетке. Рассматривается каждая позиция, для нее рассчитывается значения по трем критериям. Первый критерий, рассчитывает процент пересечения «лучшего» компонента с другими элементами. Второй, подсчитывает длину цепей, в который входит данный компонент по манхетовской метрике. Последний критерий, рассчитывает длину удаления от начальной позиции элемента. Каждая позиция сравнивается с предыдущей и выбирается наилучшая, путем минимизации значений критериев. Процесс повторяется , пока не разместятся все компоненты либо не будет коэффициентов , значение которых не будет превышать единицу. Если остались незафиксированные компоненты и при этом значение коэфициентов не превышает единицу, то они размещаются аналогичным путем.

**Алгоритм работы стратегии**

Параметры алгоритма:

– стратегия сравнения двух позици по критериям.

Обозначения:

номера элементов схемы

номера регионов, на которые разбивается пространство

– коэффициент заполненности региона с номером

B, C, D – критерии размещения элемента.

множество рассмотренных областей

Алгоритм:

1. Разбить пространство на сетку

=∅

1. Если - ВЫХОД

Иначе перейти на шаг 3

1. Для каждой области ∉ посчитать коэффициент

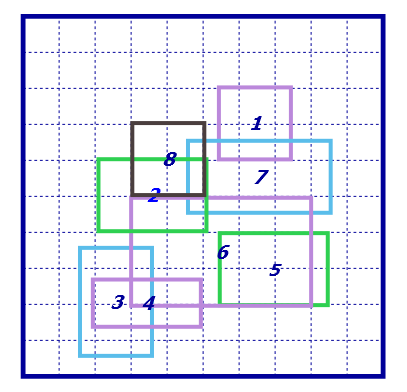
Если есть хотя бы одна область, для которой , перейти на шаг 4

Иначе - ВЫХОД

1. Выбрать область ∉ с максимальным значением
2. Если ∅ , выбрать из самый большой по площади элемент c и установить
3. Для элемента с выбрать наилучшую позицию:
   1. Использовать каждый критерий, для подсчета значений.
   2. По стратегии А выбрать наилучшую позицию.
4. Поместить в выбранную позицию, установить Добавить в . Перейти на шаг 2

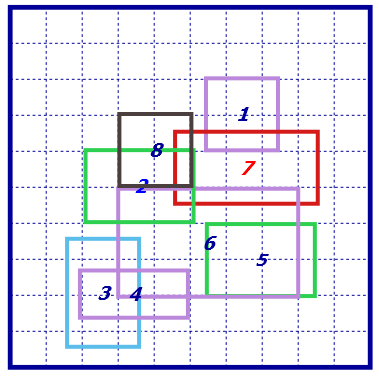
**Пример работы стратегии**

Исходные данные:

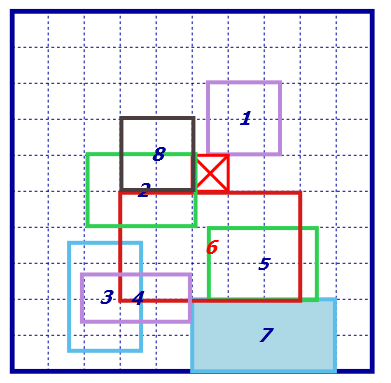
* Количество элементов ( цвет элемента соответствует цвету цепи)
* Количество цепей
* Габариты элементов

|  |  |  |
| --- | --- | --- |
| **№ Элемента** | **Длина** | **Ширина** |
| 1 | 2 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 2 |
| 4 | 1 | 3 |
| 5 | 2 | 3 |
| 6 | 3 | 5 |
| 7 | 2 | 4 |
| 8 | 2 | 2 |

|  |  |  |
| --- | --- | --- |
| **№ Цепи** | **Элементы цепи** | **Цвет цепи** |
| 1 | 1,4,6 | фиолетовый |
| 2 | 2,5 | зеленый |
| 3 | 3,7 | голубой |

Шаг 1 ( Размещение 1ого элемента.)

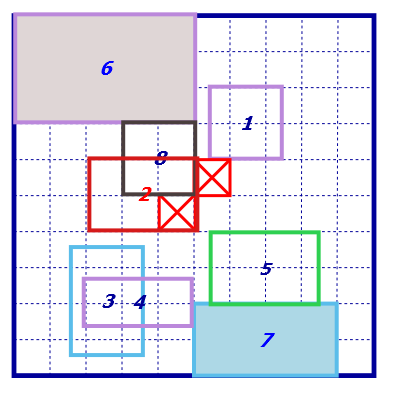
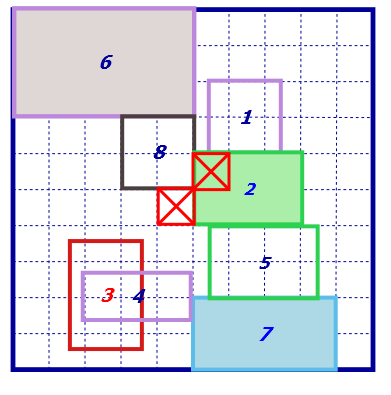
* Рассчитываем коэффициенты заполненности областей.
* Находим «тяжелую» позицию с максимальным значением коэффициента.
* Из набора элементов входящих в «тяжелую» позицию, выбирается компонент с максимальной площадью. Лучший элемент под № 7 .
* Элемент № 7 помечается, как зафискированный элемент.
* Ищем позицию для размещения лучшего элемента используя 3 критерия. Размещаем в лучшей позиции.
* Помечаем исходную позицию, как недоступную.

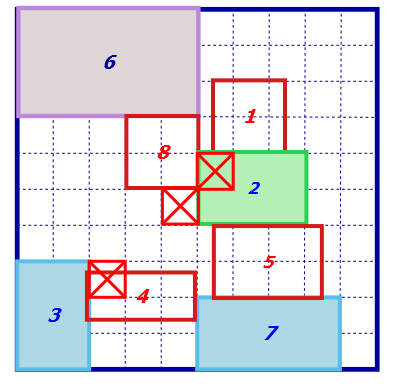


Шаг 2 ( Размещение 2ого элемента)

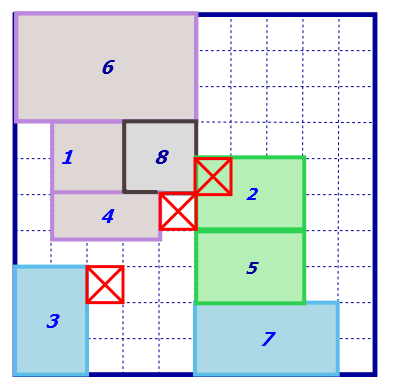
* Стратегия поиска «тяжелой» позиции аналогична.
* Лучший компонент под № 6.
* Элемент № 6 фиксируется.
* Выбор «лучшей» позиции. Размещение компоненты в выборанной позиции.
* Исходная позиция, помечается как недоступная.

Шаг 3 размещает 2-ой компоненты Шаг 4 размещает 3-ий компонент



Шаг 5 (Размещение нефиксированных элементов)

* «Тяжелых» ячеек в области больше нет.
* Размещаются все незафикисированные элементы по аналогичным стратегиям, которые представлены выше.



**Итоговое размещение:**

# Итерационные стратегии испольнения решения

Описание стратегий

**Набор эвристик**

Данная стратегия основа на 4 стратегиях итерационного алгоритма. Данными стратегиями являются:

1. Выбор очередного элемента размещения;
2. Поиск позиций для элемента;
3. Оценка позиции;
4. Сортировка позиции по оценке.

**Выбор очередного элемента размещения**

Рассмотрим выбор компонентов из общего набора. Упорядочим все цепи по частичному критерию. Под цепи частичному критерием будем понимать Манхэттенскую метрику цепи. При таком упорядочении мы выявляем наихудшие цепи с точки зрения вклада в общий критерий. Так как размер компонентов цепи может существенно влиять на эту оценку см. рисунок 1, введём дополнительную оценку как суммарную площадь всех компонентов цепи.

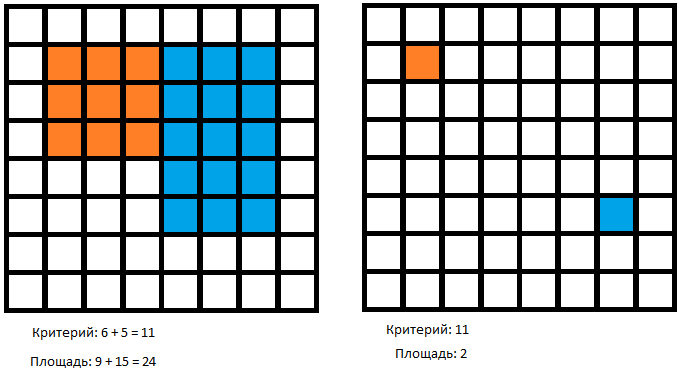


Рисунок 1 Критерий и площадь отдельных цепей. Пусть оба компонента каждого рисунка составляю цепь.

Такая оценка позволит выявить те компоненты размешенние которых может быть затруднено после размещения меньших см. рисунок 2.

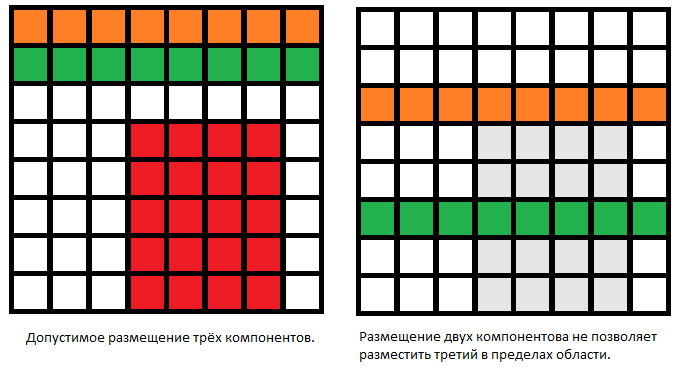


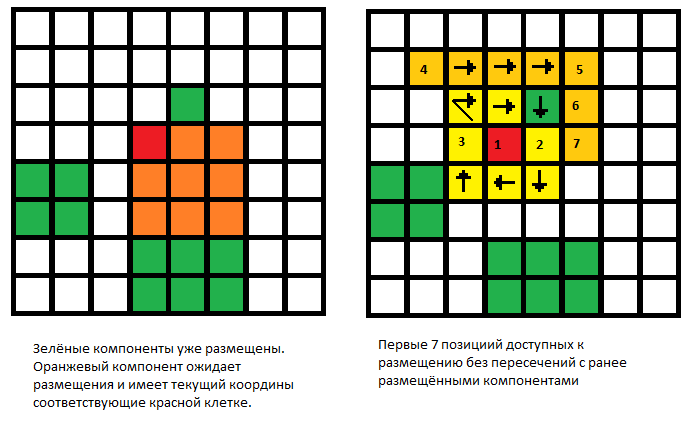
Рисунок 2 Размещение крупных компонентов в ограниченной области.

Таким образом получаем упорядоченный список цепей. Затем выбираем к размещению компонент первой из упорядоченных цепей имеющий наибольшую площадь.

**Поиск позиций для элемента**

Рассмотрим поиск позиций для размещения. Примем к рассмотрению текущий глобальные координаты размещаемого компонента

Так как другие компоненты уже могут быть размещены и иметь локальные целочисленные координаты конченного результата, то необходимо исключить те позиции которые допускают пересечение. Позиции будут рассмотрены по спирали от , до тех по пока не будет набрано необходимое число координат для дальнейшей оценки. Подробнее можно увидеть на рисунке 3.



**Рисунок 3 Поиск позиций по спирали. С учётом текущих глобальных координат и позиций размещённых компонентов.**

**Оценка позиции**

После нахождения необходимого числа позиций необходимо выбрать одну для конечного размещения. Данный выбор можно осуществить учитывая следующий список критериев:

1. Частичный критерий. То есть суммарную Манхэттенскую метрику цепей, в которые входит текущий компонент.
2. Пересечение с другими компонентами. Под другим тут рассматриваются только те компоненты которые ещё не имеют целочисленных конечных координат, а только глобальное приближение.

Таким образом будет выбрана позиция предоставляющая минимальный критерий и имеющая наименьшую площадь пересечений с далее размещаемыми копонентами.

Сортировка позиции по оценке

Сортировка позиций не входит в описание эвристики, так как входит предыдущий пункт.

# Force Directed алгоритм размещения

Данный алгоритм так же является итерационным, где одну итерацию модно описать следующим образом:

1. Выделение группы рассматриваемых компонентов (в общем случае);
2. Выбор направлений возможных перемещений каждого компонента;
3. Оценка каждого направления с учётом силы (длины) сдвига по некоторым критериям;
4. Сдвиг компонента в наилучшую позицию.

**Выделение группы рассматриваемых компонентов**

Под группой рассматриваемых компонент будем рассматривать все неразмещённые компоненты.

**Выбор направлений возможных перемещений**

Под направлениями сдвига будем рассматривать 8 направлений как показано на рисунке 4.

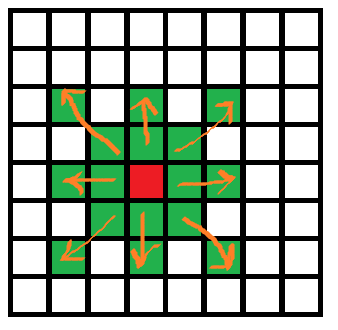


Рисунок 4 Направления возможных сдвигов компонента.

Оценка направления сдвига по некоторым критериям

Для оценки будем использовать два критерия. Первым является частичный суммарный критерий для текущего компонента, аналогично рассмотренному выше алгоритму. Под вторым будем понимать площадь пересечения с другими компонентами. Подробнее можно видеть на рисунке 5.

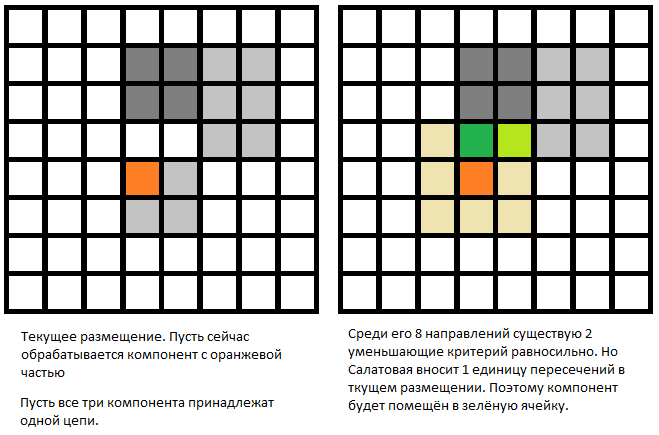


Рисунок 5 Оценка и выбор направления по критериям.

**Сдвиг компонента в наилучшую позицию.**

После выбора лучшей позиции для компонента она становится его текущей позицией и происходит переход к следующему компоненту.

# Гибридные схемы генераций решений

Альтернативные стратегии исполнения решения представляют собой комбинации различных стратегий сортировки компонент, поиска позиций, сортировки позиций и сравнения позиций.

Примеры альтернативных стратегий:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Сортировка компонент** | **Поиск позиций** | **Сортировка позиций** | **Сравнение позиций** |
| 1 | Сортировка с использованием раскраски графа | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 2 | Сортировка с использованием раскраски графа | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 3 | Сортировка с использованием раскраски графа | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |
| 4 | Сортировка с использованием раскраски графа | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 5 | Сортировка с использованием раскраски графа | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 6 | Сортировка с использованием раскраски графа | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |
| 7 | Сортировка с использованием раскраски графа | Линейный перебор | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 8 | Сортировка с использованием раскраски графа | Линейный перебор | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 9 | Сортировка с использованием раскраски графа | Линейный перебор | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |
| 10 | Частичный критерий | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 11 | Частичный критерий | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 12 | Частичный критерий | Поиск позиций по спирали от текущих координат | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |
| 13 | Частичный критерий | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 14 | Частичный критерий | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 15 | Частичный критерий | Позиции по спирали. C выходом за границы | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |
| 16 | Частичный критерий | Линейный перебор | Пузырьковая сортировка | Сравнение с использованием частичного критерия и минимизацией пересечений |
| 17 | Частичный критерий | Линейный перебор | Пузырьковая сортировка | Сравнение с использованием частичного критерия |
| 18 | Частичный критерий | Линейный перебор | Пузырьковая сортировка | Жадный выбор позиций со строго меньшими или строго большими координатами |

# Аспекты программной реализации

# Требования

Предоставить реализацию методов для выполнения размещения компонент схемы.

    /// <summary>

    /// Интерфейс алгоритмов глобального размещения компонентов интегральной схемы

    /// </summary>

    public interface IGlobalPlacer

    {

        /// <summary>

        /// Осуществляет глобальное размещение компонентов интегральной схемы

        /// </summary>

        /// <param name="design">Описание интегральной схемы</param>

        /// <param name="result">Результат - вектор глобального размещения (действительные координаты)</param>

        void Place(Design design, PlacementGlobal result);

    }

    /// <summary>

    /// Интерфейс алгоритмов детального размещения компонентов интегральной схемы

    /// </summary>

    public interface IDetailPlacer

    {

        /// <summary>

        /// Осуществляет детальное размещение компонентов интегральной схемы

        /// </summary>

        /// <param name="design">Описание интегральной схемы</param>

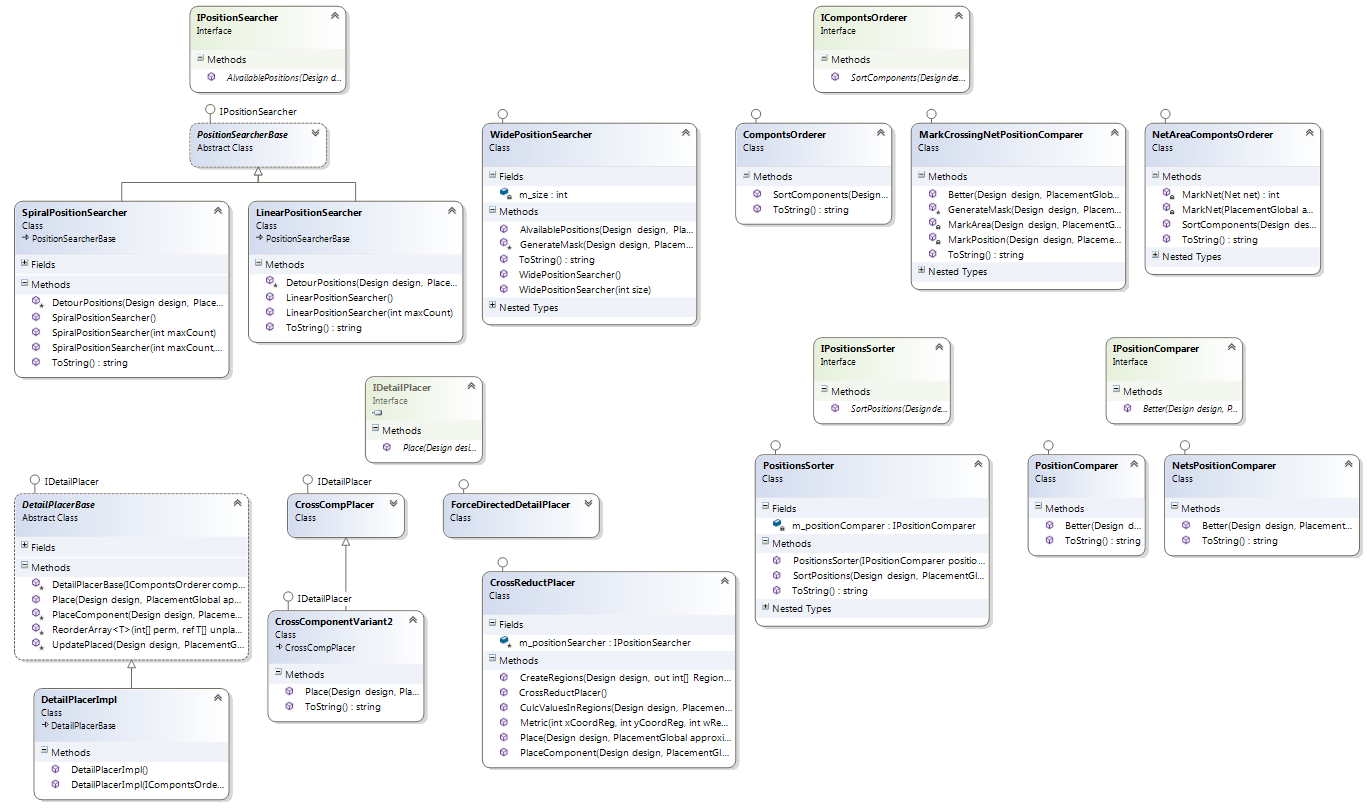
        /// <param name="approximate">Некоторое начальное размещения (действительные координаты)</param>

        /// <param name="result">Результат - вектор детального размещения (целочисленные координаты)</param>

        void Place(Design design, PlacementGlobal approximate, out PlacementDetail result);

    }

# Архитектура



# Сущность схем

# Реализация стратегий

# Программная реализация стратегии минимизации пересечений

**Контракты основных функций**

/// <summary>

/// Основная функция размещения элементов

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="approximate"></param> глобальное размещение

/// <param name="result"></param> полученное размещение размещение

public void Place(Design design, PlacementGlobal approximate, out PlacementDetail result)

/// <summary>

/// Функция размещения текущего элемента

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="approximate"></param> глобальное размещение

/// <param name="current"></param> текущий размещаемый элемент

/// <param name="result"></param> текущее размещение

/// <param name="placed"></param> признак того, что элемент размещен

/// <param name="xCoordReg"></param> коррдината x левого верхнего угла региона

/// <param name="yCoordReg"></param> координата y левого верхнего угла региона

/// <param name="wRegs"></param> ширина региона

/// <param name="hRegs"></param> высота региона

public virtual void PlaceComponent(Design design, PlacementGlobal approximate, Component current, PlacementDetail result, out bool placed, int xCoordReg, int yCoordReg, int wRegs, int hRegs)

/// <summary>

/// Функция разбиения области на регионы

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="RegionsValues"></param> массив для рассчета коэфициентов заполнености регионов

/// <param name="xCoordRegions"></param> x коррдинаты левых верхних углов регионов

/// <param name="yCoordRegions"></param> y координаты левых верхних углов регионов

/// <param name="wRegs"></param> ширина регионов

/// <param name="hRegs"></param> длина регионов

/// <param name="regsCount"></param> количество регионов

public virtual void CreateRegions(Design design, out int[] RegionsValues, out int[] xCoordRegions, out int[] yCoordRegions, out int wRegs, out int hRegs, out int regsCount)

/// <summary>

/// Функция рассчета коэффициентов заполненнсти регионов

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="approximate"></param> глобальное размещение

/// <param name="result"></param> локальное(текущее) размещение

/// <param name="compInRegions"></param> списки элементов попадающих в каждый регион

/// <param name="RegionsValues"></param> значения коэффициентов заполненности регионов

/// <param name="xCoordRegions"></param> x коррдинаты левых верхних углов регионов

/// <param name="yCoordRegions"></param> y координаты левых верхних углов регионов

/// <param name="wRegs"></param> ширина регионов

/// <param name="hRegs"></param> длина регионов

/// <param name="regsCount"></param>

public void CulcValuesInRegions(Design design, PlacementGlobal approximate, PlacementDetail result, out List<int>[] compInRegions, int[] RegionsValues, int[] xCoordRegions,int[] yCoordRegions, int wRegs, int hRegs, int regsCount)

# Программная реализация стратегии размещения компонентов по критериям

**Контракты основных функций**

/// <summary>

/// Основная функция размещения элементов

/// </summary>

/// <param name="design"></param>элементы схемы

/// <param name="approximate"></param>глобальное размещение

/// <param name="result"></param>полученное размещение размещение

public virtual void Place(Design design, PlacementGlobal approximate, out PlacementDetail result)

/// <summary>

/// Выбор позиции для размещения текущего компонента

/// </summary>

/// <param name="xCellCoord"></param> координата х левого верхнего угла ячейки

/// <param name="yCellCoord"></param> координата х левого верхнего угла ячейки

/// <param name="ValueCell"></param> массив коэфициентов

/// <param name="qtcells"></param> количество ячеек

/// <param name="Current"></param> текущий компонент

/// <param name="design"></param> элементы схемы

/// <param name="myApproximate"></param> глобальное размещение

/// <returns></returns>

public int GetBestCell(int[] хCellCoord, int[] yCellCoord, int[] ValueCell, int qtcells, int indexcurrent, Component Current, Design design, PlacementGlobal myApproximate)

/// <summary>

/// Функция подсчета процента пересечения текущей компоненты с элементами схемы

/// </summary>

/// <param name="xCellCoord "></param> координата х левого верхнего угла ячейки

/// <param name=" yCellCoord "></param> координата y левого верхнего угла ячейки

/// <param name="Current"></param> текущий компонент

/// <param name="design"></param> элементы схемы

/// <param name="myApproximate"></param> глобальное размещение

/// <returns></returns>

public double GetPercentCross(int xCellCoord, int yCellCoord, Component Current, Design design, PlacementGlobal myApproximate)

/// <summary>

/// Функция расчета длины от текущей ячейки до начальной позиции компонента

/// </summary>

/// <param name="myApproximate"></param> глобальное размещение

/// <param name="current"></param> текущий компонент

/// <param name="xcurrCoord"></param> координата х левого верхнего угла текущей ячейки

/// <param name="ycurrCoord"></param> координата y левого верхнего угла текущей ячейки

/// <returns></returns>

public double GetCloselyCell(PlacementGlobal myApproximate, Component current, int xcurrCoord, int ycurrCoord)

/// <summary>

/// Функция расчета длины цепей, в которые входит текущая компонента

/// </summary>

/// <param name="Current"></param> текущий компонент

/// <param name="myApproximate"></param> глобальное размещение

/// <param name="design"></param> элементы схемы

/// <param name="xCellCoord"></param> координата х левого верхнего угла ячейки

/// <param name="yCellCoord"></param> координата y левого верхнего угла ячейки

/// <returns></returns>

public double GetNearNet(Component Current, PlacementGlobal myApproximate, Design design, int xCellCoord, int yCellCoord)

/// <summary>

/// Функция разбиения пространства на сетку

/// </summary>

/// <param name="width"></param> длина пространства

/// <param name="height"></param> длина пространства

/// <param name="design"></param> элементы схемы

/// <param name="xCellCoord"></param> координата х левого верхнего угла ячейки

/// <param name="yCellCoord"></param> координата y левого верхнего угла ячейки

/// <param name="qtcells"></param> количество ячеек

public void CreateCells(int width, int height, Design design, out int[] xCellCoord, out int[] yCellCoord, int qtcells)

/// <summary>

/// Функция рассчета коэфициентов заполнености для каждой ячейки

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="myApproximate"></param> глобальное размещение

/// <param name="xCellCoord"></param> координата х левого верхнего угла ячейки

/// <param name="yCellCoord"></param> координата y левого верхнего угла ячейки

/// <param name="qtcells"></param> количество ячеек

/// <param name="ValueCell"></param> массив коэфициентов

/// <param name="h"></param> длина пространства

/// <param name="w"></param> ширина пространства

/// <param name="CompInCell"></param> список ячеек и список компонентов которые входят в ячейку

public void FillCells(Design design, PlacementGlobal myApproximate, int[] xCellCoord, int[] yCellCoord, int qtcells, int[] ValueCell, int h, int w, List<List<Component>> CompInCell)

/// <summary>

/// Очищение ячеек сетки

/// </summary>

/// <param name="qtcells"></param> количество ячеек

/// <param name="ValueCell"></param> массив коэфициентов

/// <param name="CompInCell"></param> список ячеек и список компонентов которые входят в ячейку

public void Clear(int qtcells, int[] ValueCell, List<List<Component>> CompInCell)

/// <summary>

/// Копирование глобального размещения

/// </summary>

/// <param name="approximate"></param> глобальное размещение

/// <param name="design"></param> элементы схемы

/// <returns></returns>

public PlacementGlobal CreateMyApproximate(PlacementGlobal approximate, Design design)

/// <summary>

/// Выбор из набора компонентов, элемента по максимально площади

/// </summary>

/// <param name="cell"></param> массив компонентов

/// <param name="fixedComponents"></param> массив зафиксированных компонентов

/// <returns></returns>

public Component GetComponentWithMaxSquare(List<Component> cell, List<Component> fixedComponents)

# Визуализация решений

**Требования к реализации.**

1. Визуализация компонентов интегральной микросхемы согласно укладке (не размещенные элементы недолжны отображаться).
2. Элементы должны отрисовываться серым цветом с красными рамками.
3. Левый верхний угол каждого элемента должен выделяться с целью удобного определения позиционирования элемента
4. Отрисовка сетевых линий, соединяющих элементы цепей, точки соединения должны отмечаться.

**Реализовано:**

1. Система визуализации элементов интегральной микросхемы с заявленными свойствами.
2. Система визуализации компонентов и сетей, согласно требованиям.

**Контракты основных функций:**

**DrawImpl.cs**

/// <summary>

/// Функция отрисовки размещения с целыми координатыми (без сетей)

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="placement"></param> глобальное размещение

/// <param name="size"></param> размер участка для отрисовки

/// <param name="canvas"></param> хэндл визуализации

public override void Draw(PlaceModel.Design design, PlacementDetail placement, Size size, System.Drawing.Graphics canvas)

/// <summary>

/// Функция отрисовки глобального размещения с вещественными координатыми (без сетей)

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="placement"></param> глобальное размещение

/// <param name="size"></param> размер участка для отрисовки

/// <param name="canvas"></param> хэндл визуализации

public override void Draw(PlaceModel.Design design, PlacementGlobal placement, Size size, System.Drawing.Graphics canvas)

**DrawImplNets.cs**

/// <summary>

/// Функция отрисовки размещения с целыми координатыми с сетями

/// </summary>

/// <param name="design"></param> элементы схемы

/// <param name="placement"></param> глобальное размещение

/// <param name="size"></param> размер участка для отрисовки

/// <param name="canvas"></param> хэндл визуализации

///

public override void Draw(PlaceModel.Design design, PlacementDetail placement, Size size,System.Drawing.Graphics canvas)

/// <summary>

/// Функция отрисовки глобального размещения с вещественными координатыми с сетями

/// </summary>

/// <param name="design"></param> элементы схемы

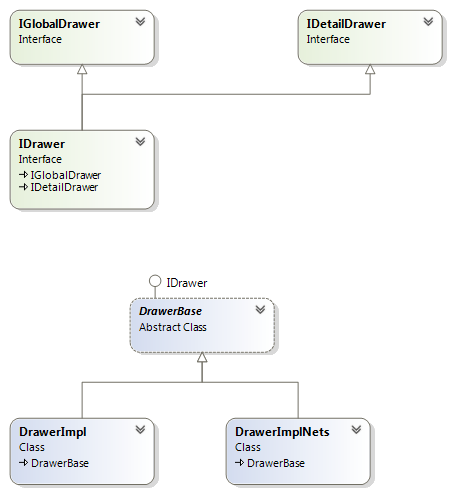
/// <param name="placement"></param> глобальное размещение

/// <param name="size"></param> размер участка для отрисовки

/// <param name="canvas"></param> хэндл визуализации

public override void Draw(PlaceModel.Design design, PlacementGlobal placement, Size size, System.Drawing.Graphics canvas)

**Диаграмма классов:**



# Компонент сбора статистических данных

**Требования к реализации**

1. Реализовать компонент сбора статистики по проведенным испытаниям с выводом статистики в форме html таблицы
2. Сбор статистики о общем числе элементов, числе размещенных элементов
3. Левый верхний угол каждого элемента должен выделяться с целью удобного определения позиционирования элемента
4. Отрисовка сетевых линий, соединяющих элементы цепей, точки соединения должны отмечаться.

Реализовано:

1. Система визуализации элементов интегральной микросхемы с заявленными свойствами.
2. Система визуализации компонентов и сетей, согласно требованиям.

**Контракты основных функций:**

/// <summary>

/// Функция инициализирующая парсер

/// </summary>

/// <param name="filepath"></param> путь к файлам статистики

## void Parce(string filepath = "");

/// <summary>

/// Функция парсинга файлов статистики и упаковки данных в объект

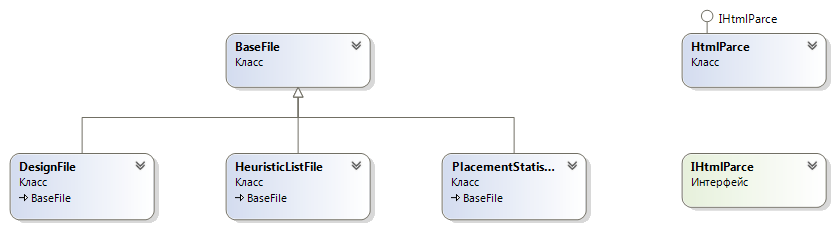
/// </summary>

/// <param name="fileInfo"></param>

/// <param name="file"></param>

private void ParceFileRegEx(FileInfo fileInfo, out BaseFile file)

Диаграмма классов:

****

# Тестовые задачи и тестирование

# Описание тестовых задач

Для проведения экспериментов использовались следующий набор сгенерированных задач.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Количество компонент | Максимальная ширина и высота | Процент заполнения | Число цепей | Максимальная длина цепи |
| Задача 1 | 30 | 4x4 | 50 | 15 | 5 |
| Задача 2 | 30 | 4x4 | 75 | 15 | 5 |
| Задача 3 | 30 | 4x4 | 90 | 15 | 5 |
| Задача 4 | 30 | 4x4 | 50 | 30 | 5 |
| Задача 5 | 30 | 4x4 | 75 | 30 | 5 |
| Задача 6 | 30 | 4x4 | 90 | 30 | 5 |

# Система тестирования

**Описание системы**

Система тестирования представляет собой консольное приложение, работающее с директорией, которая содержит тестовые данные. Через аргументы командной строки приложению передается путь к данной директории, а также путь к директории для сохранения результатов. Приложение собирает и сохраняет статистику по всем имеющимся тестовым данным, а так же их визуальное представление в виде PNG изображений. Затем приложение запускает на всех тестовых данных все возможные стратегии размещения, включая альтернативные, после чего сохраняет статистику полученных размещений и их визуальное представление.

**Технология запуска тестов и расчета метрик**

1. Чтение тестовых данных из директории
2. Сбор и сохранение статистики по всем тестовым данным
3. Формирование набора стратегий поиска позиций
4. Формирование набора стратегий сравнения позиций
5. Формирование набора стратегий сортировки позиций
6. Формирование набора стратегий сортировки элементов схемы
7. Повтор следующих действий для всех стратегий размещения элементов, полученных путем формирования всевозможных комбинаций стратегий поиска, сравнения, сортировки позиций и сортировки элементов.
   1. Запуск стратегии на всех тестовых данных
   2. Сохранение статистики полученных размещений и их визуальных представлений
8. Формирование набора альтернативных стратегий размещения
9. Повтор следующих действий для всех альтернативных стратегий размещения
   1. Запуск стратегии на всех тестовых данных
   2. Сохранение статистики полученных размещений и их визуальных представлений

# Программная реализация системы тестирования

**Контракты основных функций**

/// <summary>

/// Основная функция запуска тестов

/// </summary>

/// <param name="args"></param> аргументы командной строки

private static void Main(string[] args)

/// <summary>

/// Функция Загрузки массивов исходных данных. Инициализирует размеры и

/// картинки для дальнейших тестов

/// </summary>

/// <param name="design"></param> массив входных данных

/// <param name="approximate"></param> массив размещений входных данных

/// <param name="sizes"></param> массив с размерами изображений

/// <param name="bitmaps"></param> массив изображений

/// <returns></returns>

private static void LoadFromDirectory(string path, out Design[] design, out PlacementGlobal[] approximate,out Size[] sizes, out Bitmap[] bitmaps)

/// <summary>

/// Функция чтения тествых данных

/// </summary>

/// <param name="args"></param> аргументы командной строки , путь к директории с данными

/// <param name="design"></param> массив входных данных

/// <param name="approximate"></param> массив размещений входных данных

/// <param name="sizes"></param> массив с размерами изображений

/// <param name="bitmaps"></param> массив изображений

/// <returns></returns>

private static bool ReadInput(string[] args, out Design[] design, out PlacementGlobal[] approximate,out Size[] sizes, out Bitmap[] bitmaps)

/// <summary>

/// Функция сохранения информации о проведённом тесте

/// </summary>

/// <param name="path"></param> путь к директории для сохранения информации

/// <param name="testNum"></param> номер теста

/// <param name="compOrder"></param> стратегия сортировки элементов

/// <param name="posComparer"></param> стратегия сравнения позиций

/// <param name="posSearcher"></param> стратения поиска позиций

/// <param name="posSorter"></param> стратегия сортировки позиций

private static void SaveTestInfo(string path, int testNum, object compOrder, object posComparer, object posSearcher, object posSorter)

/// <summary>

/// Функция сохранения статистики входных данных

/// </summary>

/// <param name="path"></param> путь к директории для сохранения статистики

/// <param name="designNum"></param> номер тестовых данных

/// <param name="designStatistic"></param> статистика входных данных

private static void SaveDesignsInfo(string path, int designNum, IStatisticResult<double> designStatistic)

/// <summary>

/// Функция сохранения результатов те

/// </summary>

/// <param name="path"></param> путь к директории для сохранения результата

/// <param name="designNum"></param> номер тестовых данных

/// <param name="testNum"></param> номер теста

/// <param name="design"></param> тестовые данные

/// <param name="resultPlacement"></param> полученное размещение

/// <param name="placementStatistic"></param> статистика размещения

/// <param name="size"></param> размер изображения

/// <param name="bitmap"></param> изображение размещения

private static void SaveTestResults(string path, int designNum, int testNum, Design design,PlacementDetail resultPlacement, IStatisticResult<double> placementStatistic, Size size, Bitmap bitmap)

# Вычислительный эксперименты

Для всех экспериментов все компоненты в исходной задачи размещены вблизи центра области с пересечениями.

Используемые эвристики при проведении экспериментов.

|  |  |  |
| --- | --- | --- |
| Номер алгоритма | Название | Описание |
| 1 | Набор эвристик | Ссылки на соответствующие разделы |
| 2 | CrossCompPlacer |  |
| 3 | CrossComponent вариант 2 |  |
| 4 | CrossReductPlacer |  |
| 5 | ForceDirectedDetailPlacer |  |
| 6 | ForceDirectedDetailPlacer + CrossReductPlacer |  |

# Результаты экспериментов

Во всех проведённых экспериментах все компоненты размещены.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Задача | Начальный критерий | Эвристики, значение критерия | | | | | | Лучший результат (номер эвристики) |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Задача 1 | 123 | 119 | 95 | 94 | 152 | 64 | 92 | 92 (6) |
| Задача 2 | 119 | 114 | 110 | 106 | 162 | 62 | 118 | 106 (3) |
| Задача 3 | 108 | 119 | 127 | 126 | 132 | 60 | 114 | 114 (6) |
| Задача 4 | 210 | 244 | 231 | 228 | 271 | 140 | 199 | 199 (6) |
| Задача 5 | 167 | 187 | 184 | 177 | 198 | 115 | 176 | 176 (6) |
| Задача 6 | 221 | 222 | 226 | 234 | 239 | 132 | 218 | 218 (6) |

Далее представлены наглядные иллюстрации проведённых экспериментов для первой задачи. Результаты на других задачах имеют схожую структуру.

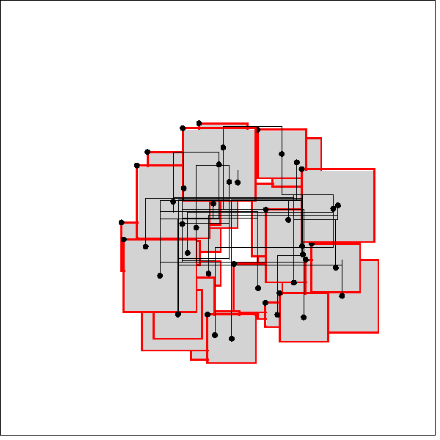


Рисунок 6 Начальное размещение

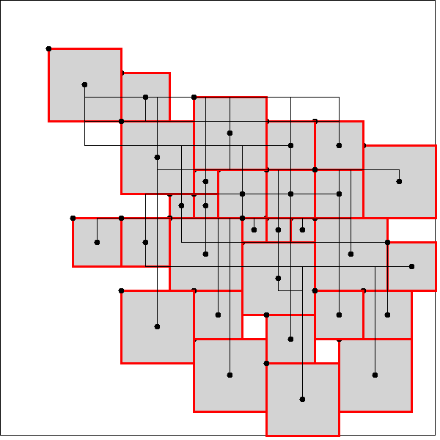


Рисунок 7 Эвристика 1

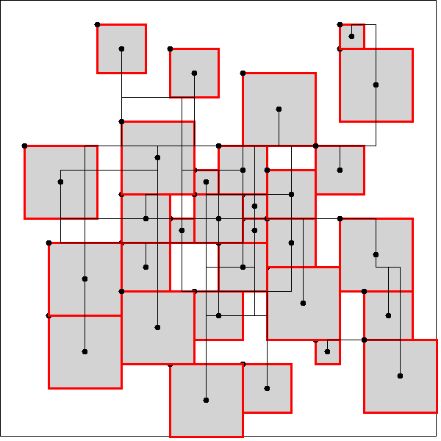


Рисунок 8 Эвристика 2

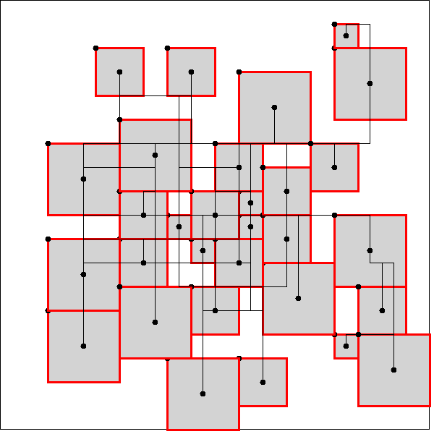


Рисунок 9 Эвристика 3

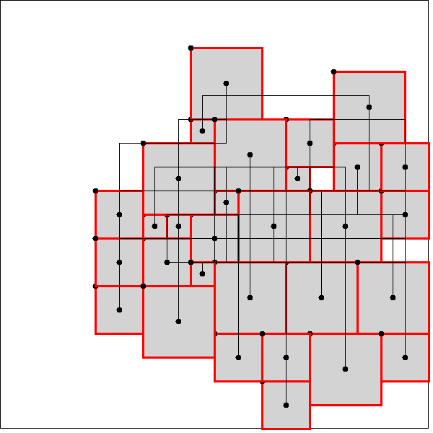


Рисунок 10 Эвристика 4

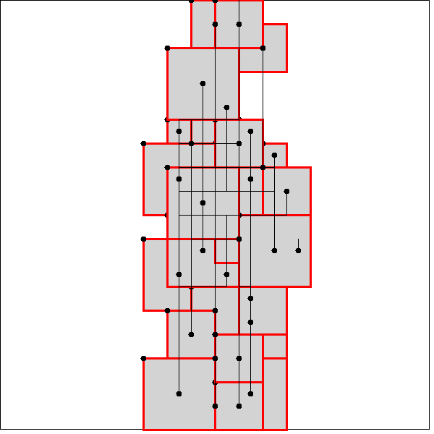


Рисунок 11Эвристика 5

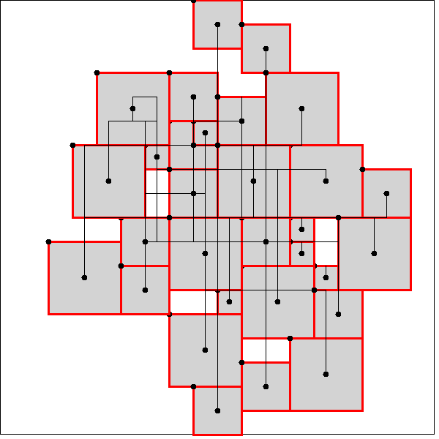


Рисунок 12 Эвристика 6

Нами были рассмотрены задачи из бенчмарков IBM. Перечень задач можно видеть в таблице 1. Данные задачи обладают достаточно большим процентом заполнения и разнообразием размеров компонент и цепей.

Таблица 1 Задачи из IBM бенчмарка

|  |  |  |
| --- | --- | --- |
| Название задачи | Количество элементов | Количество закреплённых элементов |
| ibm01 | 12752 | 246 |
| ibm02 |  |  |
| …. |  |  |
| Ibm18 |  |  |

Нами было подробно исследовано поведение алгоритмов на задаче ibm01 результаты можно видеть в таблице 2.

Алгоритм 1. Net and area sorter  
Search position by spiral  
Bubble sort with Criteria and minimize crossing

Алгоритм 2 CrossReductPlacer

Алгоритм 3 CrossComponent Variant 2

Запуск 1Предразмещённые алгоритмом силовой укладки графа

Запуск 2 Детальное размещение алгоритмом силовой укладки графа 10 итераций

Запуск 3 Детальное размещение алгоритмом силовой укладки графа 50 итераций

Запуск 4 Детальное размещение алгоритмом силовой укладки графа 100 итераций

Таблица 2 Исследование на первом бенчмарке

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Алгоритм | Процент размещения | Манхэттенская метрика | | Количесство пересечений | | Суммарная площадь пресечений | |
|  |  | начального размещения | детального размещения | начального размещения | детального размещения | начального размещения | детального размещения |
| 1.1 | 100% | 4679579,87287802 | 6305570 | 143528 | 74 | 22346624,5678548 | 74 |
| 1.2 | 100% | 3458566 | 5218640 | 44897 | 98 | 9720390 | 98 |
| 1.3 | 100% | 4538000 | 6136339 | 98652 | 104 | 17113172 | 104 |
| 1.4 | 100% | 4028679 | 5777737 | 51904 | 98 | 10796554 | 98 |
| 2.1 | 100% | 4679579,87287802 | 6010242 | 143528 | 0 | 22346624,5678548 | 0 |
| 2.2 | 100% | 3458566 | 5473963 | 44897 | 0 | 9720390 | 0 |
| 2.3 | 100% | 4538000 | 6105173 | 98652 | 0 | 17113172 | 0 |
| 2.4 | 100% | 4028679 | 5788507 | 51904 | 0 | 10796554 | 0 |
| 3.1 | 100% | 4679579,87287802 | 6059317 | 143528 | 0 | 22346624,5678548 | 0 |
| 3.2 |  | 3458566 |  | 44897 |  | 9720390 |  |
| 3.3 |  | 4538000 |  | 98652 |  | 17113172 |  |
| 3.4 |  | 4028679 |  | 51904 |  | 10796554 |  |

Ниже на рисукнках 13 - 21 представлены диаграммы удалённости компонент от начального, приближённого решения в зависимости от площади элеметов.

Рисунок 13 Алгоритм 1, запуск 1

Рисунок 14 Алгоритм 1 запуск 2

Рисунок 15 Алгоритм 1 запуск 3

Рисунок 16 Алгоритм 1 запуск 4

Рисунок 17 Алгоритм 2 запуск 1

Рисунок 18 Алгоритм 2 запуск 2

Рисунок 19 Алгоритм 2 запуск 3

Рисунок 20 Алгоритм 2 запуск 4

Рисунок 21 Алгоритм 3 запуск 1

# Выводы

# Заключение

# Список использованных источников

# Приложения