**Алгоритмы компоновки**

**Последовательный алгоритм компоновки по гиперграфу**

На каждом шаге выбирается вершина (элемент схемы) и помещается в первый кусок. Процесс заканчивается, когда все элементы распределены либо становится невозможным выполнение ограничений. Аналогично формируются следующие куски.

Опишем используемый алгоритм, при чём обозначим:

Eir – множество распределённых элементов схемы для узла r на шаге i;

Xir – множество нераспределённых элементов схемы для узла r на шаге i.

В начале из X1r выбирается вершина xj, такая что значение L1(xj) является максимальным. Элемент xj называется базовым и помещается в узел. Если таких элементов несколько, выбирается первый по порядку.

Далее из множества X1r удаляется xj и получается множество X2r. Среди этого множества находим вершины xt и вычисляем L2(xj , xt). Для тех xt, для которых L2(xj , xt) подчиняется ограничению на количество проводов (m), вычисляем L3(xj , xt). По итогу в узел выбираем элемент с максимальным значением L3(xj , xt), и, если таких элементов несколько, с минимальным значением L2(xt). Формулы для L1, L2, L3 приведены ниже:

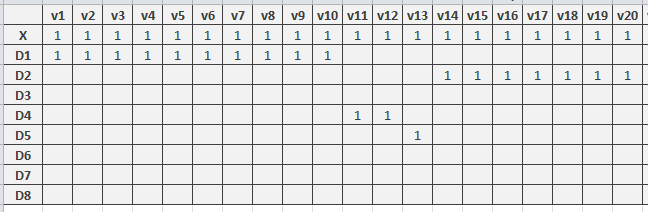
*– количество связей текущего элемента с множеством нераспределённых элементов;*

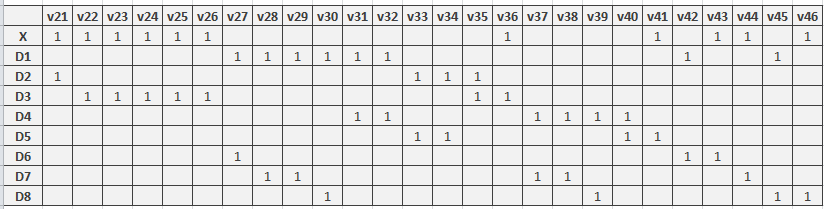
*– количество внешних связей узла, полученного добавлением к уже распределённым элементам элемента xt;*

*– количество внутренних связей между кандидатом в узел и элементами узла.*

Процесс продолжается, пока все элементы не распределены, или, если не может выполняться ограничение на количество элементов на одной плате, переходим к следующей плате и заново считаем L1.

Пример. Матрица элементных комплексов Q:





Компоновка схемы при ограничениях:

На количество элементов на плате n = 4;

На количество проводов m = 20.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r | X1r | L1 | E1r | X2r | L2 | L3 | E2r | X3r | L2 | L3 | E3r | X4r | L2 | L3 | E4r |
| 1 | D1 | **8** | D1 | D2 | 29 | - | D1 | D2 | 28 | - | D1 | D2 | 31 | - | D1 |
|  | D2 | 3 |  | D3 | 25 | - | D6 | D3 | 23 | - | D6 | D3 | 24 | - | D6 |
|  | D3 | 1 |  | D4 | 24 | - |  | D4 | **19** | **6** | D4 | D5 | 24 | - | D4 |
|  | D4 | 6 |  | D5 | 21 | - |  | D5 | 20 | 5 |  | D7 | **18** | **1** | D7 |
|  | D5 | 3 |  | D6 | **16** | **2** |  | D7 | 18 | 3 |  | D8 | 19 | 1 |  |
|  | D6 | 2 |  | D7 | 19 | 2 |  | D8 | 17 | 2 |  |  |  |  |  |
|  | D7 | 4 |  | D8 | 18 | 2 |  |  |  |  |  |  |  |  |  |
|  | D8 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Последовательный алгоритм компоновки по мультиграфу**

В графе G(X, U), где X – множество вершин (элементов), U – множество связей (проводов) находится вершина xi, для которой , где – локальная степень (то есть количество связей вершины с остальными). Если таких вершин несколько, выбирается та, у которой минимальное количество связанных с ней вершин. Эта вершина помещается на плату X1.

Далее из вершин, не содержащихся на плате Х1 и инцидентных вершине xi (связанных с ней) находится та вершина xr, которая обеспечивает минимальное приращение числа внешних связей. Приращение , где – количество связей данного элемента с множеством элементов узла.

Процесс продолжается, пока соблюдаются оба ограничения – на количество внешних связей и элементов в узле (на плате).

Пример. Матрица соединений R и достроенный столбец локальных степеней:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | Локальная степень вершины |
| D1 |  |  |  | 2 |  | 2 | 2 | 2 | 8 |
| D2 |  |  | 1 |  | 2 |  |  |  | 3 |
| D3 |  | 1 |  |  |  |  |  |  | 1 |
| D4 | 2 |  |  |  | 1 |  | 2 | 1 | 6 |
| D5 |  | 2 |  | 1 |  |  |  |  | 3 |
| D6 | 2 |  |  |  |  |  |  |  | 2 |
| D7 | 2 |  |  | 2 |  |  |  |  | 4 |
| D8 | 2 |  |  | 1 |  |  |  |  | 3 |

Компоновка схемы при ограничениях:

На количество элементов на плате n = 4;

На количество проводов m = 20.

Шаг 1

На плате – пусто.

D3 имеет минимальное – помещаем элемент D3 на плату.

Количество внешних связей = 1. Количество элементов = 1.

Шаг 2

На плате – D1.

Кандидаты: D2.

Помещаем элемент D2 на плату.

Количество внешних связей = 2. Количество элементов = 2.

Шаг 3

На плате – D1, D2.

Кандидаты: D5.

Помещаем элемент D5 на плату.

Количество внешних связей = 1. Количество элементов = 3.

Шаг 4

На плате – D1, D2, D5.

Кандидаты: D4.

Помещаем элемент D4 на плату.

Количество внешних связей = 5. Количество элементов = 4.

Результат  
 На плате – D1, D2, D5, D4.

**Итерационный алгоритм компоновки по мультиграфу**

Некоторое начальное разрезание (в нашем случае полученное с помощью последовательного алгоритма) улучшается в соответствии с выбранным критерием с помощью итерационного парного обмена вершин из различных кусков (плат).

Получаем приращение рёбер после перестановки пары вершин по формуле:

*,* где

- первая скобка обозначает количество внешних минус внутренних связей первого узла ко второму;

- вторая скобка обозначает количество внешних минус внутренних связей второго узла к первому;

- третье слагаемое обозначает удвоенное количество связей между узлами.

Имеет смысл менять узлы местами при > 0. При нескольких таких вариантах следует выбрать максимальное значение.

Пример. Допустим, имеются две скомпонованные платы – D1, D4, D6, D7 и D2, D3, D5, D8. Следует перестроить матрицу соединений R (меняя в ней столбцы и строки) так, чтобы элементы одного узла оказались рядом, после чего вычислить приращение.

Первая итерация.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D4 | D6 | D7 | D2 | D3 | D5 | D8 | 1 - 2 |
| D1 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 2 | -4 |
| D4 | 2 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | +1 |
| D6 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |
| D7 | 2 | 0 | 0 | 2 | 0 | 0 | 1 | 1 | -2 |
| D2 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | -1 |
| D3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | +1 |
| D5 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | +3 |
| D8 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | +3 |

= 1 + 3 – 2\*2 = 0

= 1 + 3 – 2\*0 = 4

Меняем местами четвёртый и восьмой узлы, переставляя соответствующие строки и столбцы таблицы и меняя их на платах.

Вторая итерация.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D1 | D8 | D6 | D7 | D2 | D3 | D5 | D4 | 1 - 2 |
| D1 | 0 | 2 | 2 | 2 | 0 | 0 | 0 | 2 | -4 |
| D8 | 2 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | -1 |
| D6 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |
| D7 | 2 | 1 | 0 | 2 | 0 | 0 | 1 | 0 | -3 |
| D2 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | -1 |
| D3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 |
| D5 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 2 | +1 |
| D4 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | +1 |

Вычислим возможные положительные :

= 1 - 1 - 2\*2 = -4

= 1 - 1 – 2\*0 = 0

= 1 - 1 - 2\*0 = 0

= 1 - 1 – 2\*0 = 0

Как видно из расчётов, не существует больше положительных что значит, что процесс завершён.

**Итерационный алгоритм компоновки по гиперграфу**

Рассмотрим граф, представленный двумя кусками – X1 и Х2, где (Х1 + Х2) = Х (исходный граф). Приращение рёбер после перестановки пары вершин xi X1 и xj X2 равно:

*,* где

- – количество рёбер, которые удалятся из разреза после перестановки пары вершин;

- – количество рёбер, которые появятся в разрезе после перестановки пары вершин.

Осуществляется цикл по всем ребрам сначала для вершины xi (вычисляются величины и ). а потом – для xj (вычисляются величины и ).

Рассмотрим ребро uk, инцидентное вершине xi

1. Ребро будет удалено из разреза между кусками при перестановке вершин xi и xj, если множество Xk () не содержит вершину xj и не содержит ни одну вершину из X1, кроме xi.
2. Ребро появится в разрезе между кусками при перестановке вершин xi и xj, если Xk не содержит ни одной вершины из X2.

Рассмотрим ребро ul, инцидентное вершине xj

1. Ребро будет удалено из разреза между кусками при перестановке вершин xi и xj, если множество Xl () не содержит вершину xi и не содержит ни одну вершину из X2, кроме xj.
2. Ребро появится в разрезе между кусками при перестановке вершин xi и xj, если Xl не содержит ни одной вершины из X1.

Эти четыре случая можно описать формулами:

Пример.

Два куска – X1 (x1, x2, x3, x4) и X2 (x5, x6, x7).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 | v7 |
| x1 | 1 |  |  | 1 |  |  | 1 |
| x2 | 1 | 1 |  |  | 1 | 1 |  |
| x3 |  |  | 1 |  |  |  | 1 |
| x4 | 1 |  | 1 |  |  |  | 1 |
| x5 |  | 1 | 1 |  | 1 |  |  |
| x6 |  |  |  | 1 | 1 | 1 |  |
| x7 |  |  |  | 1 |  |  | 1 |

Вычислим величину .

X1 (x1, x2, x3, x4) и X2 (x5, x6, x7). xi = x2, xj = 7.

1. Рассматриваем xi

u2 = v1, v2, v5, v6.

2. Рассматриваем xj

u7 = v4 v7

**Алгоритмы размещения**

**Последовательный алгоритм размещения по критериям – максимум связей с размещённым элементом, максимум связей с размещёнными элементами, максимум связей с размещёнными элементами и минимум связей с неразмещёнными элементами**

Для размещения на плате понадобится матрица соединений R. В вектор-столбце вычислим количество связей с остальными элементами и отберём первый элемент по определённому критерию и поместим его на плате. Критерием выбора первого элемента может быть максимум или минимум связей с остальными элементами, максимум или минимум связей с разъёмом, и так далее.

После размещения первого элемента по заданному из трёх критерию будем выбирать элемент, и размещать его на плате. При равном количестве связей будем брать первый по порядку элемент. При спорных ситуациях, когда одинаково выгодно разместить элемент в нескольких местах следует считать длину связей по формуле:

Следует помнить о том, критерии необходимо вычислять только для множества неразмещённых элементов, а разъём считается размещённым по умолчанию.

При критерии максимум связей с ранее размещённым элементом.

Пример – при размещении на первом шаге D6.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | Шаг первый, кол-во связей с D6 |
| X | 0 | 10 | 8 | 6 | 2 | 2 | 1 | 1 | 1 | ­­­­– |
| D1 | 10 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | **2** |
| D2 | 8 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 |
| D3 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D4 | 2 | 2 | 0 | 0 | 0 | 1 | 0 | 2 | 1 | 2 |
| D5 | 2 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| D6 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |
| D7 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| D8 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

При критерии максимум связей с ранее размещёнными элементами.

Пример – при размещении на первом шаге D6, на втором – D1.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | Шаг второй, кол-во связей с D6 и D1 |
| X | 0 | 10 | 8 | 6 | 2 | 2 | 1 | 1 | 1 | ­­­­– |
| D1 | 10 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | – |
| D2 | 8 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 |
| D3 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D4 | 2 | 2 | 0 | 0 | 0 | 1 | 0 | 2 | 1 | **2** |
| D5 | 2 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| D6 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |
| D7 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 2 |
| D8 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 |

При критерии максимум связей с ранее размещёнными элементами и минимум с неразмещёнными.

Пример – при размещении на первом шаге D6, на втором – D1, на третьем – D4.

В данном примере количество связей можно вычислить по формуле: *связей со всеми размещёнными – связей со всеми неразмещёнными*. То есть в нашем случае: *связей с D1, D4, D6 – связей с D2, D3, D5, D7, D8*.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | Шаг третий, кол-во связей с D6, D1, D4 –  кол-во связей с остальными |
| X | 0 | 10 | 8 | 6 | 2 | 2 | 1 | 1 | 1 | ­­­­– |
| D1 | 10 | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 | – |
| D2 | 8 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | -3 |
| D3 | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| D4 | 2 | 2 | 0 | 0 | 0 | 1 | 0 | 2 | 1 | – |
| D5 | 2 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | -1 |
| D6 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | – |
| D7 | 1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | **4** |
| D8 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 3 |

**Общий итерационный алгоритм размещения**

В этом методе для улучшения исходного размещения осуществляется перестановка всех возможных пар элементов в коммутационном поле.

Приращение равно , где  
f, g – позиции элементов i, j.  
p, h – номер элемента и позиция смежных по крайней мере с одним из элементов.

На каждой итерации выбирается пара, для которой больше нуля и максимальна.

Процесс повторяется до тех пор, пока для всех возможных пар элементов на некотором шаге не станет меньше или равно нулю.

**Итерационный алгоритм размещения по парам**

Метод аналогичен общему итерационному алгоритму размещения, однако приращение вычисляется не для всех возможных комбинаций, а для пар элементов.

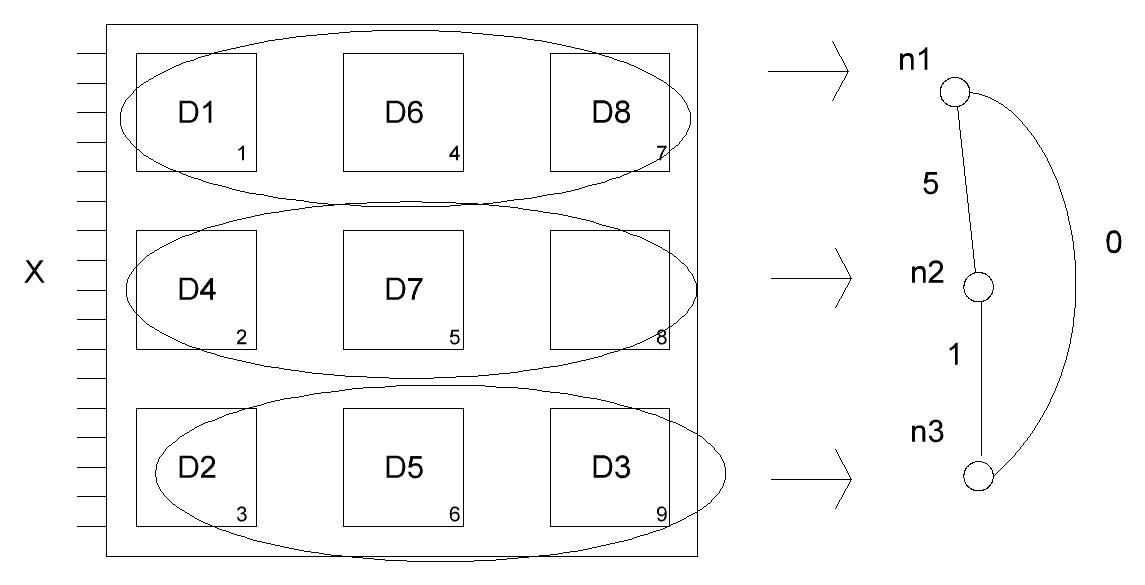
**Итерационный алгоритм размещения – метод Шафера**

Для метода Шафера требуется представить элементы по строкам и столбцам как один элемент.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
| X | 0 | 10 | 8 | 6 | 2 | 2 | 1 | 1 | 1 |
| D1 |  | 0 | 0 | 0 | 2 | 0 | 2 | 2 | 2 |
| D2 |  |  | 0 | 1 | 0 | 2 | 0 | 0 | 0 |
| D3 |  |  |  | 0 | 0 | 0 | 0 | 0 | 0 |
| D4 |  |  |  |  | 0 | 1 | 0 | 2 | 1 |
| D5 |  |  |  |  |  | 0 | 0 | 0 | 0 |
| D6 |  |  |  |  |  |  | 0 | 0 | 0 |
| D7 |  |  |  |  |  |  |  | 0 | 0 |
| D8 |  |  |  |  |  |  |  |  | 0 |

Например, по строкам. Надо вычислить количество связей элементов первой строки с элементами второй и третьей строк. Количество связей первой строки со второй – 5 (у D1 2 связи с D4, 2 с D7, у D8 1 связь с D4).

После этого вычисляются величины L12, L21, и если они больше нуля, строки надо менять местами.



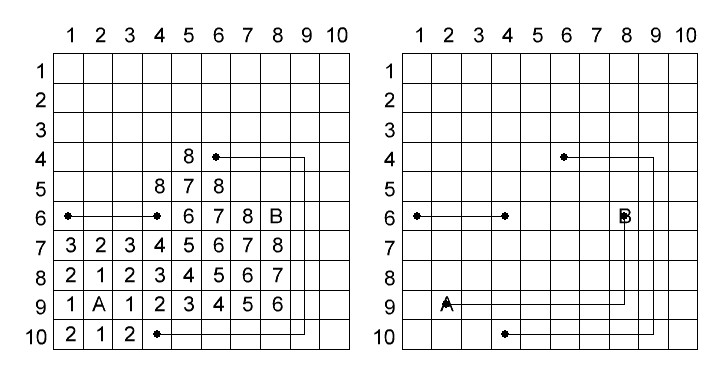
Аналогично по столбцам.

**Алгоритмы трассировки**

**Метод Ли**

На первом шаге соседним ячейкам точки А присваивается вес равный единице. На последующих шагах соседним с текущими клеткам присевается вес + 1. Это происходит, пока одна из соседних клеток не окажется клеткой В.

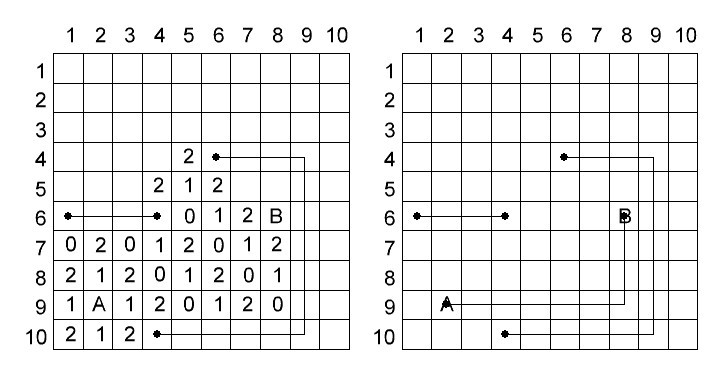
Проводник проводится в обратной последовательности – по уменьшению весов.



**Метод Ли по модулю три**

Присваиваемый ячейкам вес кодируется по модулю три.  
Вместо 1, 2, 3, 4, 5, 6, 7 – 1, 2, 0, 1, 2, 0, 1.

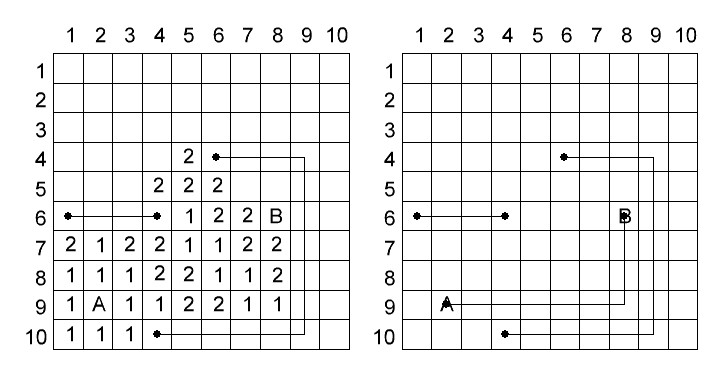
Проводник проводится в обратной последовательности – 0, 2, 1, 0, 2, 1.

****

**Метод Аккерса**

Присваиваемый ячейкам вес кодируется следующей последовательностью – 1, 1, 2, 2, 1, 1. То есть вместо 1, 2, 3, 4, 5, 6, 7 – 1, 1, 2, 2, 1, 1, 2.

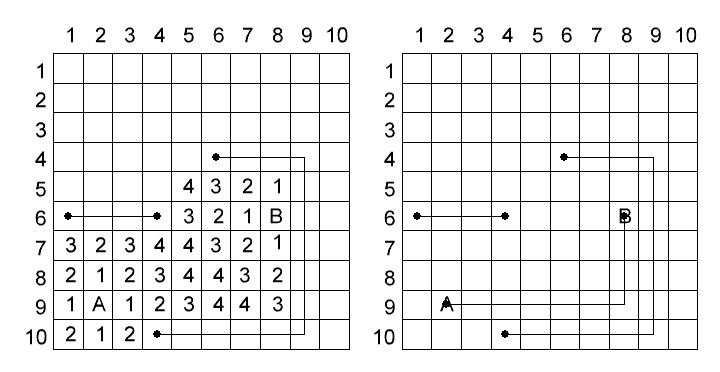
Проводник проводится в обратной последовательности – 2, 2, 1, 1, 2, 2.



**Метод встречной волны**

Метод Ли применяется не только к точке А, но и к точке В одновременно.

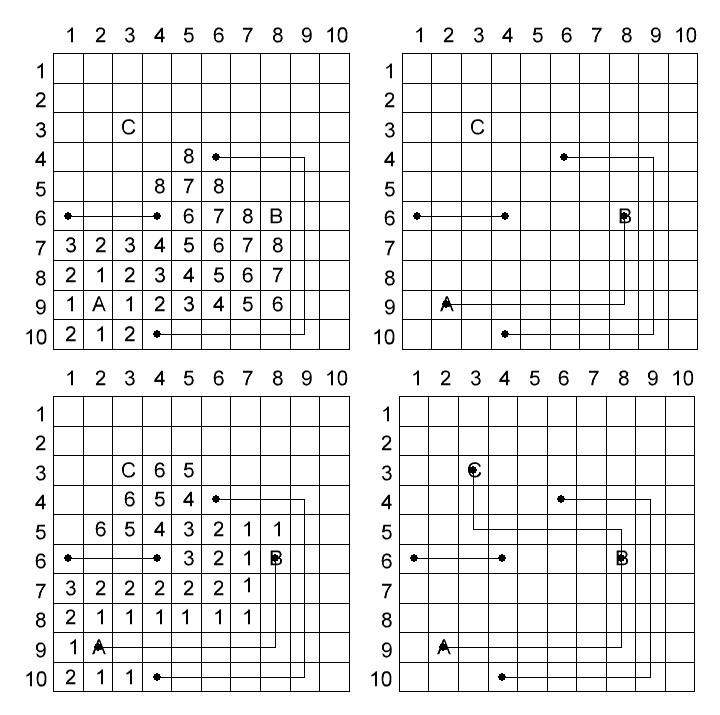
Проводник проводится в прямой последовательности до места встречи волн и в обратной после него.



**Метод соединения комплекса**

Используется для трассировки нескольких точек.

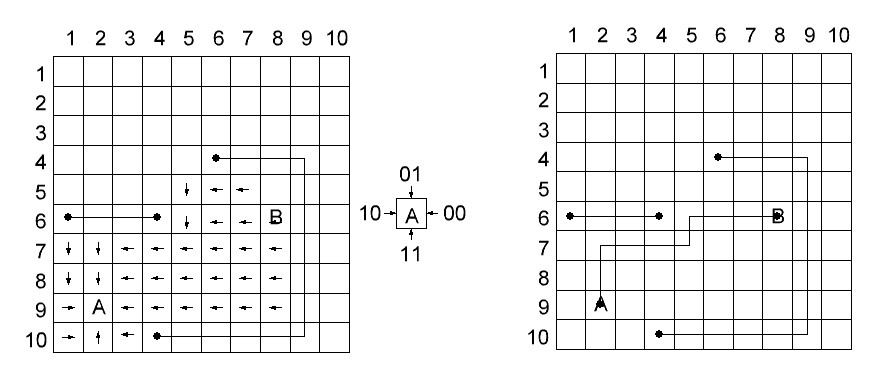
По методу Ли находится путь к одной из точек – ближайшей. После этого ДРП обнуляется и вес, равный единице присваивается не соседним точкам от точки В, а соседним точкам для всего ранее проведённого проводника.

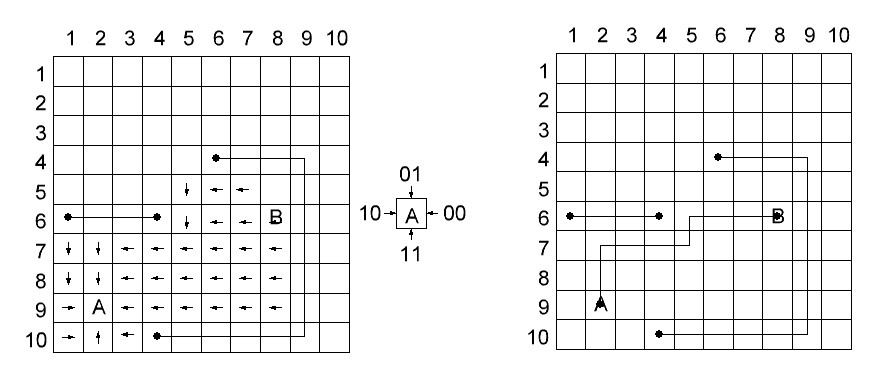


**Метод путевых координат**

Четырём направлениям присваиваются приоритеты от 00 до 11, где 00 – высший приоритет, 11 – низший. Приоритеты присваиваются по расстояниям от точки В к точке А – то есть приоритеты устанавливаются относительно источника. Так, если точка А находится от точки В на расстоянии 6 клеток влево и 3 клеток вниз, то высший приоритет 00 – влево, потом 01 – вниз, остальные два направления – по желанию.

В случае, когда подойти к точке можно с нескольких сторон, выбирается направление с наивысшим приоритетом. Когда вариант один, используется единственное направление.

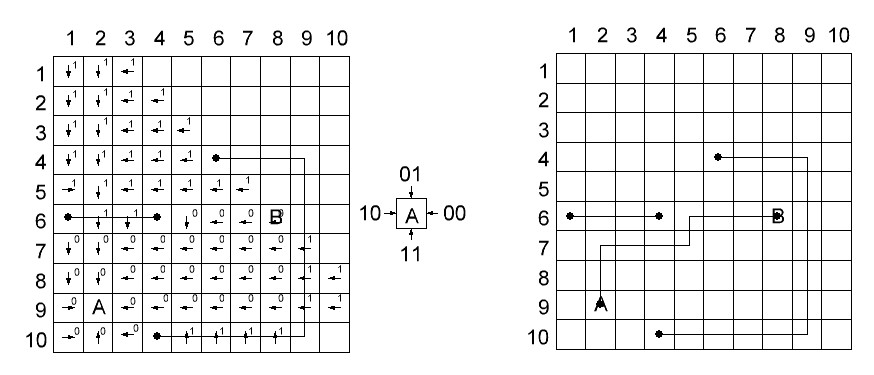


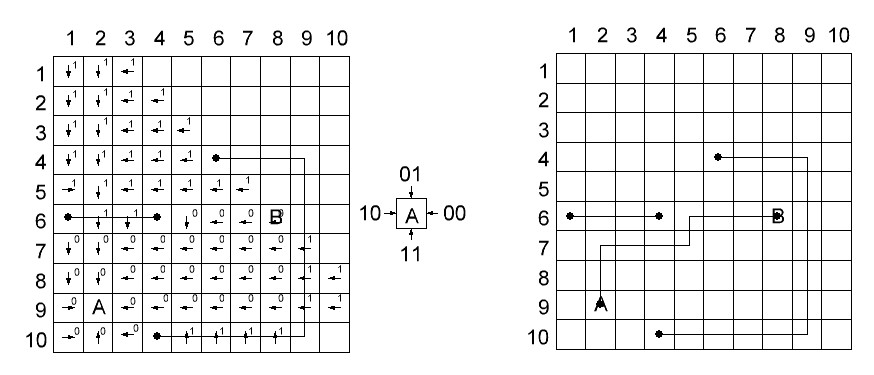


**Метод с минимальным числом пересечений**

Ячейки делятся на свободные, занятые и полузанятые. Занятыми считаются ячейки, на которых находятся выводы проводников, их изгибы и пересечения. Полузанятыми считаются ячейки, заполненные прямыми вертикальными и горизонтальными проводниками, учитывая, что проводник нельзя проводить параллельно уже проведённому проводнику – только перпендикулярно.

Вес текущей ячейки равен весу предыдущей ячейки, учитывая, что текущая ячейка свободна, и увеличивается на единицу, если текущая ячейка полузанятая.

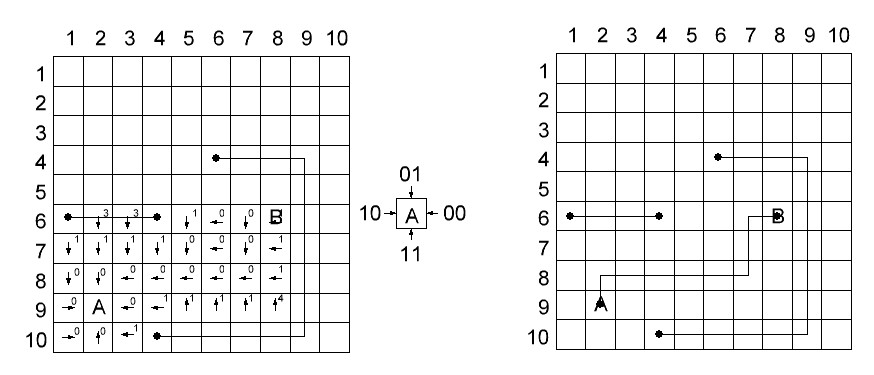




**Метод равномерного распределения проводников**

Вес текущей ячейки принимается равному весу предыдущей ячейки + число соседних ячеек, через которые проходят проводники. При возможности провести направление к нескольким точкам из одной ячейки, проводится направление к ячейке с меньшим приоритетом (смотри клетку (7;5) в примере).

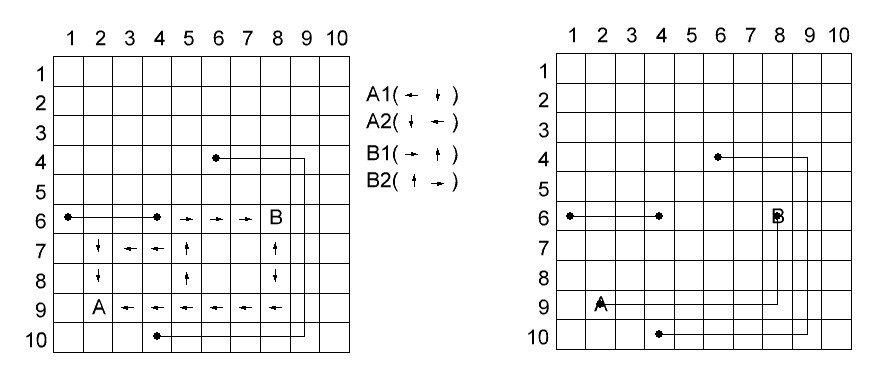
Используются и свободные, и полузанятые ячейки.



**Двухлучевой метод**

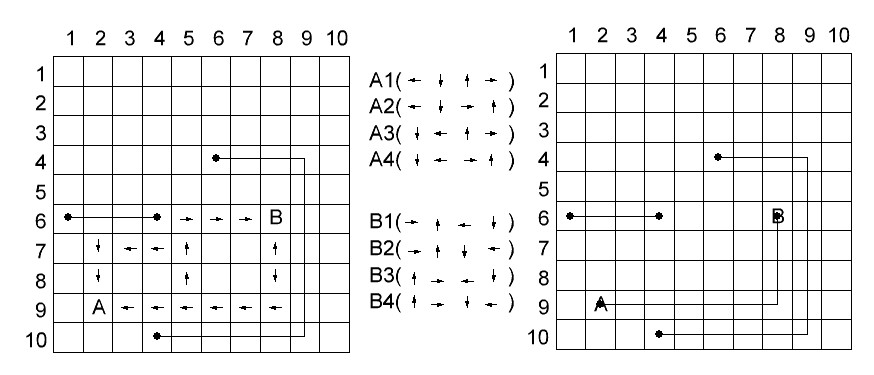
От источника и приёмника проводятся по два луча А1 и А2, В1 и В2. У лучей по два приоритета. В случае, когда подойти к точке можно с нескольких сторон, выбирается направление с наивысшим приоритетом. Когда вариант один, используется единственное направление.

Не всегда решает задачу трассировки – лучи могут не встретиться.



**Четырёхлучевой метод**

Аналогичен двухлучевому, но проводятся четыре луча с четырьмя приоритетами.



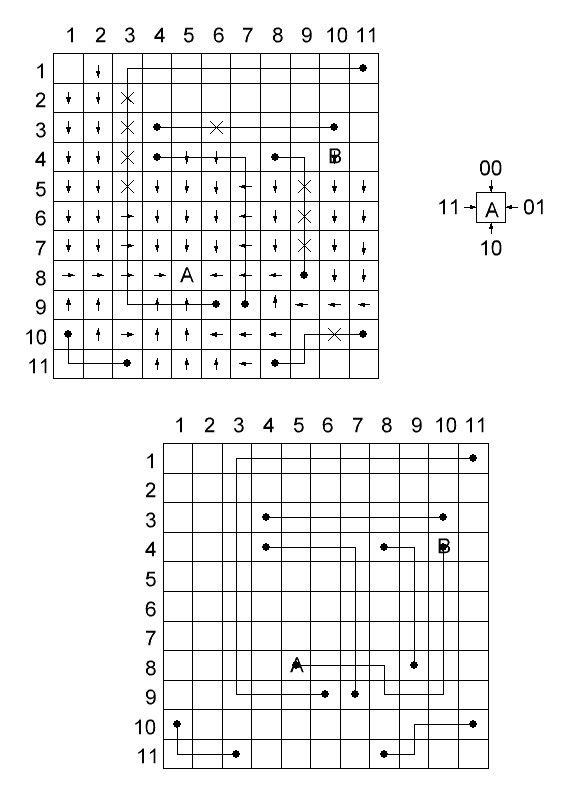
**Метод, оптимизированный по длине проводника и числу пересечений**

Используются следующие переменные:

1. L – источники числовой волны (текущие ячейки).
2. L1 – свободные ячейки относительно списка L.
3. L2 – полузанятые ячейки относительно списка L.
4. L3 – индекс длины до источника.

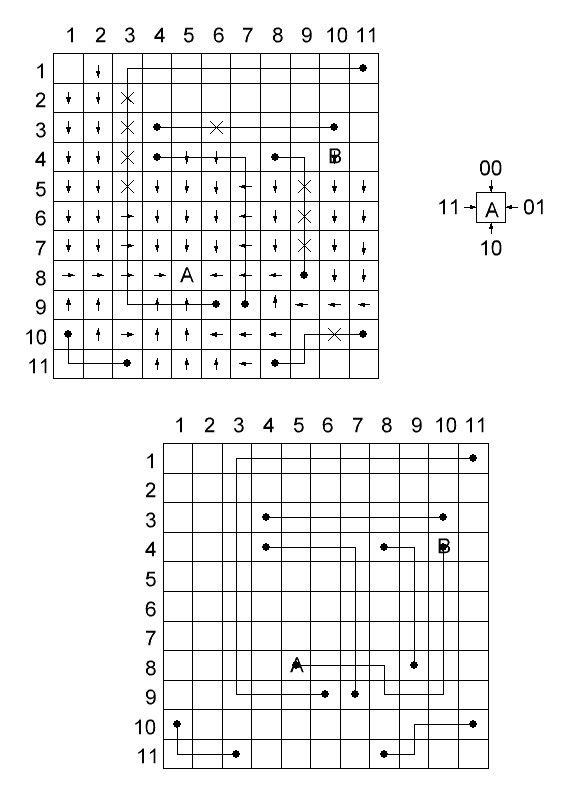
Алгоритм:

1. В окрестности первой ячейки (точки-источника) из списка L выбираются свободные и полузанятые ячейки. Индекс длины обнуляется. Переход к пункту 2.
2. Каждой из этих ячеек приписываются путевые координаты в соответствии с приоритетом. Переход к пункту 3.
3. Свободные ячейки (относительно текущих) записываются в L1, полузанятые в – L2. Индекс длины инкрементируется. Переход к пункту 4.
4. Шаги 1 – 3 повторяются для всех ячеек из L1, проверяя наличие ячейки В в L1. Если она найдена, переход к пункту 11, иначе – к пункту 5.
5. Список L1 пополняется теми ячейками из L2, которые имеют тот же индекс длины, а если L1 пуст, ячейками из L2 с минимальным индексом длины. Переход к пункту 6.
6. Списку L присваивается L1, L1 очищается. Переход к пункту 7.
7. Шаги 1 – 6 повторяются, пока L1 не станет пустым (что означает, что построить путь без пересечений невозможно). Переход к пункту 8.
8. Ячейки из L2 с минимальным индексом длины записываются в L. Переход к пункту 9.
9. Шаги 1 – 9 повторяются, пока не произойдёт одно из двух. Точка В окажется в L1 – переход к пункту 10, или L = L1 = L2 = 0 – переход к пункту 11.
10. Задача не имеет решения.
11. Конец алгоритма.



Подчёркнутым шрифтом обозначены ячейки, которые были переписаны из L2, **жирным** – координаты точки В. Крестиками обозначены полузанятые ячейки из L2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шаг | L | L1 | L2 | L3 |
| 1 | (8;5) | (8;4) (7;5) (8;6) | (9;5) | 1 |
| 2 | (8;4) (7;5) (8;6) | (7;4) (6;5) (7;6) | (8;3) (8;7) (9;4) | 2 |
| 3 | (7;4) (6;5) (7;6) | (6;4) (5;5) (6;6) | (7;3) (7;7) | 3 |
| 4 | (6;4) (5;5) (6;6) | (5;4) (5;6) | (6;3) (4;5) (6;7) | 4 |
| 5 | (5;4) (5;6) | - | (5;3) (4;6) (5;7) | 5 |
| 6 | (9;5) | (10;5) (8;3) (8;7) (9;4) |  |  |
| 7 | (10;5) (8;3) (8;7) (9;4) | (8;2) (10;4) (11;5) (10;6) (8;8) (7;3) (7;7) |  |  |
| 8 | (8;2) (10;4) (11;5) (10;6) (8;8)  (7;3) (7;7) | (7;2) (8;1) (9;2) (10;3)  (11;4) (11;6) (10;7) (9:8) (7;8) (6;3) (4;5) (6;7) |  |  |
| 9 | (7;2) (8;1) (9;2) (10;3)  (11;4) (11;6) (10;7) (9:8)  (7;8) (6;3) (4;5) (6;7) | (8;2) (7;1) (9;1) (10;2)  (11;7) (10;8) (9:9)  (5;3) (4;6) (5;7) | (7;9) (3;5) | 6 |
| 10 | (8;2) (7;1) (9;1) (10;2)  (11;7) (10;8) (9:9)  (5;3) (4;6) (5;7) | (6;1) (5;2) (5;8) (9;10) | (11;2) (3;6) (6;9) | 7 |
| 11 | (6;1) (5;2) (5;8) (9;10) | (5;1) (4;2) (8;10) (9;11) | (5;9) (10;10) | 8 |
| 12 | (5;1) (4;2) (8;10) (9;11) | (4;1) (3;2) (7;10) (8;11) | (4;3) | 9 |
| 13 | (4;1) (3;2) (7;10) (8;11) | (3;1) (2;2) (6;10) (7;11) | (3;3) | 10 |
| 14 | (3;1) (2;2) (6;10) (7;11) | (2;1) (1;2) (5;10) (6;11) | (3;2) | 11 |
| 15 | (2;1) (1;2) (5;10) (6;11) | (1;1) **(4;10)** (5;11) |  |  |



**Метод Хейса**

Для каждого слоя используются следующие переменные:

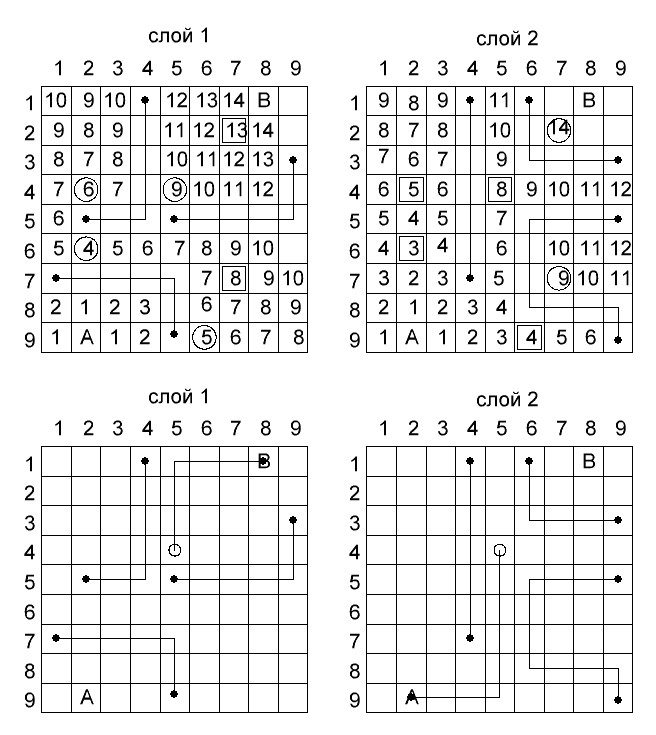
1. ДРПi – состояние ячеек i-того слоя печатной платы.
2. Pi – индекс длины.
3. Li – состояние текущего фронта волны в i-того
4. Mi – ячейки слоя i, соседние для ячеек из списка Li.
5. Vi – количество переходов на слой i.

Алгоритм (для двухслойной печатной платы):

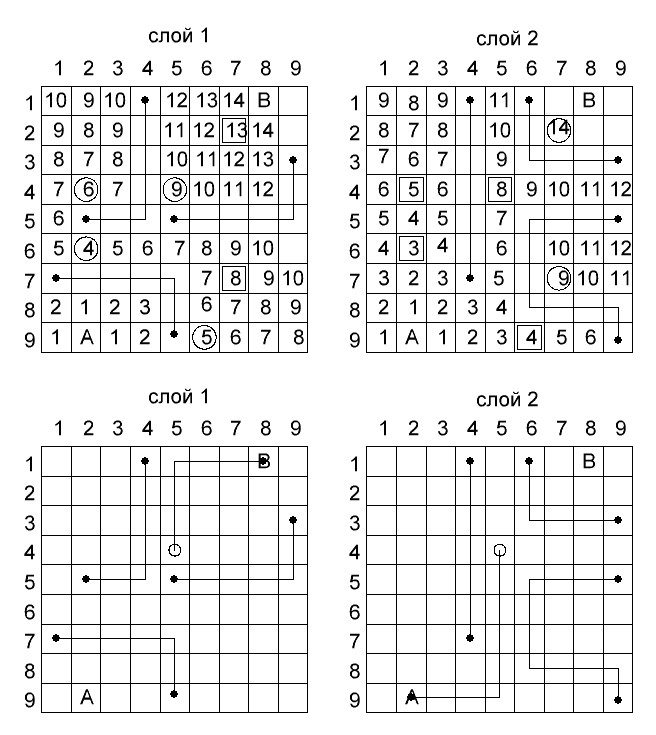
1. Индекс длины увеличивается на единицу. Переход к пункту 2.
2. Исследуем окрестность ДРП1 из списков L1. Свободные и неотмеченные ячейки записываем в M1. Переход к пункту 3.
3. Для каждой из ячеек в M1 присваивается индекс длины и перехода. Переход к пункту 4.
4. Шаги 1 – 3 повторяются для всех ячеек из L1. Переход к пункту 5.
5. Списку L1 присваивается M1, M1 очищается. Переход к пункту 6
6. Список L1 пополняется теми ячейками из L2, в которые возможен переход из слоя 2 в слой 1 и которые не были отмечены в ДРП1. Если такие есть, индекс перехода увеличивается на единицу. Переход к пункту 7.
7. Этим ячейкам в ДРП1 присваивается индекс длины и перехода. Переход к пункту 4.
8. Шаги 2 – 7 для слоя 2. Переход к пункту 9.
9. Если ячейка В не обнаружилась в L1 или L2, повторяются шаги 1-8. Если эти списки пусты, то пути не существует.

Если ячейка В обнаружилась, то к ней строится путь, учитывая минимальное количество переходов со слоя на слой.

Переход из слоя отображается квадратиком, в слой – кружочком.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Шаг | L1 | M1 | V1 | L2 | M2 | V2 |
| 1 | (9;2) | (9;1) (8;2) (9;3) | 0 | (9;2) | (9;1) (8;2) (9;3) | 0 |
| 2 | (9;1) (8;2) (9;3) | (8;1) (8;3) (9;4) | 0 | (9;1) (8;2) (9;3) | (8;1) (8;3) (9;4) (7:2) | 0 |
| 3 | (8;1) (8;3) (9;4) | (8;4) | 0 | (8;1) (8;3) (9;4) (7:2) | (7;1) (6;2) (7;3) (8:4) (9;5) | 0 |
| 4 | (8;4) | - | 0 | (7;1) (6;2) (7;3) (8:4) (9;5) | (6;1) (5;2) (6;3) (8:5) (9;6) | 0 |
| 5 | (6;2) | (6;1) (6;3) | 1 | (6;1) (5;2) (6;3) (8:5) (9;6) | (5;1) (4;2) (5;3) (7:5) (9;7) | 0 |
| 6 | (6;1) (6;3) (9;6) | (5;1) (6;4) (8;6) (9;7) | 2 | (5;1) (4;2) (5;3) (7:5) (9;7) | (4;1) (3;2) (4;3) (6:5) (9;8) | 0 |
| 7 | (5;1) (6;4) (8;6) (9;7) (4;2) | (4;1) (3;2) (4;3) (6;5) (7;6) (8;7) (9;8) | 3 | (4;1) (3;2) (4;3) (6:5) (9;8) | (3;1) (2;2) (3;3) (5:5) | 0 |
| 8 | (4;1) (3;2) (4;3) (6;5) (7;6) (8;7) (9;8) | (3;1) (2;2) (3;3) (6;6) (7;7) (8;8) (9;9) | 3 | (3;1) (2;2) (3;3) (5:5) | (2;1) (1;2) (2;3) (4:5) | 0 |
| 9 | (3;1) (2;2) (3;3) (6;6) (7;7) (8;8) (9;9) | (2;1) (1;2) (2;3) (6;7) (7;8) (8;9) | 3 | (2;1) (1;2) (2;3) (4:5) | (1;1) (1;3) (3;5) (4:6) | 0 |
| 10 | (2;1) (1;2) (2;3) (6;7) (7;8) (8;9) (4;5) | (1;1) (1;3) (3;5) (4;6) (6;8) (7;9) | 4 | (1;1) (1;3) (3;5) (4:6) (7;7) | (2;5) (4:7) (7;7) (6;7) (6;8) | 1 |
| 11 | (1;1) (1;3) (3;5) (4;6) (6;8) (7;9) | (2;5) (3;6) (4;7) (6;9) | 4 | (2;5) (4:7) (7;7) (6;7) (6;8) | (1;5) (4:8) (6;8) (7;9) | 1 |
| 12 | (2;5) (3;6) (4;7) (6;9) | (1;5) (2;6) (3;7) (4;8) | 4 | (1;5) (4:8) (6;8) (7;9) | (4:9) (6;9) | 1 |
| 13 | (1;5) (2;6) (3;7) (4;8) | (1;6) (2;7) (3;8) | 4 | (4:9) (6;9) | - | 1 |
| 14 | (1;6) (2;7) (3;8) | (1;7) (2;8) | 4 | - | - | 1 |
| 15 | (1;7) (2;8) | **(1;8)** (2;9) | 4 | (2;7) | (1;7) (2;8) | 2 |



**Алгоритмы расслоения**

**Последовательный алгоритм расслоения**

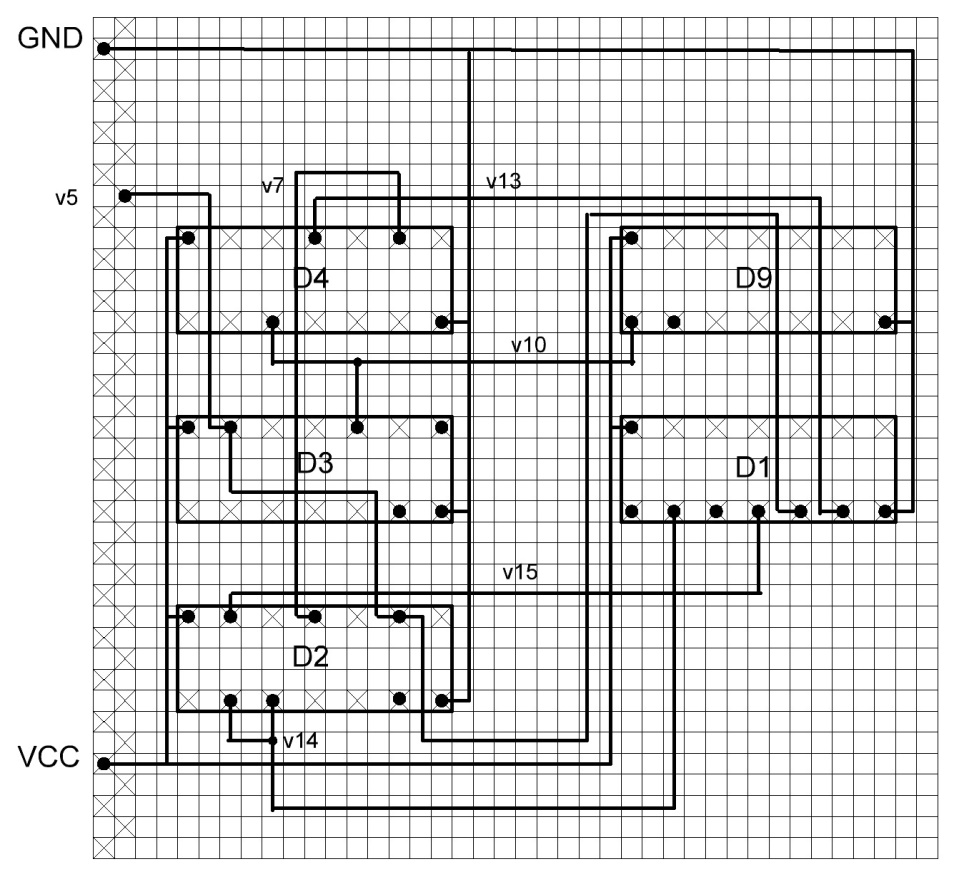
По схеме разведённых контактов составляется матрица пересечений проводников P (n x n), где n – количество проводников, каждый элемент которой указывает на количество пересечений проводника с проводником.

После составления матрицы P составляется вектор-столбец (сумма по строкам), который будет указывать на количество пересечений конкретного проводника со всеми остальными.

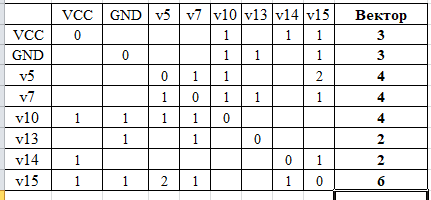
Выбирается максимальный элемент из вектор-столбца и он переносится на отдельный слой, соответственно, удаляется из матрицы. Когда весь вектор столбец будет нулевым, получим готовый первый слой печатной платы – проводники на нём не пересекаются.

После этого матрица P составляется заново для проводников, которые не были перенесены на первый слой, и алгоритм начинается заново.

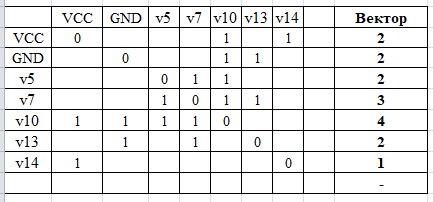
Пример. ДРП:



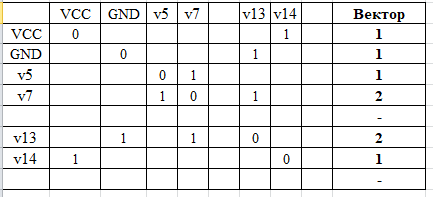
Матрица P:



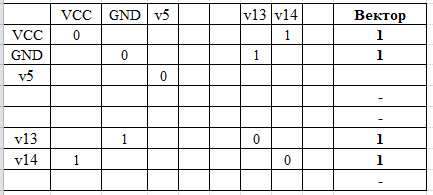
Убираем на следующий слой – v15



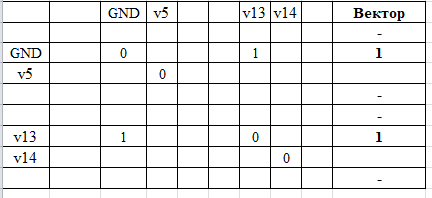
Убираем на следующий слой – v15, v10



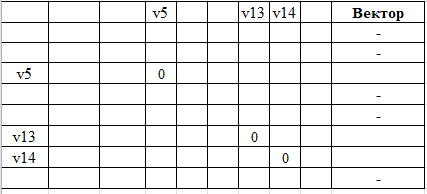
Убираем на следующий слой – v15, v10, v7



Убираем на следующий слой – v15, v10, v7, VCC

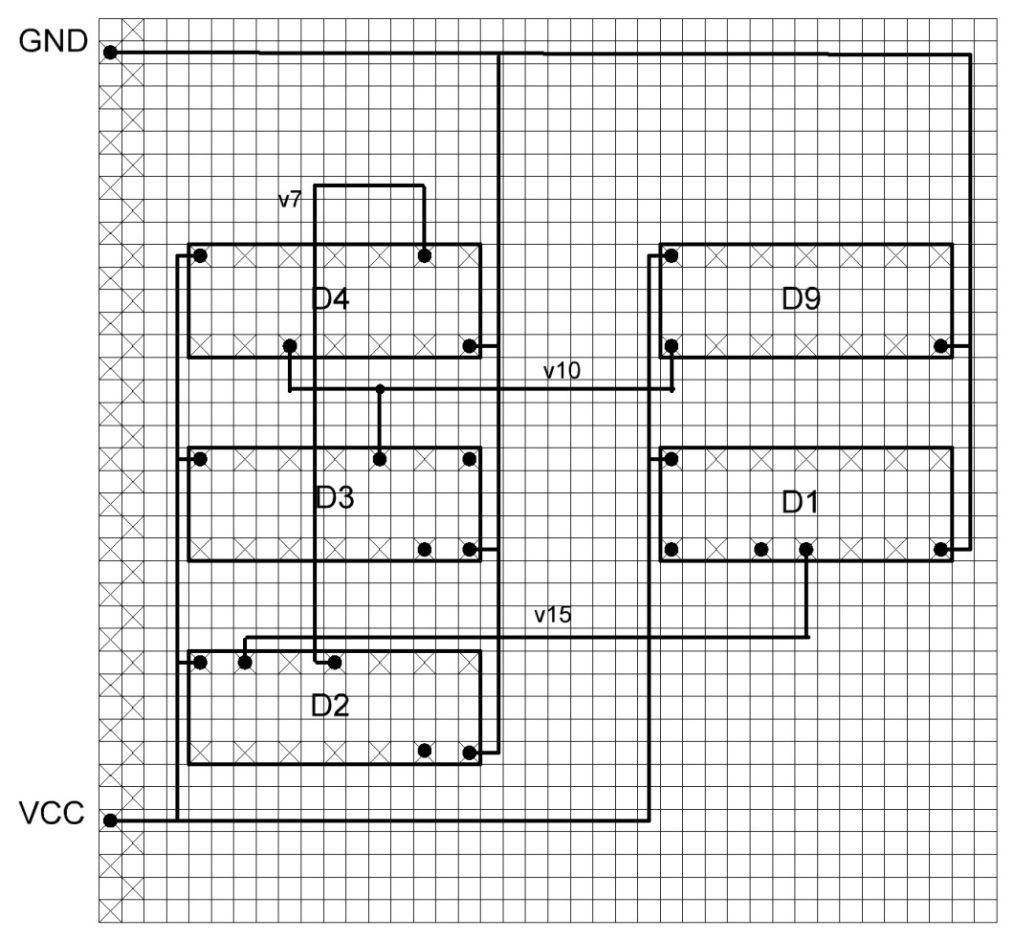


Убираем на следующий слой – v15, v10, v7, VCC, GND



Вектор нулевой – первый слой платы сформирован – v5, v13, v14.

Оставшиеся на ДРП проводники:



Для обновлённого ДРП снова составляется матрица P и алгоритм начинается с начала.