АЛГОРИТМИЧЕСКИЕ МЕТОДЫ КОНСТРУИРОВАНИЯ ЭВС

§ 1. ОБЩАЯ ХАРАКТЕРИСТИКА ОСНОВНЫХ ЗАДАЧ ЭТАПА КОНСТРУКТОРСКОГО ПРОЕКТИРОВАНИЯ

В основу построения современных вычислительных комплексов положен модульный принцип. Он предполагает иерархическое построение ЭВС из модулей, разбитых на несколько конструктивных уровней (рис. 1.1).

Рис. 1.1. Иерархия конструктивных уровней.

К числу основных конструкторских задач, типовых для всех уровней конструктивной иерархии, относятся:

- ✓ компоновка конструктивных модулей;
- ✓ размещение модулей низших иерархических уровней в монтажном пространстве модулей следующих уровней иерархии;
- ✓ трассировка монтажных соединений;
- ✓ получение конструкторской документации;
- ✓ тестирование аппаратуры.

Компоновка – этап конструкторского проектирования, заключающийся в распределении всех функциональных элементов схемы в группы, соответствующие конструктивным единицам различного ранга, из которых реализуются разрабатываемые ЭВС. На этапе компоновки осуществляется распределение элементов в микросхемы (МС), МС в ячейки, ячеек в панели, панелей в шкафы и т.д.
В дальнейшем будем называть конструктивную единицу низшего ранга – элементом, конструктивную единицу высшего ранга – блоком.
Возможны два варианта практической постановки задачи компоновки:

1) компоновка схемы конструктивно унифицированными блоками, состав которых формируется непосредственно в процессе компоновки (количество выводов и элементов блока известно заранее);
2) компоновка схемы функционально типизированными блоками из заданного набора.

В первом случае задача заключается в разбиении схемы устройства на блоки заданного размера с известным числом элементов и количеством внешних выводов. Это так называемая задача разбиения. Задачами такого типа являются задачи разбиения по ТЭЗам, ТЭЗов по панелям, разбиение БИС на ИС частного применения.
При решении задачи компоновки обычно используются следующие критерии качества:

а) минимальное число межблочных связей;
б) минимальное количество блоков;
в) минимальное число типов используемых блоков;
г) возможно более полное использование каждого блока.

В качестве основных ограничений в задаче компоновки обычно выступают:

а) максимальное число элементов в блоке;
б) максимальное число внешних выводов блока;
в) ограничения на совместную или раздельную компоновку некоторых элементов, обусловленные тепловыми требованиями, требованиями помехозащищенности и т.д.

Размещение – задача определения такого местоположения элементов в заданном монтажном пространстве, при котором наилучшим образом удовлетворяются некоторые требования. В качестве элемента здесь могут выступать радиодетали, ИС, ячейки, панели, шкафы. При этом предполагается, что элементы в монтажном пространстве определенным образом соединяются между собой. Эти соединения могут быть выполнены посредством навесных или печатных проводников, жгутовых соединений или других информационных магистралей.

Имеются два типа задач размещения:

1) размещение однотипных элементов на заранее заданных и регулярно размещенных позициях;
2) размещение элементов разного типа, разногабаритных, когда установочные места заранее не определены, а установка элементов осуществляется в процессе размещения (БИС).

Задача размещения носит в общем случае многоцелевой характер, и при ее решении должна производится оптимизация по совокупности критериев качества. Однако, в каждом конкретном случае обычно удается выделить наиболее важный критерий, учитывая другие показатели в качестве ограничений задачи. Известные алгоритмы размещения оптимизируют следующие показатели качества:

а) суммарную длину всех соединений;
б) длину самой длинной связи;
в) число связей между модулями, находящимися в соседних позициях (максимизация);
г) число пересечений проводников;
д) число цепей с возможно более простой конфигурацией.

Трассировка монтажных соединений – задача реализации соединений между выводами модулей предыдущего ранга в монтажном пространстве модуля следующего ранга в соответствии со схемой и с учетом конструктивно-технологических ограничений.

Имеются два типа задач трассировки:

1) трассировка проводного монтажа;
2) трассировка печатного монтажа.

Критерии оптимизации:

а) минимум суммарной длины проводников;
б) минимальное число переходных отверстий;
в) минимальное число слоев;
г) максимальная удаленность проводников друг от друга (снижение значений паразитных емкостей).

Ограничения:

а) отсутствие пересечений проводников в одном слое для печатного монтажа;
б) ограниченное число паек к одному контакту при проводном монтаже.

Примечание 1. Крысиная цепь – кратчайшая цепь.

Примечание 2. При втором варианте компоновки (компоновка схемы функционально типизированными блоками из заданного набора) номенклатура блоков известна заранее, и задача заключается в представлении исходной схемы совокупностью связанных между собой блоков из заданного набора. Данная задача возникает при переходе от функциональной схемы к принципиальной – так называемая задача покрытия. Она состоит в привязке элементов функциональной схемы к конкретным типовым модулям из заданного набора. При этом каждый модуль может включать в себя несколько типовых элементов, в общем случае связанных между собой.

§ 2. МАТЕМАТИЧЕСКИЕ МОДЕЛИ СХЕМ ЭВС

Для описания функционально-логических схем цифровых устройств и принципиальных электрических схем аналоговых устройств используются три основные математические модели (ММ):

1) граф коммутационной схемы (ГКС);
2) гиперграф (ГГ);
3) взвешенный неориентированный граф (ВНГ).

Граф коммутационной схемы

Введем следующие обозначения:

$\boldsymbol{E} = \{\boldsymbol{e_0}, \boldsymbol{e_1}, \ldots, \boldsymbol{e_n}\}$ – множество элементов схемы, причем $\boldsymbol{e_0}$ соответствует фиктивному элементу, например, разъему.

$\boldsymbol{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_l}\}$ – множество цепей схемы.

$\boldsymbol{K_i} = \{\boldsymbol{k_{i1}}, \boldsymbol{k_{i2}}, \ldots, \boldsymbol{k_{i\rho}}\}$ – множество контактов $\boldsymbol{i}$-го элемента.

$\boldsymbol{K} = \bigcup_{i=0}^{n} \boldsymbol{K_i}$ – множество контактов всех ($\boldsymbol{n}$+1) элементов схемы.

$\boldsymbol{ÃÊÑ = G = (Y, U)}$

$\boldsymbol{Y = E\ U\ V\ U\ K}$

$\boldsymbol{U = W\ U\ F}$

ГКС – треххроматический граф с вершинами трех типов («элемент», «контакт», «цепь») и ребрами двух типов $\boldsymbol{W}$ и $\boldsymbol{F}$ («элемент–контакт» – $\boldsymbol{W}$, «контакт–цепь» – $\boldsymbol{F}$).

Для примера рассмотрим схему, приведенную на рисунке 2.1.

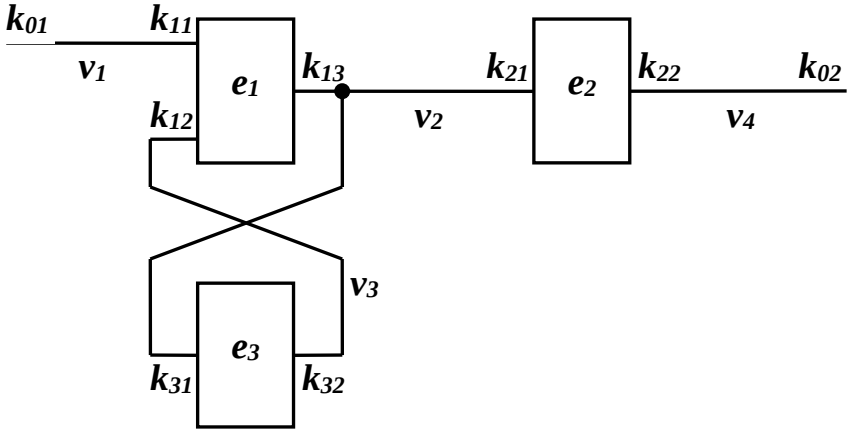


Рис. 2.1.

ГКС для схемы рис. 2.1 приведен на рисунке 2.2.

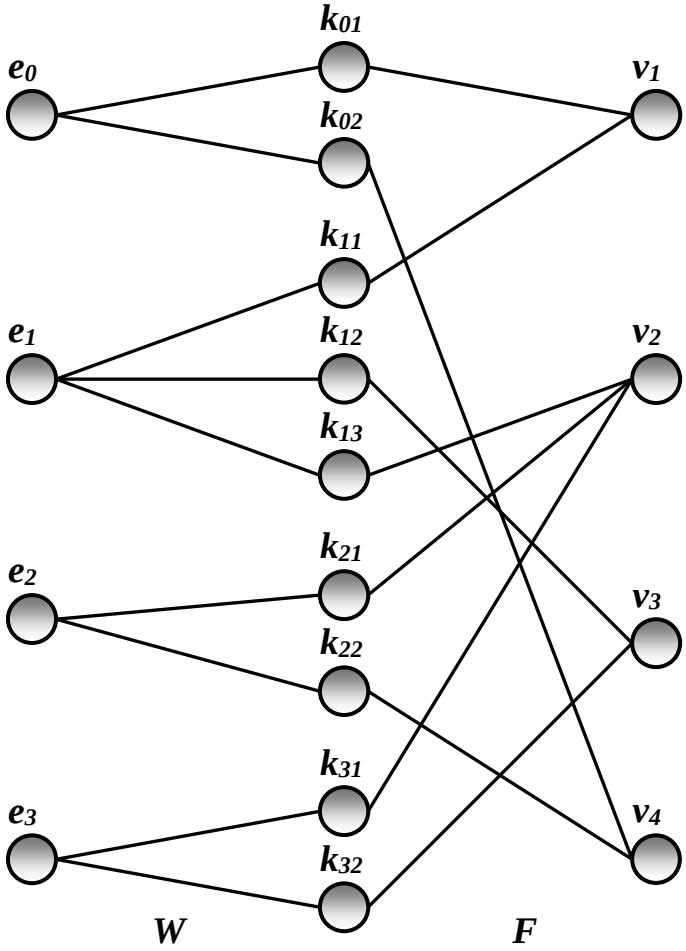


Рис. 2.2.

Для описания ГКС можно использовать списковые структуры (динамические массивы). Различают два вида списков:

1) списки элементов по цепям;
2) списки цепей по элементам.

Список элементов по цепям представляется последовательностью пар чисел, первое из которых описывает элемент схемы, подключенный к некоторой цепи, а второе число задает номер контакта этого элемента, при помощи которого он подключен к цепи. Последовательность элементов списка, отделенная знаком “**;**” (точка с запятой), описывает одну цепь схемы. Список элементов по цепям для схемы рис. 2.1 имеет вид:

$$\overbrace{(0,1),(1,1)}^{v_1};\ \overbrace{(1,3),(3,1),(2,1)}^{v_2};\ \overbrace{(1,2),(3,2)}^{v_3};\ \overbrace{(2,2),(0,2)}^{v_4}$$

Наиболее удобным является список цепей по элементам, из которого сразу видно какие цепи подключены к элементу, а именно: каждая из пар чисел первым числом описывает номер контакта некоторого элемента, а вторым – номер цепи, к которой подключен элемент при помощи данного контакта. Последовательность элементов списка, отделенная знаком “**;**” (точка с запятой), описывает один элемент схемы. Список цепей по элементам для схемы рис. 2.1 имеет вид:

$$\overbrace{(1,1),(2,4)}^{e_0};\ \overbrace{(1,1),(2,3),(3,2)}^{e_1};\ \overbrace{(1,2),(2,4)}^{e_2};\ \overbrace{(1,2),(2,3)}^{e_3}$$

ГКС может быть описан с помощью специальных языковых средств:

L 1 2 1(1), 3(2), 2(1),

где ***L*** – признак элемента; далее, не менее чем через один пробел, следуют номер элемента – 1, тип элемента – 2, а также список цепей, в котором через запятую перечисляются номера цепей, инцидентных данному элементу, с указанием в скобках за номерами цепей, номеров контактов, подключающих цепь к элементу.

Для описания ГКС можно использовать матрицы инцидентности (***A*** и ***B***), определяющих множество ребер типа ***F*** («контакт–цепь») и типа ***W*** («элемент–контакт»).

$$\boldsymbol{A} = \left\| \boldsymbol{a_{ij}} \right\|_{\boldsymbol{l*k}},$$

где $\boldsymbol{a_{ij}} = \begin{cases} 1, \text{ если цепь } v_i \text{ инцидентна контакту } k_j, \\ 0 \text{ в противном случае.} \end{cases}$

$$\boldsymbol{B} = \left\| \boldsymbol{b_{ij}} \right\|_{\boldsymbol{(n+1)*k}},$$

где $\boldsymbol{b_{ij}} = \begin{cases} 1, \text{ если элемент } e_i \text{ инцидентен контакту } k_j, \\ 0 \text{ в противном случае.} \end{cases}$

Для схемы рис. 2.1 матрица ***A*** будет иметь следующий вид:

		k_{01}	k_{02}	k_{11}	k_{12}	k_{13}	k_{21}	k_{22}	k_{31}	k_{32}
A =	v_1	1	0	1	0	0	0	0	0	0
	v_2	0	0	0	0	1	1	0	1	0
	v_3	0	0	0	1	0	0	0	0	1
	v_4	0	1	0	0	0	0	1	0	0

Для схемы рис. 2.1 матрица ***B*** будет иметь следующий вид:

		k_{01}	k_{02}	k_{11}	k_{12}	k_{13}	k_{21}	k_{22}	k_{31}	k_{32}
B =	e_0	1	1	0	0	0	0	0	0	0
	e_1	0	0	1	1	1	0	0	0	0
	e_2	0	0	0	0	0	1	1	0	0
	e_3	0	0	0	0	0	0	0	1	1

ГКС может быть описан с помощью таблицы контактов:

$$\boldsymbol{T} = \left\| \boldsymbol{t_{ij}} \right\|_{\boldsymbol{(n+1)*\rho_{max}}},$$

где $\boldsymbol{t_{ij}} = \begin{cases} s, \text{ если контакт } k_j \text{ подключает элемент } e_i \text{ к цепи } v_s, \\ \text{отсутствует в противном случае.} \end{cases}$

ρ_{max} – наибольшее число контактов у одного элемента.

Таблица контактов для примера рис. 2.1 выглядит следующим образом:

	1	**2**	**3**
e_0	1	4	
e_1	1	3	2
e_2	2	4	
e_3	2	3	

ГКС является наиболее точной моделью и обычно используется при решении задач трассировки и размещения разногабаритных элементов.

Гиперграф

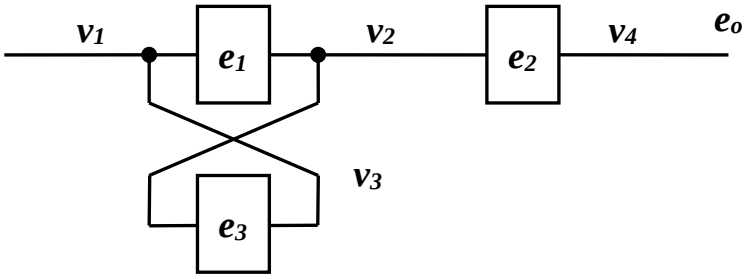
Гиперграф – это такое обобщение графа, в котором каждому ребру соответствует не пара вершин как в обычном графе, а произвольное подмножество вершин.

Данная модель обычно используется при решении задачи компоновки. Это связано с тем, что при компоновке контакты всегда распределяются в блок вместе с элементами, которым они инцидентны. Поэтому на ГКС можно исключить множество элементарных ребер ***W*** и ***F***. При этом каждая цепь, описываемая множеством контактов будет описываться множеством инцидентных ей элементов.

Гипеграф – это граф вида $H=(E,V)$, где E – множество вершин, соответствующее множеству элементов, V – множество ребер, при этом каждое ребро представляет подмножество элементов цепи, инцидентных V_i.

Для схемы примера (рис. 2.1) эти подмножества имеют вид:

$$U(v_1)=\{e_0,e_1\} \qquad U(v_3)=\{e_1,e_3\}$$
$$U(v_2)=\{e_1,e_2,e_3\} \qquad U(v_4)=\{e_2,e_0\}$$



ГГ для схемы примера (рис. 2.1) представлен на рисунке 2.3.

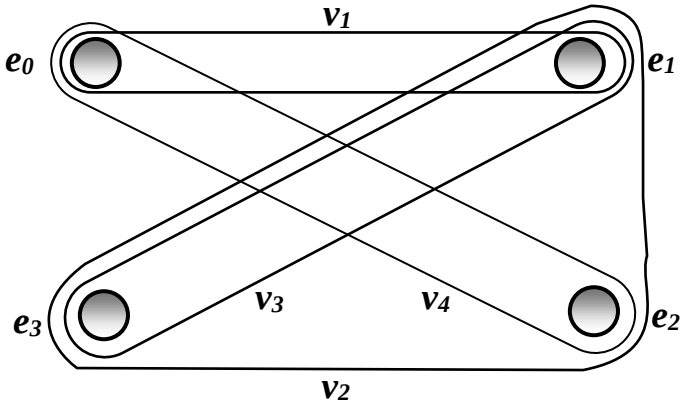


Рис. 2.3

ГГ может быть описан с помощью матрицы инцидентности ***H***:

$$H=\|h_{ik}\|_{(n+1)\times l},$$

где $h_{ik}=\begin{cases}1, \text{ если элемент } e_i \text{ инцидентен цепи } v_k,\\ 0 \text{ в противном случае.}\end{cases}$

Для схемы примера (рис. 2.1) матрица инцидентности имеет вид:

		v_1	v_2	v_3	v_4
H =	e_0	1	0	0	1
	e_1	1	1	1	0
	e_2	0	1	0	1
	e_3	0	1	1	0

Более удобно описывать гиперграф с помощью списка цепей по элементам и списка элементов по цепям.

Список цепей по элементам представляется последовательностью чисел, определяющих номера цепей, причем подпоследовательность чисел, выделенная знаком “**;**” (точка с запятой), содержит номера цепей инцидентных одному элементу схемы.

e_0 e_1 e_2 e_3

1,4; 1,2,3; 2,4; 2,3

Ц Ц

Список элементов по цепям каждой подпоследовательностью чисел задает номера элементов, инцидентных некоторой цепи.

v_1 v_2 v_3 v_4

0,1; 1,2,3; 1,3; 0,2

Э Э

Вместо одного списка элементов по цепям с разделителями (“**;**”) можно использовать два списка ***SP*** и ***RSP*** вида:

SP = 0, 1, 1, 2, 3, 1, 3, 0, 2

RSP = 0, 2, 5, 7, 9

Аналогично для списка цепей по элементам:

ST = 1, 4, 1, 2, 3, 2, 4, 2, 3

RST = 0, 2, 5, 7, 9

При использовании пар списков ***SP–RSP*** или ***ST–RST*** в позициях начиная с ***RSP[i]*** (***RST[i]***) до ***RSP[i+1]-1*** (***RST[i+1]-1***) в списке ***SP*** (***ST***) определяются номера элементов (цепей), инцидентных ***i***–й(–му) цепи (элементу).

Примечание. При использовании приведенного выше правила необходимо полагать, что индексация элементов списков ***RSP*** и ***RST*** совпадает с индексацией соответствующих структур схемы, по которым происходит объединение (т.е. с индексацией цепей для ***RSP***, с индексацией элементов для ***RST***). Индексация списков ***SP*** и ***ST*** определяется из значений элементов списков ***RSP*** и ***RST*** (т.е. в данном случае индексы элементов списков ***RSP*** и ***RST*** лежат в диапазоне ***0..8***).

Пример: Пусть I=2. Имеем

RST(2)+1=6 (позиция)

RST(2+1)=7

Следовательно, в позициях 6 и 7 списка ST определяются номера цепей (2, 4), инцидентных элементу 2.

Взвешенный неориентированный граф

ВНГ это граф вида $G = (E, U)$, где E – множество вершин графа, соответствующих множеству элементов схемы; U – множество ребер графа, при этом каждое ребро $u_{ij} \in U$ взвешено некоторым числом c_{ij}, определяющим степень связности элементов e_i и e_j. Другими словами, c_{ij} – количество общих цепей, связывающих элементы e_i и e_j.

ВНГ для схемы примера (рис. 2.1) показан на рисунке 2.4.

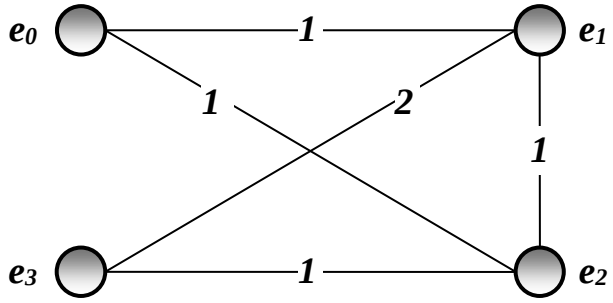


Рис. 2.4

ВНГ описывается с помощью матрицы смежности ***C***.

$$C = \left\| c_{ij} \right\|_{(n+1)\times(n+1)},$$

где c_{ij}-вес ребра.

Для схемы рис. 2.1 матрицы смежности ***C*** имеет вид:

		e_0	e_1	e_2	e_3
e_0		0	1	1	0
e_1		1	0	1	2
e_2		1	1	0	1
e_3		0	2	1	0

Часто вместо матрицы ***C*** используют матрицу ***C'***, которая отличается от ***C*** тем, что каждый диагональный элемент в ней не равен нулю, а равен числу цепей, инцидентных соответствующему элементу.

		e_0	e_1	e_2	e_3
e_0		2	1	1	0
e_1		1	3	1	2
e_2		1	1	2	1
e_3		0	2	1	2

Недостаток модели ВНГ – она является грубой и, чаще всего, используется в задачах размещения разногабаритных модулей.

§ 3. МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ КОМПОНОВКИ СХЕМ КОНСТРУКТИВНО УНИФИЦИРОВАННЫМИ МОДУЛЯМИ

Пусть имеется схема, состоящая из ***n*** элементов: $E = \{e_0, e_1, \ldots, e_n\}$, где e_0 – разъем.

Данную схему необходимо скомпоновать в ***r*** блоков: $B = \{b_1, b_2, \ldots, b_r\}$.

Обозначим через S_j вес – число элементов ***j***-го блока.

Ставится задача разбить схему на блоки так, чтобы суммарный вес каждого блока был бы не больше некоторой заданной величины $\overline{S}_j$, т.е. $S_j \le \overline{S}_j \quad (j=\overline{1, r})$, а число внешних выводов P_j каждого блока не превышало бы заданное $\overline{P}_j$, т.е. $P_j \le \overline{P}_j \quad (j=\overline{1, r})$. Если все блоки одинаковые, то $\overline{S}_j=\overline{S}$ и $\overline{P}_j=\overline{P}$.

При решении этой задачи необходимо выработать критерий разбиения:

1) минимальное число межблочных связей;
2) минимальное число внешних выводов каждого блока;

3) любой другой критерий.

Математическая постановка задачи компоновки зависит от выбора исходной математической модели схемы, оптимизирующего критерия качества и от учета тех или иных ограничений.

Математическая постановка задачи компоновки с использованием модели ВНГ

Критерий оптимизации – число межблочных связей. ВНГ будем представлять в виде матрицы C:

$$C = \|c_{ij}\|_{(n+1)\times(n+1)}$$

Введем в рассмотрение матрицу решений ξ, которая определяет вариант компоновки элементов схемы в блоки.

$$\xi = \|\xi_{ij}\|_{n*r}, \qquad (3.1)$$

где $\xi_{ij} = \begin{cases} 1, \text{ если элемент } e_i \text{ скомпонован в блок } b_j, \\ 0 \text{ в противном случае.} \end{cases}$

На элементы матрицы решений накладываются следующие ограничения:

$$\sum_{j=1}^{r} \xi_{ij} = 1 \text{ для } \forall\ i = \overline{1, n} \qquad (3.2)$$

Т.е. каждый элемент может быть расположен только в одном блоке.

$$\sum_{j=1}^{r} \xi_{ij} \le S_j \text{ для } \forall\ j = \overline{1, r} \qquad (3.3)$$

Число элементов в блоке должно быть не больше заданного.

Выведем формулу для числа внешних выводов блока.

Очевидно, что выражение $c_{ik}\xi_{ij}(1 - \xi_{ik})$ означает число межблочных связей, которые выходят из блока b_j от элемента e_i на элемент e_k ,расположенный вне блока b_j.

Очевидно, что выражение $\sum_{k=1}^{n} c_{ik}\xi_{ij}(1 - \xi_{ik}) + c_{i0}\xi_{ij}$ означает число межблочных связей, выходящих из блока b_j от элемента e_i на все остальные элементы схемы, расположенные вне блока b_j, включая элемент e_0.

Для того, чтобы получить общее число выводов от блока b_j, необходимо просуммировать предыдущее выражение по i:

$$\sum_{i=1}^{n}\left[\sum_{k=1}^{n} c_{ik}\xi i_j(1-\xi_{ik}) + c_{i0}\xi_{ij}\right] \le P_j \quad \text{äëÿ } \forall\ j = \overline{1, r} \qquad (3.4)$$

Теперь получим формулу для суммарного числа межблочных связей:

$$\underset{min}{L} = \frac{1}{2}\sum_{j=1}^{r}\left[\sum_{i=1}^{n}\sum_{k=1}^{n} c_{ik}\xi_{ij}(1-\xi_{ik})\right] + \sum_{j=1}^{r}\sum_{i=1}^{n} c_{i0}\xi_{ij} \qquad (3.5)$$

С учетом выражения (3.2) второй член выражения (3.5) перепишем в виде:

$\sum_{i=1}^{n} c_{i0}\sum_{j=1}^{r}\xi_{ij} = \sum_{i=1}^{n} c_{i0} = const$ – число связей с внешним разъемом схемы.

Поэтому в качестве можно взять первую часть целевой функции (3.5). Т.о., необходимо найти такую матрицу решений ξ, удовлетворяющую ограничениям (3.2)–(3.4), для которой обеспечивается минимум целевой функции (3.5). Данная задача

является задачей квадратичного целочисленного программирования с булевыми переменными c_{ij}.

В связи с тем, что модель ВНГ является грубой моделью, то и полученные формулы (3.4), (3.5) носят грубый характер.

Для примера рассмотрим схему, приведенную на рисунке 3.1.

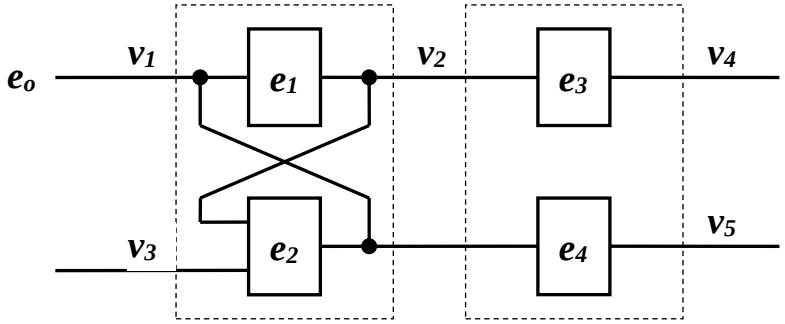


Рис. 3.1

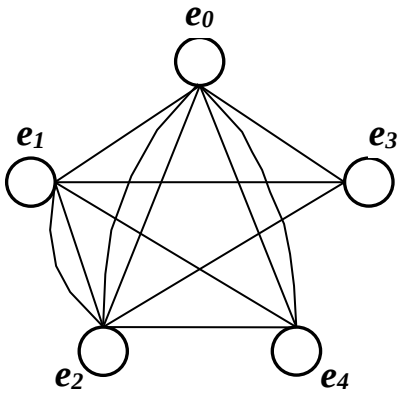


Рис. 3.2

ВНГ для данной схемы приведен на рисунке 3.2 (примечание: вес ребра на рисунке обозначен количеством связей).

Пусть требуется разбить приведенную схему на блоки с ограничениями:

$\overline{S} \le 2, \quad \overline{P} \le 4$.

Из рассмотрения ВНГ нетрудно видеть, что по данной модели невозможно разбить схему не только на два, но и на четыре блока. На самом же деле это не так (см. схему). Это говорит о неточности данной модели.

Математическая постановка задачи компоновки с использованием модели ГГ

Гиперграф опишем как $H = \|h_{ik}\|_{(n+1)\times l}$.

Рассмотрим ту же задачу, т.е. задачу нахождения матрицы решений ξ для данного варианта разбиения. На элементы матрицы ξ_{ij} накладываются те же ограничения (3.2 и 3.3), что и выше.

Отметим, что каждая цепь v_k в блоке b_j может потребовать либо один, либо ноль выводов. Для того чтобы цепь v_k потребовала один вывод необходимо и достаточно, чтобы хотя бы один элемент, инцидентный данной цепи, был бы расположен в блоке b_j, и хотя бы один элемент, инцидентный данной цепи, включая элемент e_0, был бы расположен вне блока b_j.

Математически вышесказанное записывается следующим образом.

$$\sum_{i=1}^{n} h_{ik}\xi_{ij} \ge 1 \qquad (3.6)$$

– сумма элементов, инцидентных цепи v_k и расположенных в блоке b_j.

$$\sum_{i=1}^{n} h_{ik}(1-\xi_{ij}) + h_{0k} \ge 1 \qquad (3.7)$$

– сумма элементов, инцидентных цепи v_k и расположенных вне блока b_j.

Введем в обозначения функцию: $\Psi(x) = \begin{cases} 1, & \text{при } x > 0; \\ 0, & \text{при } x \le 0. \end{cases}$

С учетом данного обозначения выражение

$$\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right)=1 \quad (3.8)$$

будет справедливо, если цепь v_k содержит хотя бы один элемент e_i в блоке b_j. Выражение

$$\Psi\left(\sum_{i=1}^{n} h_{ik}(1-\xi_{ij})+h_{0k}\right)=1 \quad (3.9)$$

справедливо, если цепь v_k содержит хотя бы один элемент вне блока b_j.

Число внешних выводов блока b_j, которые требуют цепи v_k, определяется из выражения:

$$\sum_{k=1}^{l}\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right)\Psi\left(\sum_{i=1}^{n} h_{ik}(1-\xi_{ij})+h_{0k}\right)\le \overline{P}_j \quad (3.10).$$

Теперь получим формулу для числа межблочных связей. Очевидно, что выражение

$$\sum_{j=1}^{r}\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right) \quad (3.11)$$

определяет количество блоков, в которых содержатся элементы, инцидентные цепи v_k. Количество межблочных связей, которые требует цепь v_k (без учета элемента e_0), определяется из выражения:

$$\sum_{j=1}^{r}\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right)-1 \quad (3.12).$$

Количество межблочных связей, которые требует цепь v_k с учетом внешнего разъема e_0, определяется из выражения:

$$\sum_{j=1}^{r}\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right)-1+h_{0k} \quad (3.13).$$

Для определения суммарного числа межблочных связей просуммируем выражение (3.13) по всем цепям v_k $\left(k=\overline{1,\ l}\right)$.

$$\underset{min}{L}=\sum_{k=1}^{l}\sum_{j=1}^{r}\Psi\left(\sum_{i=1}^{n} h_{ik}\xi_{ij}\right)-\underbrace{l+\sum_{k=1}^{l} h_{0k}}_{const} \quad (3.14)$$

Задача (3.14) это задача нелинейного целочисленного программирования с булевыми переменными.

Теперь вернемся к нашему примеру (рис. 3.1).

Матрица инцидентности (цепей) ГГ имеет вид:

$$H=\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ e_0 & 1 & 0 & 1 & 1 & 1 \\ e_1 & 1 & 1 & 0 & 0 & 0 \\ e_2 & 1 & 1 & 1 & 0 & 0 \\ e_3 & 0 & 1 & 0 & 1 & 0 \\ e_4 & 1 & 0 & 0 & 0 & 1 \end{array}$$

Т.о., элементы h_{ik} известны.

Матрица решений ξ для нашего примера имеет вид:

$$\begin{array}{c|cc|c|cc} & b_1 & b_2 & & & \\ e_1 & \xi_{11} & \xi_{12} & & 1 & 0 \end{array}$$

Т.о., элементы ξ_{ij} для

$$\xi = \begin{array}{c|cc|c|cc|} e_2 & \xi_{21} & \xi_{22} & = & 1 & 0 \\ e_3 & \xi_{31} & \xi_{32} & & 0 & 1 \\ e_4 & \xi_{41} & \xi_{42} & & 0 & 1 \end{array}$$

нашего варианта компоновки тоже известны.

Формула (3.13), примененная для каждой цепи v_k, показывает, что каждая из них требует только один вывод.

Формула (3.14) для нашего варианта разбиения (компоновки) дает следующее число межблочных связей: $L = \underset{v_1}{2} + \underset{v_2}{2} + \underset{v_3}{1} + \underset{v_4}{1} + \underset{v_5}{1} - \underset{l}{5} + \underset{\sum h_{0k}}{4} = 6$

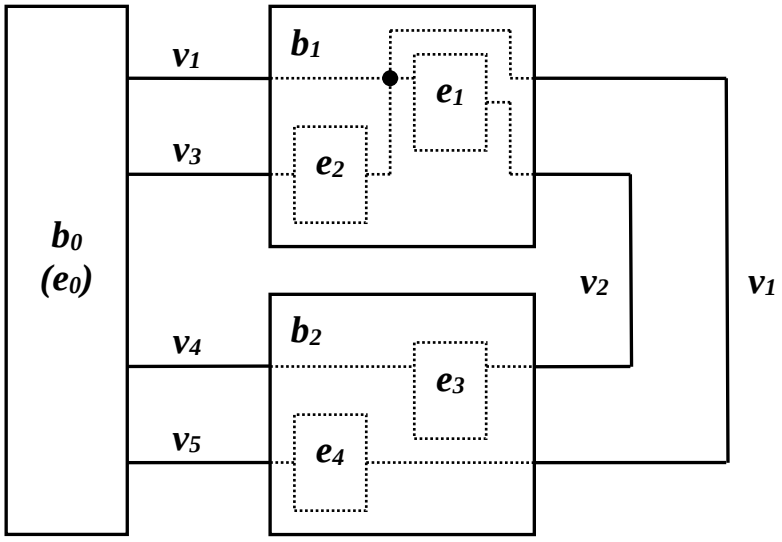


Рис. 3.3

Блочная организация схемы рисунка 3.1 при таком варианте размещения приведена на рисунке 3.3.

Общая характеристика алгоритмов компоновки конструктивных модулей

Все существующие алгоритмы компоновки конструктивных модулей можно разделить на два основных класса: точные и приближенные алгоритмы, которые, в свою очередь представлены несколькими подклассами (см. рис. 3.4).

Рис. 3.4 Классификация алгоритмов компоновки

Алгоритмы Лаурера основаны на сведении задачи компоновки к так называемой задаче покрытия, возникающей при минимизации булевых функций. Эти алгоритмы в качестве модели схемы используют гиперграф (матрицу ***H***).

Существует несколько модификаций алгоритмов на основе метода ветвей и границ, в которых в качестве модели схемы используются ГГ или ВНГ. Они отличаются способом вычисления оценки.

При современном быстродействии ЭВМ точные алгоритмы позволяют решать задачи с числом переменных не более 20-ти. В связи с этим на практике широко применяются приближенные, эвристические алгоритмы, математически слабо обоснованные, но дающие неплохие результаты. Суть последовательных алгоритмов – последовательное заполнение блоков еще не распределенными в блоки элементами. В качестве очередного элемента выбирается элемент максимально связанный с элементами, уже включенными в формируемый блок. Процесс заполнения блока элементами продолжается до тех пор, пока не нарушаются ограничения на модульную ($\overline{S}$) и контактную ($\overline{P}$) емкости блока. Имеются различные модификации таких алгоритмов, отличающиеся выбором критерия для кандидата на включение и способом вычисления оценки.

Суть параллельно-последовательных алгоритмов: здесь сначала выделяются некоторые группы элементов, например, сильно связанные между собой, а затем эти группы параллельно распределяются по блокам с учетом контактных и модульных ограничений.

Итерационные алгоритмы служат для улучшения некоторого начального варианта компоновки, полученного либо вручную, либо с помощью последовательных алгоритмов, и состоят в обмене (перестановках) элементов, принадлежащих разным блокам. При этом осуществляются одинарные, двойные и, в общем случае, групповые перестановки элементов.

§ 4. ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ КОМПОНОВКИ

Будем описывать схему с помощью ГГ: $H = (E, V)$

$E = \{e_0, e_1, \ldots, e_n\}$ - множество элементов схемы,

$V = \{v_1, v_2, \ldots, v_l\}$ - множество цепей схемы

Задача: требуется разбить схему на блоки $B = \{b_1, b_2, \ldots, b_r\}$ так, чтобы каждый блок содержал не более чем заданные число элементов $\overline{S}$ и число внешних выводов $\overline{P}$, т.е. $S_j \le \overline{S}$, $P_j \le \overline{P}$, $j = \overline{1,r}$.

В рассматриваемом алгоритме процесс формирования очередного блока начинается с выбора первого, так называемого базового элемента (БЭ). В качестве БЭ выбирается элемент, имеющий максимальное число общих цепей со всеми еще нераспределенными элементами (элемент e_0 уже распределен). Далее блок последовательно заполняется элементами из числа еще нераспределенных. Для этого из множества всех нераспределенных элементов выделяется подмножество таких элементов, распределение которых в блок не приводит к превышению заданного числа выводов из блока. Из них в блок распределяется элемент, имеющий максимальное число общих цепей со всеми элементами, уже распределенными в блок. Если таких элементов несколько, то среди них выбирается элемент, при добавлении которого в блок число выводов из блока минимально. Если и таких элементов несколько, то

выбирается элемент с максимальным порядковым номером (для формализации задачи).

Рассмотрим более подробно процедуру формирования блока b_j. Будем считать, что блоки b_1, b_2, … , b_{j-1} уже сформированы. Пусть I_j – множество элементов, нераспределенных в эти предшествующие блоки, т. е.

$$I_j = E \setminus \left[\left(\bigcup_{k=1}^{j-1} E(b_k) \right) \bigcup e_0 \right],$$

где $E(b_k)$ – множество элементов блока b_k.

Последовательный алгоритм компоновки можно представить в следующем виде.

П. 1.

При выборе БЭ для всех элементов $x \in I_j$ вычисляется оценка $L_1(x)$.

$$\max_{} L_1(x) = \sum_{v_k \in V(x)} q_k \quad (4.1),$$

где $q_k = \begin{cases} 1, \text{ если } E(v_k) \bigcap (I_j \setminus x) \neq \varnothing; \\ 0 \text{ в противном случае.} \end{cases}$

Здесь q_k – связи элемента со всеми нераспределенными элементами,

$V(x)$ – множество цепей, инцидентных элементу x,

$E(v_k)$ – множество элементов, инцидентных цепи v_k,

$L_1(x)$ – число связей элемента x со всеми еще неразмещенными элементами.

В блок распределяется элемент с максимальной оценкой $L_1(x)$. Если таких элементов несколько, то выбирается элемент с большим порядковым номером (для формализации задачи).

П. 2.

Пусть на t-м шаге формирования блока b_j имеется множество элементов $E(b_j^t)$, распределенных к этому шагу в блок b_j (b_j^t). При определении очередного элемента для всех нераспределенных элементов $x \in (I_j \setminus E(b_j^t))$ вычисляется оценка $L_2(x)$.

$$\min_{} L_2(x) = \sum_{v_k \in V(E(b_j^t) \bigcup x)} p_k \quad (4.2),$$

где $p_k = \begin{cases} 1, \text{ если } E(v_k) \bigcap \left[Ij \setminus E(b_j^t) \bigcup x \right] = \varnothing; \\ 0, \text{ в противном случае.} \end{cases}$

Здесь $V(E(b_j^t) \bigcup x)$ – множество цепей, инцидентных всем элементам блока b_j к шагу t, включая элемент x,

$L_2(x)$ – число выводов блока b_j на шаге t с учетом элемента x.

П. 3.

Для всех элементов x, для которых выполняется условие $L_2(x) \leq \overline{P}$, вычисляется оценка $L_3(x)$ – число связей элемента x с формируемым блоком b_j^t.

$$\max_{} L_3(x) = \sum_{v_k \in V(x)} q_k \quad (4.3),$$

где $q_k = \begin{cases} 1, \text{ если } E(v_k) \bigcap E(b_j^t) \neq \varnothing; \\ 0 \text{ в противном случае.} \end{cases}$

В блок b_j включается элемент с максимальной оценкой $L_3(x)$. Если таких элементов несколько, то среди них выбирается элемент с минимальной оценкой $L_2(x)$. Если и таких элементов несколько, то выбирается элемент с максимальным порядковым номером (для формализации задачи).

П. 4.

Если ни для одного элемента $x \in (I_j \setminus E(b_j^t))$ условие $L_2(x) \le \overline{P}$ не выполняется, то дополнение блока b_j прекращается и начинается формирование следующего блока (переход к п.1).

П. 5.

Алгоритм заканчивает работу, когда все элементы схемы распределены в блоки.

При практической реализации алгоритма вместо формулы (4.2) для определения числа выводов из блока удобно использовать формулы для приращения числа выводов из блока b_j при добавлении в него элемента x.

$$L_2(x) = L_2(b_j^t) + \Delta L_2(x) \qquad (4.4),$$

где $L(b_j^t)$ – количество выводов блока b_j до добавления в него элемента x.

$$\Delta L_2(x) = \sum_{v_k \in V(x)} \Delta p_k \qquad (4.5),$$

где $\Delta p_k = \begin{cases} -1, \text{ если } (A \neq \varnothing) \& (B = \varnothing); \\ 0, \text{ если } (A \neq \varnothing) \& (B \neq \varnothing); \\ 1, \text{ если } A = \varnothing. \end{cases}$

Здесь $A = E(v_k) \bigcap E(b_j^t)$; $\quad B = I_j \setminus (E(b_j^t) \bigcup x)$.

Последнее выражение (для Δp_k) можно пояснить с помощью рисунка 4.1.

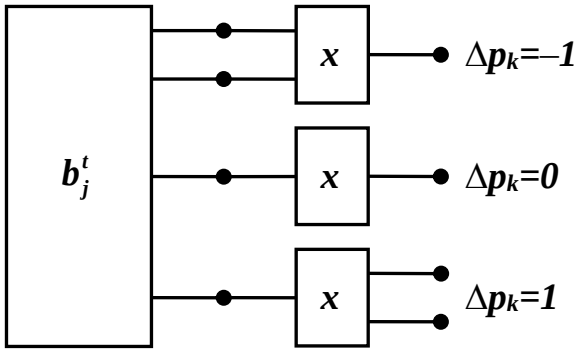


Рис. 4.1.

Рассмотрим работу последовательного алгоритма компоновки на примере.

Пусть необходимо разбить схему, приведенную на рисунке 4.2 на типовые блоки с числом элементов $\overline{S} \le 3$ и числом выводов $\overline{P} \le 6$, минимизируя при этом количество межблочных связей.

Перед началом компоновки множество нескомпонованных элементов I_1 содержит все элементы схемы кроме e_0 (элемент e_0 уже распределен).

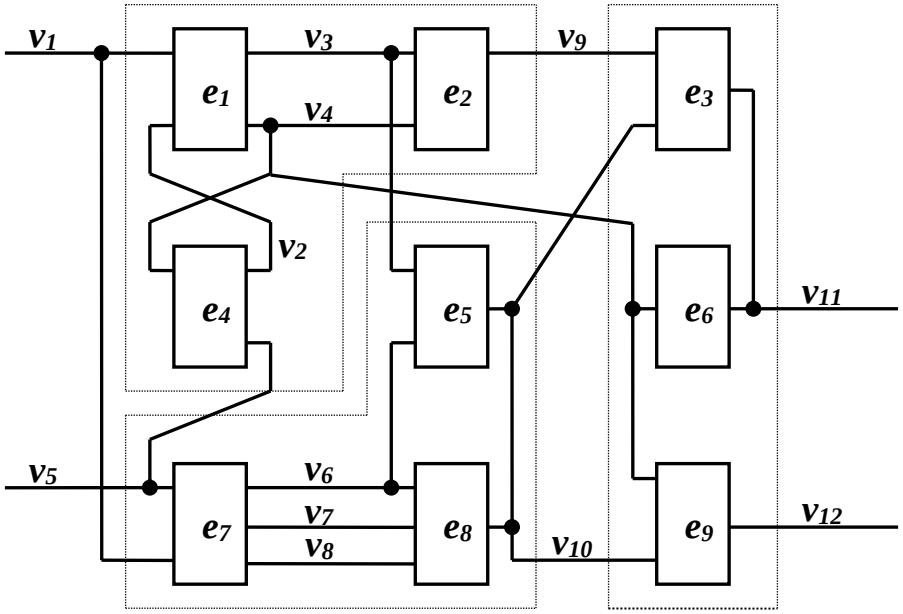


Рис. 4.2.

Для выбора БЭ вычислим оценку $L_1(x)$ по формуле (4.1) для всех элементов множества I_1.

$$I_1 = \{e_1, e_2, \ldots, e_9\}$$

$$V(e_1) = \{v_1, v_2, v_3, v_4\}$$

$$E(v_1) = \{e_1, e_4, e_7\}$$

$$E(v_2) = \{e_1, e_4\}$$

$$E(v_3) = \{e_1, e_2, e_5\}$$

$$E(v_4) = \{e_1, e_2, e_4, e_6, e_9\}$$

В соответствии с формулой (4.1) $L_1(e_1) = q_1 + q_2 + q_3 + q_4 = 1 + 1 + 1 + 1 = 4$.

Вычисляя подобным образом оценки $L_1(x)$ для элементов $e_2, e_3, \ldots, e_9$ определим, что максимальным числом связей со всеми еще неразмещенными элементами обладает элемент e_7 – $L_{1max} = L_1(e_7) = 5$.

Аналогично вычисляются оценки $L_2(x)$ и $L_3(x)$. Вычислим, например, $L_3(e_9)$ – связи элемента e_9 с элементами формируемого блока, в который вошли элементы e_7 и e_8 (см. ниже).

$$V(e_9) = \{v_4, v_{10}, v_{12}\}$$

$$E(v_4) = \{e_1, e_2, e_4, e_6, e_9\}$$

$$E(v_{10}) = \{e_3, e_5, e_8, e_9\}$$

$$E(v_{12}) = \{e_9\}$$

Таким образом $L_3(e_9) = q_4 + q_{10} + q_{12} \Big|_{b_1^1 = \{e_7, e_8\}}^{\text{ï ðè}} = 0 + 1 + 0 = 1$.

Результаты вычислений удобно свести в таблицу.

	L_1	L_2	L_2	L_3	L_1	L_2	L_1	L_2	L_3
1	2	3	4	5	6	7	8	9	10
e_1	**4**	**5+3>6**	**4+3>6**	—	**4***	—	—	—	—
e_2	**3**	**5+3>6**	**4+3>6**	—	**3**	**4+1**	**2**	**4+1**	**2***
e_3	**3**	**5+3>6**	**4+2=6**	**1**	**3**	**4+3**	—	**4+3**	—

e_4	4	5+2>6	4+2=6	2	3	4+0	3*	—	—
e_5	3	5+2>6	4+0<6*	2	—	—	—	—	—
e_6	2	5+2>6	4+2=6	0	2	4+1	1	4+1	1
e_7	5*	—	—	—	—	—	—	—	—
e_8	4	5-1<6*	—	—	—	—	—	—	—
e_9	2	5+3>6	4+2=6	1	2	4+2	1	4+2	1

В качестве БЭ блока b_1 выбираем элемент e_7 с максимальной оценкой $L_1(x)=5$.
Для остальных элементов схемы вычисляем оценку $L_2(x)$ – выводы. Результаты сведем в графу 3 таблицы.
Всего один элемент e_8 может быть включен в блок b_1 без нарушения контактного ограничения $\overline{P} \le 6$. Таким образом, $b_1^1 = \{e_7, e_8\}$, количество выводов из блока – четыре.
Для выбора очередного элемента блока b_1 вычислим оценки $L_2(x)$ для нескомпонованных элементов (графа 4 таблицы).
Для элементов ($\overline{P} \le 6$) e_3, e_4, e_5, e_6, e_9 вычислим оценки $L_3(x)$ – связи (графа 5 таблицы).
При одинаковых максимальных оценках $L_{3max}=2$ элемент e_5 дает меньшее число выводов из блока (четыре) чем элемент e_4, поэтому состав блока $b_1=\{e_7, e_8, e_5\}$; $P_1=4$, $S_1=3$.
Формирование второго блока проведем аналогично первому (графы 6 – 10).
Состав $b_2=\{e_1, e_4, e_2\}$.
Оставшиеся элементы распределим в блок b_3 с $P_3=6$.
Количество межблочных связей при данном варианте компоновки – десять.

При решении задачи компоновки на ПЭВМ целесообразно использовать списки элементов по цепям и списки цепей по элементам с соответствующими разделителями (см. § 2, Гиперграф).

Примечания: вычисление оценок $L_1(x)$ и $L_2(x)$ для примера:

$x_1 = I_1$

$I_1 = e_1 - e_9,\ V(e_1) = v_1, v_2, v_3, v_4$

$e(V_1) = e_1, e_4, e_7$

$e(V_2) = e_1, e_4$

$e(V_3) = e_1, e_2, e_5$

$e(V_4) = e_1, e_4, e_6, e_9$

$q_1 = q_2 = q_3 = q_4 = 1+1+1+1 = 4$

$L_{1\max}(e_7) = 5^*$ и т.д.

Для схемы рис. 4.2 список элементов по цепям и соответствующий список разделителей имеют следующий вид:

$$\overbrace{e_0, e_1, e_4, e_7}^{v_1}, \overbrace{e_1, e_4}^{v_2}, \overbrace{e_1, e_2, e_5}^{v_3}, \overbrace{e_1, e_2, e_4, e_6, e_9}^{v_4}, \overbrace{e_0, e_4, e_7}^{v_5}, \overbrace{e_5, e_7, e_8}^{v_6}, \overbrace{e_7, e_8}^{v_7}, \overbrace{e_7, e_8}^{v_8},$$

$$\overbrace{e_2, e_3}^{v_9}, \overbrace{e_3, e_5, e_8, e_9}^{v_{10}}, \overbrace{e_0, e_3, e_6}^{v_{11}}, \overbrace{e_0, e_9}^{v_{12}}$$

$$RSM = 0,4,6,9,14,17,20,22,24,26,30,33,35$$

В позиции ***RSM(I)+1*** до ***RSM(I+1)*** в списке S_n определяются номера элементов, инцидентных ***i***-й цепи ***(I)***.

$RSM(1) = 0, RSM(2) = 4, \mathrm{RSM}(3) = 6, \mathrm{RSM}(4) = 9$ и т.д.

Пусть ***I=3, RSM(3)+1=6+1=7*** – позиция, ***RSM(4)=9*** – позиция.

Следовательно, в позициях 7-9 списка S_n определяется номер элементов, инцидентных V_3. Аналогично составляются списки цепей по модулям и списки разделителей (RS_n). Схема последовательного алгоритма компоновки приведена на рисунке 4.3.

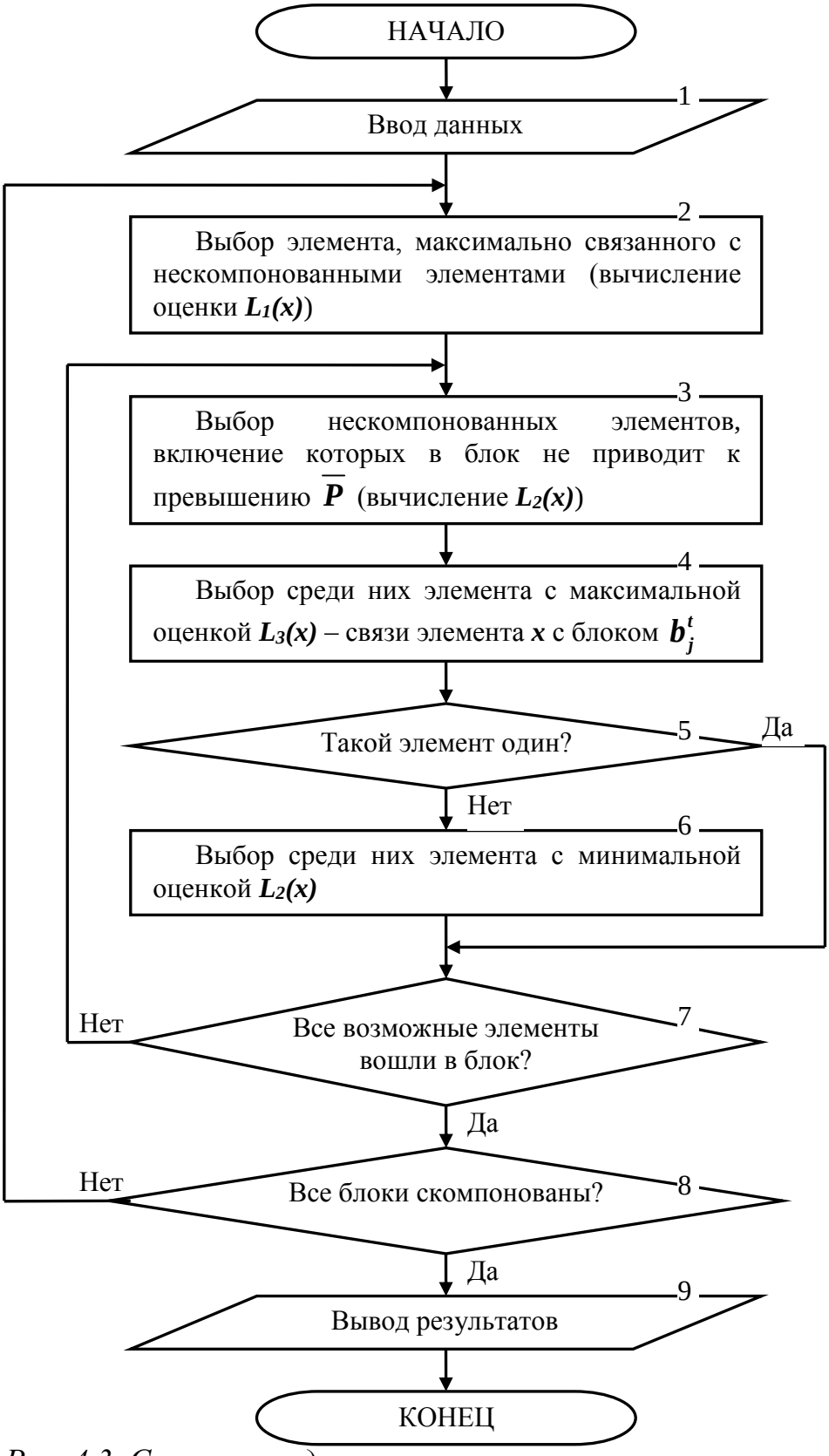


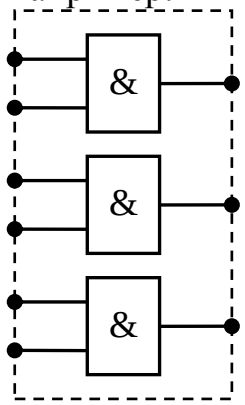
Рис. 4.3. Схема последовательного алгоритма компоновки

§ ЗАДАЧА ПОКРЫТИЯ СХЕМ НАБОРОМ КОНСТРУКТИВНЫХ МОДУЛЕЙ.

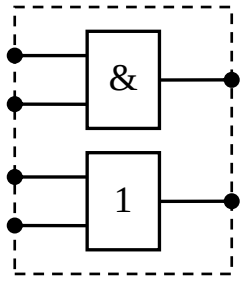
В результате решения задачи покрытия каждый элемент схемы привязывается к некоторому конструктивному модулю из набора $Q=\{Q_1, Q_2, \ldots, Q_m\}$.

Математическая постановка задачи покрытия зависит от типа конструктивных модулей набора. Различают 3 типа конструктивных модулей:

1) Каждый модуль содержит базовые логические элементы одного типа, при этом все входы и выходе логических элементов являются выводами модуля.
   Например:

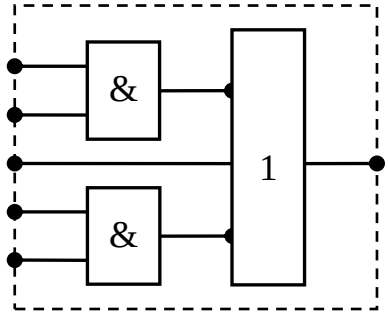


2) То же, что и предыдущий тип, однако в одном конструктивном модуле могут содержаться разнотипные логические элементы.



   Модули 1-го и 2-го типов называются модулями с несвязанными элементами.

3) Конструктивный модуль в общем случае содержит связанные между собой разнотипные элементы и не все входные и выходные выводы элементов имеются на внешних контактах конструктивного модуля.

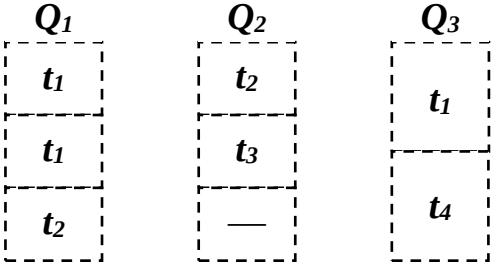


Рассмотрим математическую постановку задачи покрытия схемы конструктивными модулями с несвязанными элементами (1-го и 2-го типов).

Обозначим через b_i количество логических элементов t_i типа, содержащихся в функционально-логической схеме (ФЛС). При таком подходе состав ФЛС может быть описан матрицей-столбцом: $B=\|b_i\|_n$, где n – число типов логических элементов схемы, b_i – число логических элементов t_i типа в ФЛС. Будем считать, что каждый тип конструктивного модуля Q_j в общем случае содержит несколько типов t_i базовых элементов, а весь набор конструктивных модулей Q может быть описан матрицей

$A=\|a_{ij}\|_{nxm}$, где a_{ij} – количество базовых логических элементов типа t_i в конструктивном модуле типа Q_j ($i=\overline{1,n}$; $j=\overline{1,m}$).

Пример.



t_1 – 2ИЛИ-НЕ;
t_2 – 3ИЛИ-НЕ;
t_3 – простейший RS-триггер;
t_4 – JK-триггер.

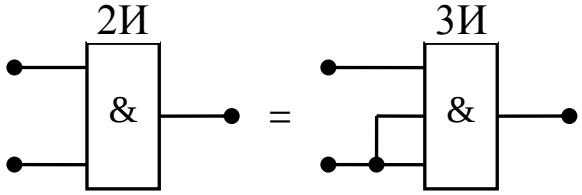
A =	Q_1	Q_2	Q_3
t_1	2	0	1
t_2	1	1	0
t_3	0	1	0
t_4	0	0	1

В качестве критерия оптимизации будем использовать минимум стоимости покрытия. Обозначим c_j стоимость конструктивного модуля Q_j.

Пусть для покрытия ФЛС необходимо выделить y_j конструктивных модулей Q_j, тогда в качестве целевой функции можно взять функцию вида:

$$\underset{min}{J}=\sum_{j=1}^{m} y_j c_j \qquad (*).$$

Будем считать, что каждый тип базового логического элемента схемы может быть реализован как базовый логический элемент того же набора, так и БЛЭ другого типа набора. Например:



Обозначим через x_{ki} количество логических элементов типа t_k, которое идет на покрытие логических элементов схемы типа t_i. Возможность замены БЛЭ типа t_k на БЛЭ типа t_i будем описывать матрицей $W=\|w_{ki}\|_{nxn}$,

$$w_{ki}=\begin{cases}1, & \text{если БЛЭ типа } t_k \text{ может быть заменен на БЛЭ типа } t_i\\ 0 & \text{в противном случае.}\end{cases}$$

Очевидно, что $x_{ki}>0$ если $w_{ki}=1$.

Учитывая введенные обозначения, задачу покрытия можно сформулировать следующим образом: если y_j количество требуемых конструктивных модулей типа Q_j, а в каждом Q_j конструктивном модуле содержится a_{ij} элементов типа t_i, то $\sum_{j=1}^{m} a_{ij} y_j$ определяет количество логических элементов типа t_i, содержащихся во всем выделенном для покрытия ФЛС наборе конструктивных модулей.

Очевидно, что для покрытия всей ФЛС необходимо выполнение следующего условия:

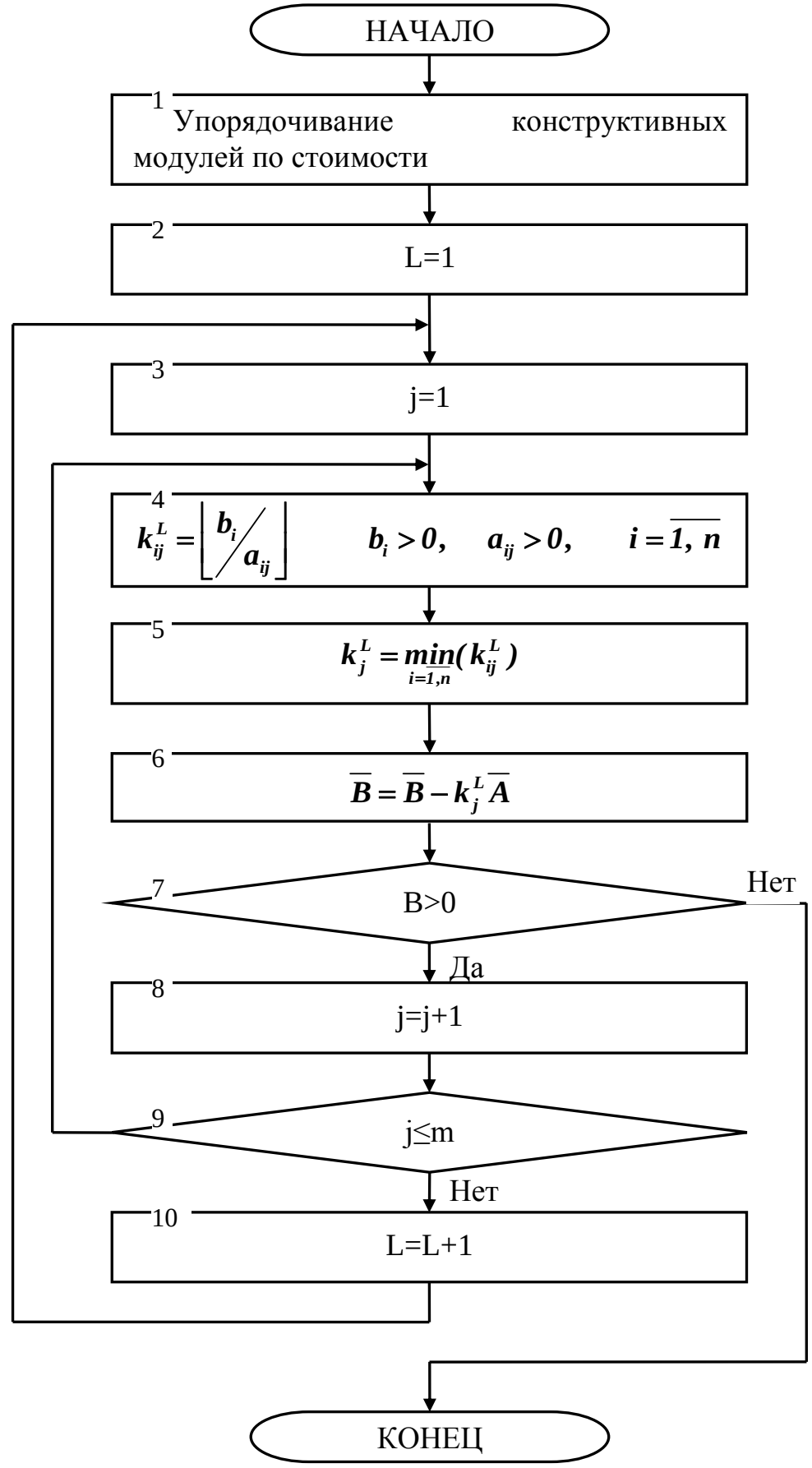
$$\sum_{j=1}^{m} a_{ij}+\sum_{k=1}^{m} x_{ki}-\sum_{k=1}^{m} x_{ik}\geq b_i \qquad (**),$$

где второе слагаемое – количество логических элементов других типов, которые идут в покрытии на реализацию логических элементов типа t_i; последнее слагаемое – количество логических элементов типа t_i, которые идут на реализацию логических элементов других типов.

Т. о., ставится задача отыскания таких значений y_j и x_{ki}, удовлетворяющих (**), чтобы обеспечивался минимум целевой функции (*) – стоимости покрытия. На практике используются приближенные методы, опирающиеся на специфику используемых конструктивных модулей. Если имеем дело с несвязанными элементами конструктивного модуля, то здесь в результате определения требуемого числа модулей не осуществляется привязка логических элементов схемы к конструктивным модулям, отсюда появляется возможность оптимизации по другому критерию, в частности по критерию минимума межблочных связей.

Пример $Q=\{Q_1, Q_2, Q_3\}$, $t=\{t_1, t_2, t_3, t_4\}$ приведен нами выше. Требуется покрыть ФЛС, описываемую вектором $B=\{5, 4, 1, 1\}$, заданным набором конструктивных модулей.

Блок-схема алгоритма определения требуемого числа конструктивных модулей по критерию минимума стоимости покрытия приведена на рисунке.


НАЧАЛО
1
Упорядочивание конструктивных модулей по стоимости
2
L=1
3
j=1
4
$k_{ij}^{L} = \left\lfloor b_i / a_{ij} \right\rfloor$ $b_i > 0,$ $a_{ij} > 0,$ $i = \overline{1, n}$
5
$k_j^L = \min_{i=\overline{1,n}}(k_{ij}^L)$
6
$\overline{B} = \overline{B} - k_j^L \overline{A}$
7
B>0
Нет
Да
8
j=j+1
9
j≤m
Нет
10
L=L+1
КОНЕЦ

Пример:

$$\overline{B} = \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix}\begin{Vmatrix} 5 \\ 4 \\ 1 \\ 1 \end{Vmatrix} \qquad A = \begin{matrix} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}\begin{matrix} Q_1 & Q_2 & Q_3 \\ \end{matrix}\begin{Vmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{Vmatrix}$$

$\overline{A_1} = \{2,1,0,0\}, \overline{A_2} = \{0,0,1,0\}, \overline{A_3} = \{1,0,0,1\}$

Решение:

Для нашего примера:

$$\lambda = 1, \mathrm{j} = 1, \mathrm{i} = 1, k_{ij}^{\lambda} = k_{11}^{1} = \frac{b_1}{a_{11}} = \left[\frac{5}{2}\right] = 2$$

$$\lambda = 1, \mathrm{j} = 1, \mathrm{i} = 2, k_{ij}^{\lambda} = k_{21}^{1} = \frac{b_2}{a_{21}} = \left[\frac{4}{1}\right] = 4$$

Так как $a_{31}=0$ и $a_{41}=0$, то в качестве коэффициента $k_1^1 = \min(k_{11}^1, k_{21}^1) = 2$

$\overline{B} = \overline{B} - k_j^{\lambda} \cdot \overline{a}_j = \{5,4,1,1\} - \{4,2,0,0\} = \{1,2,1,1\} > 0$

Т.к. $\overline{B} > 0$, полагаем, что j=2 и вычисляем

$k_{22}^1 = 4$, $k_{32}^1 = 1$ и, следовательно, $k_2^1 = \min(4,1) = 1$

$$\overline{B} = \{1,2,1,1\} - \{0,1,1,0\} = \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 1 \end{Vmatrix} > 0$$

Полагаем j=3 $\Rightarrow k_3^1 = 1$

$\overline{B} = \overline{B} - k_3^1 \cdot a_3 = \{0,1,0,0\} > 0$

Переходим к $\lambda = 2$:

$k_{21}^2 = 1$, (остальные b_i=0) $\Rightarrow k_1^2 = 1$, $\overline{B} = \overline{B} - k_1^2 \cdot a_1 < 0$

Т.о. имеем $k_1^1 = 2$, $k_2^1 = 1$, $k_3^1 = 1$, $k_1^2 = 1$.

Требуемое число корпусов j-го типа определяется по формуле:

$y_j = \sum_{\lambda} k_j^{\lambda}$

Эти данные подставляем в выражение (1) и определяем:

$y_1 = k_1^1 + k_1^2 = 2 + 1 = 3,$

$y_2 = k_2^1 = 1,$

$y_3 = k_3^1 = 1.$

После определения требуемого числа корпусов, можно осуществить привязку элементов к корпусам. Для этого могут быть использованы рассмотренные ранее алгоритмы решения задач компоновки (задачи разбиения).

§ 5. ЗАДАЧА РАЗМЕЩЕНИЯ КОНСТРУКТИВНЫХ МОДУЛЕЙ

Различают два типа задач размещения:

1) размещение конструктивных элементов в заранее фиксированные позиции;

2) размещение элементов в непрерывном монтажном пространстве, когда позиции заранее не определены, а определяются в процессе размещения (например, проектирование БИС).

Рассмотрим первую задачу.

Пусть имеется регулярное монтажное пространство с уже фиксированными позициями $P = \{ p_1, p_2, \dots, p_m \}$, а также имеется количество $n \le m$ элементов для размещения.

Будем считать, что длина связей определяется расстоянием между геометрическими центрами соответствующих позиций, т. е.

$$d_{kq} = \sqrt{(x_k - x_q)^2 + (y_k - y_q)^2} \qquad (5.\text{à})$$

Чаще всего такая метрика используется тогда, когда последующая трассировка соединений осуществляется с помощью проводного монтажа «в навал» (провода с изоляцией). При использовании печатного или жгутового монтажа используются соответственно метрики (5.б) и (5.в):

$$d_{kq} = |x_k - x_q| + |y_k - y_q| \qquad (5.\text{á})$$

$$d_{kq} = |x_k - x_q| t + |y_k - y_q| t \qquad (5.\text{â})$$

где t – количество проводников в жгуте.

Метрики (5.б) и (5.в) обычно используют тогда, когда наряду с минимизацией суммарной взвешенной длины связей стараются минимизировать также длину наиболее длинной связи (она определяет время задержки схемы).

Если для простоты рассуждений положить $m=n$, то вариантов размещения будет $n!$, при этом любой вариант размещения может быть задан перестановкой $\pi = \{ \pi(1), \pi(2), \dots, \pi(m) \}$, где $\pi(i)$ – номер позиции, в которую устанавливается элемент e_i.

Сформулируем математическую постановку задачи размещения.

Пусть $L(\pi)$ – суммарная длина межэлементных соединений, соответствующая некоторому варианту размещения π.

Пусть E_S – множество директивно размещенных элементов, к числу которых, в частности, относятся разъемы и внешние контактные площадки. Здесь S – множество индексов директивно размещенных элементов.

Рассмотрим некоторый произвольный заранее не размещенный элемент. Определим суммарную длину его связей со всеми директивными элементами. Обозначим эту длину как:

$$l_{i\pi(i)} = \sum_{s \in S} c_{is} d_{\pi(i)\pi(s)} \qquad (5.1)$$

где $\pi(i)\pi(s)$ – расстояние между i и s,

c_{is} – элемент матрицы смежности ВНГ,

$d_{\pi(i)\pi(s)}$ – расстояние между $\pi(i)$ и $\pi(s)$.

Суммарная взвешенная длина межсоединений недирективно размещенных элементов e_i и e_j определяется как:

$$c_{ij} d_{\pi(i)\pi(j)} \qquad (5.2)$$

С учетом (5.1) и (5.2) суммарная взвешенная длина межсоединений для варианта размещения π будет определяться из выражения (5.3):

$$\underset{min}{L(\pi)} = \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} c_{ij} d_{\pi(i)\pi(j)} + \sum_{i=1}^{n} l_{i\pi(i)} \qquad (5.3)$$

Таким образом, необходимо найти такой вариант размещения $\pi^* = \{\pi^*(1), \pi^*(2), \ldots, \pi^*(m)\}$ при котором обеспечивается минимум целевой функции (5.3). Это комбинаторная задача размещения.

Теперь рассмотрим постановку задачи целочисленного программирования.

Пусть перестановочная матрица (матрица решений) имеет вид:

$$X = \|x_{ik}\|_{n*n}, \qquad (5.4)$$

где $x_{ik} = \begin{cases} 1, \text{ если элемент } e_i \text{ размещен в позиции k,} \\ 0 \text{ в противном случае.} \end{cases}$

На элементы матрицы решений X можно наложить следующие ограничения:

$$\sum_{i=1}^{n} x_{ik} = 1 \text{ для } \forall\, k = \overline{1,n} \qquad (5.5)$$

Т.е. каждому элементу соответствует одна позиция.

$$\sum_{k=1}^{n} x_{ik} = 1 \text{ для } \forall\, i = \overline{1,n} \qquad (5.6)$$

Каждый элемент может занимать только одну позицию.

При таком подходе суммарная взвешенная длина межсоединений может быть представлена следующим образом:

$$\underset{min}{L(X)} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{q=1}^{n} c_{ij} d_{kq} x_{ik} x_{jq} + \sum_{i=1}^{n}\sum_{k=1}^{n} l_{ik} x_{ik} \qquad (5.7)$$

В выражении (5.7) первый член это суммарная взвешенная длина межсоединений между собой недиректив но размещенных элементов, второй член – суммарная взвешенная длина межсоединений между недиректив но размещенными и директив но размещенными элементами.

Необходимо найти такую булеву матрицу решений X, которая удовлетворяла бы ограничениям (5.4) и (5.6) и обеспечивала минимум целевой функции (5.7). Эта задача является квадратичной задачей целочисленного программирования.

В случае когда элементы не связаны между собой и заранее не фиксированы, а связаны лишь с директив но размещенными элементами, соответствующая задача является задачей линейного назначения. Этот факт используется в ряде приближенных алгоритмов размещения.

В настоящее время разработано много алгоритмов размещения, классификация которых приведена на рисунке 5.1.

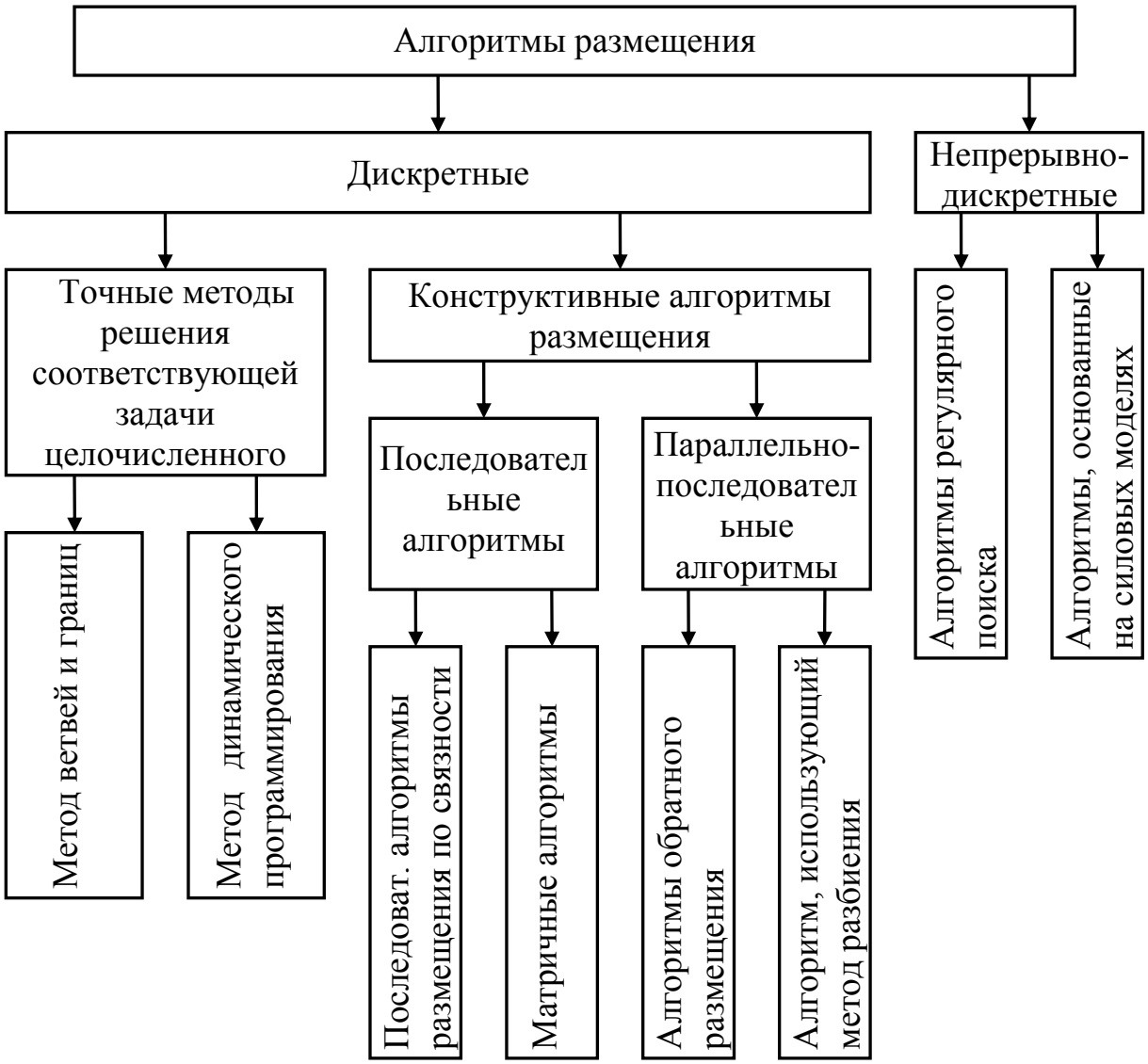


Рис. 5.1. Классификация алгоритмов размещения.

§ 6. КОНСТРУКТИВНЫЕ АЛГОРИТМЫ РАЗМЕШЕНИЯ

Среди конструктивных алгоритмов размещения выделяют последовательные и параллельно-последовательные алгоритмы.

В последовательном алгоритме используется ***n***–шаговый процесс принятия решения. На каждом шаге здесь размещается один элемент. В параллельно-последовательном алгоритме на каждом шаге размещается группа элементов (или даже все элементы).

Среди последовательных алгоритмов различают последовательные алгоритмы размещения по связности и матричные алгоритмы.

Последовательные алгоритмы размещения по связности

Сущность этого многошагового алгоритма сводится к последовательному размещению очередного модуля (элемента) в определенный узел платы. Предполагается, что часть модулей (или хотя бы один) заранее размещены на монтажной плоскости. В качестве таких модулей могут быть выбраны либо контакты разъема, либо модули, фиксированные в определенных позициях в соответствии с директивными указаниями разработчика. При выборе очередного модуля оптимизируется целевая функция, учитывающая связи этого модуля с множеством

ранее размещенных и неразмещенных модулей. В качестве такой функции, например, может быть выбрана следующая:

$$\underset{max}{J} = OC = \sum_{j \in E_r^k} c_{ij} - \sum_{j \in E_n^k} c_{ij} \qquad (6.1)$$

где c_{ij} – элемент матрицы смежности ВНГ; E_r^k è E_n^k – соответственно множества размещенных и неразмещенных на k–м шаге алгоритма модулей.

Задача размещения при этом сводится к максимизации оценки (6.1) по всем модулям, принадлежащим множеству E_n^k.

Далее для выбранного модуля находится наиболее приемлемая позиция на плате. Для выбора такой позиции используется критерий минимальности длины связей размещаемого модуля с уже размещенными модулями. При этом, очевидно, нет необходимости рассматривать все незанятые на этом шаге позиции, а достаточно оценить лишь множество позиций, соседних с занятыми. При этом позиция p_i, в которую необходимо поместить i–й модуль, определяется из условия минимизации следующего выражения:

$$F = \min_{p_i \in R^k} \sum_{j \in E_h^k} c_{ij} d_{p_i p_j} \qquad (6.2)$$

где c_{ij} – элемент матрицы смежности ВНГ; R^k –множество позиций, соседних с занятыми, на k–м шаге.

Рассмотрим работу последовательного алгоритма размещения по связности на примере (рис. 6.1).

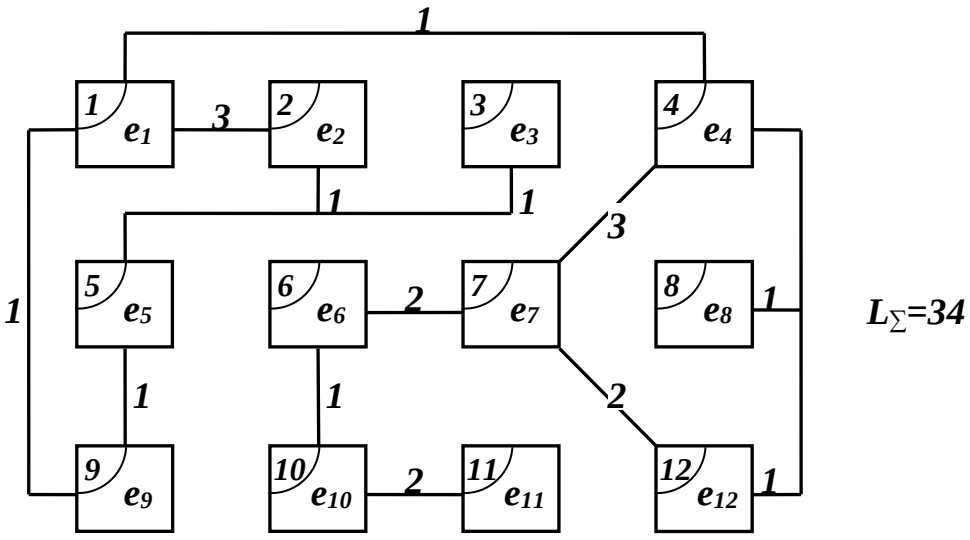


Рис. 6.1.

Пусть элемент e_2 директивно помещается в 6-ю позицию, элемент e_3 директивно помещается в 9-ю позицию.

Чтобы выбрать очередной размещаемый модуль, необходимо в соответствии с (6.1) найти модуль с максимальной оценкой J.

П. 1.

Вычисляем оценки по связности для каждого неразмещенного модуля:

$e_1 : J = 3 - 1 - 1 = +1;$ $e_8 : J = -2;$

$e_4 : J = -1 - 3 - 1 - 1 = -6;$ $e_9 : J = -2;$

$e_5 : J = +1;$ $e_{10} : J = -3;$

$e_6 : J = -3;$ $e_{11} : J = -2;$

$e_7 : J = -7;$ $e_{12} : J = -4.$

Элементы e_1 и e_5 имеют максимальную оценку, равную $+1$. Для дальнейшего анализа выбираем элемент e_1 (с меньшим порядковым номером).

П. 1.1.

Модуль e_1 в соответствии с (6.2) размещаем во 2-ю позицию (с меньшим порядковым номером) (см. ниже).

	+		
+	e_2	+	
e_3	+		

$J_{max} = 1$ - соответствующий элемент e_1;

$R^{k=1} = \{2, 5, 7, 10\};$

$E_r^{k=1} = \{e_2, e_3\};$

$P_2 = 6; P_3 = 9;$

$c_{12} = 3; \quad c_{13} = 0.$

Определим F, предполагая, что модуль e_1 размещен

во 2-й позиции: $F = c_{12} \times d_{26} = 3 \times 1 = 3;$

в 5-й позиции: $F = c_{12} \times d_{56} = 3 \times 1 = 3;$

в 7-й позиции: $F = c_{12} \times d_{76} = 3;$

в 10-й позиции: $F = c_{12} \cdot d_{10\ 6} = 3$.

Окончательно размещаем e_1 во 2-ю позицию (с меньшим порядковым номером) и корректируем массив соседних позиций.

+	e_1	+	
+	e_2	+	
e_3	+		

$R^2 = \{1, 3, 5, 7, 10\};$

$E_r^2 = \{e_1, e_2, e_3\}$

П. 2.

Определим следующий модуль для размещения.

$e_4 : J = -4;$ $e_5 : J = +1;$ $e_6 : J = -3;$

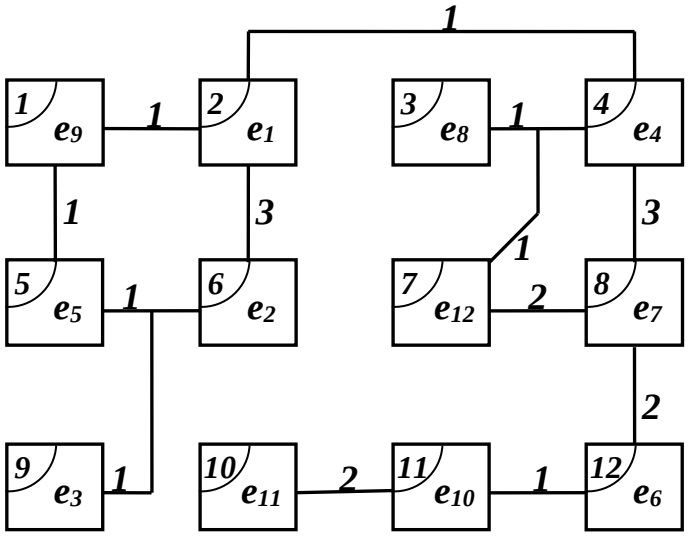
$e_7 : J = -7;$ $e_8 : J = -2;$ $e_9 : J = 0;$

$e_{10} : J = -3;$ $e_{11} : J = -2;$ $e_{12} : J = -4.$

Модуль e_5 имеет максимальную оценку, равную $+1$. в соответствии с (6.2) он помещается в 5-ю позицию.

И т. д. и т. п.

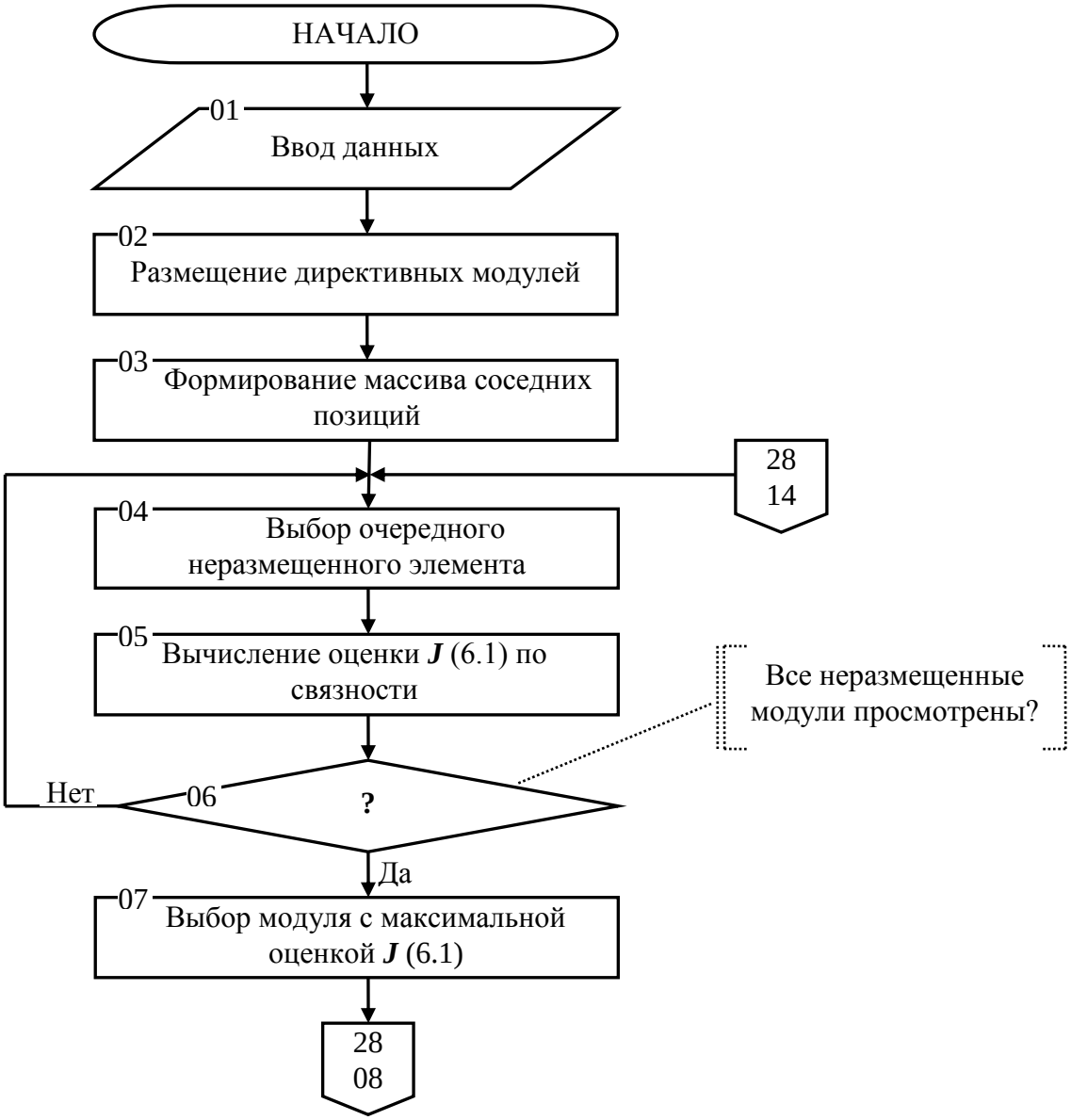
Окончательный вариант размещения представлен на рис. 6.2.



L_{Σ}=

Рис. 6.2.

Укрупненная схема последовательного алгоритма размещения приведена на рисунке 6.3.



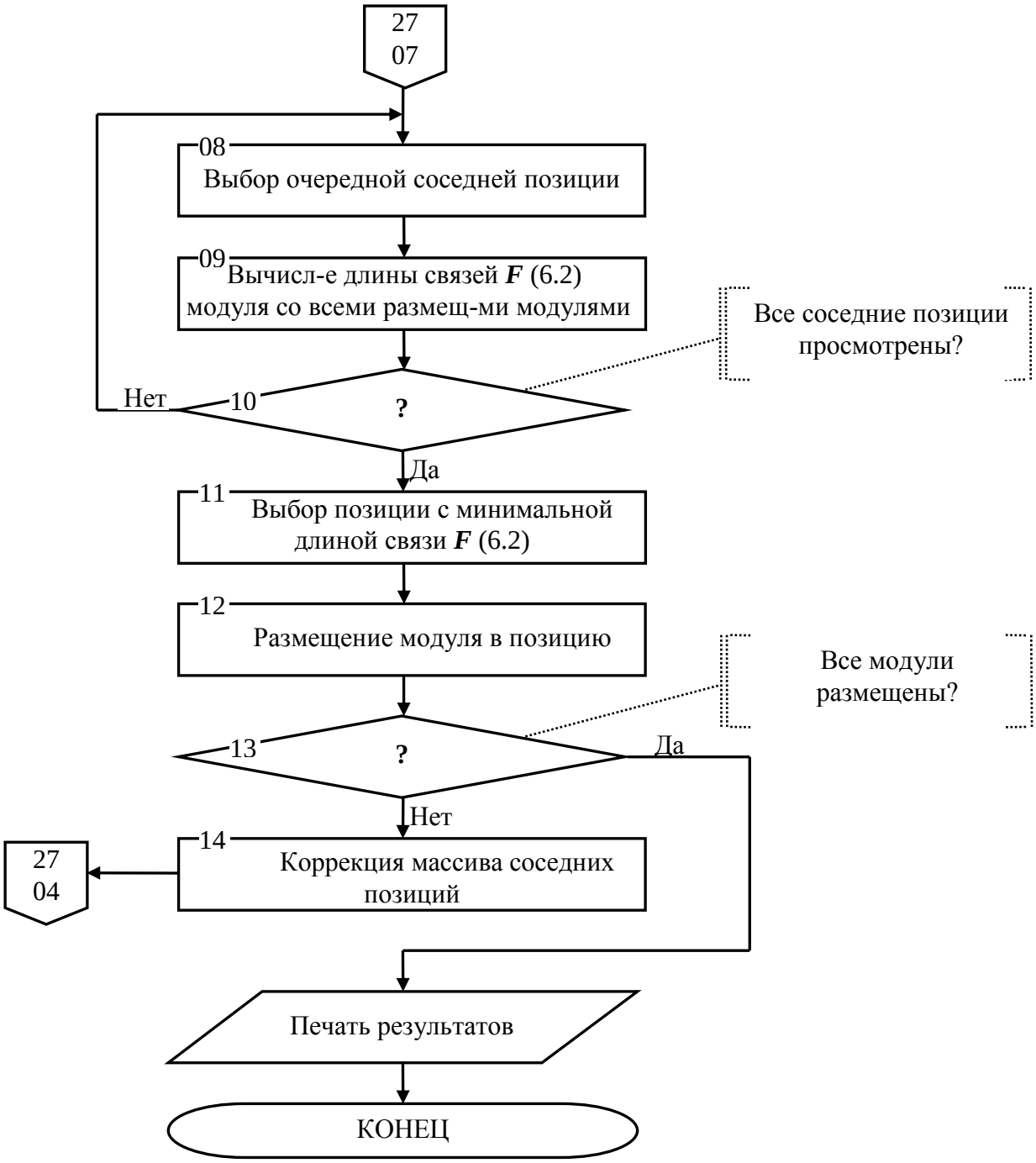


Рис. 6.3. Схема последовательного алгоритма размещения.

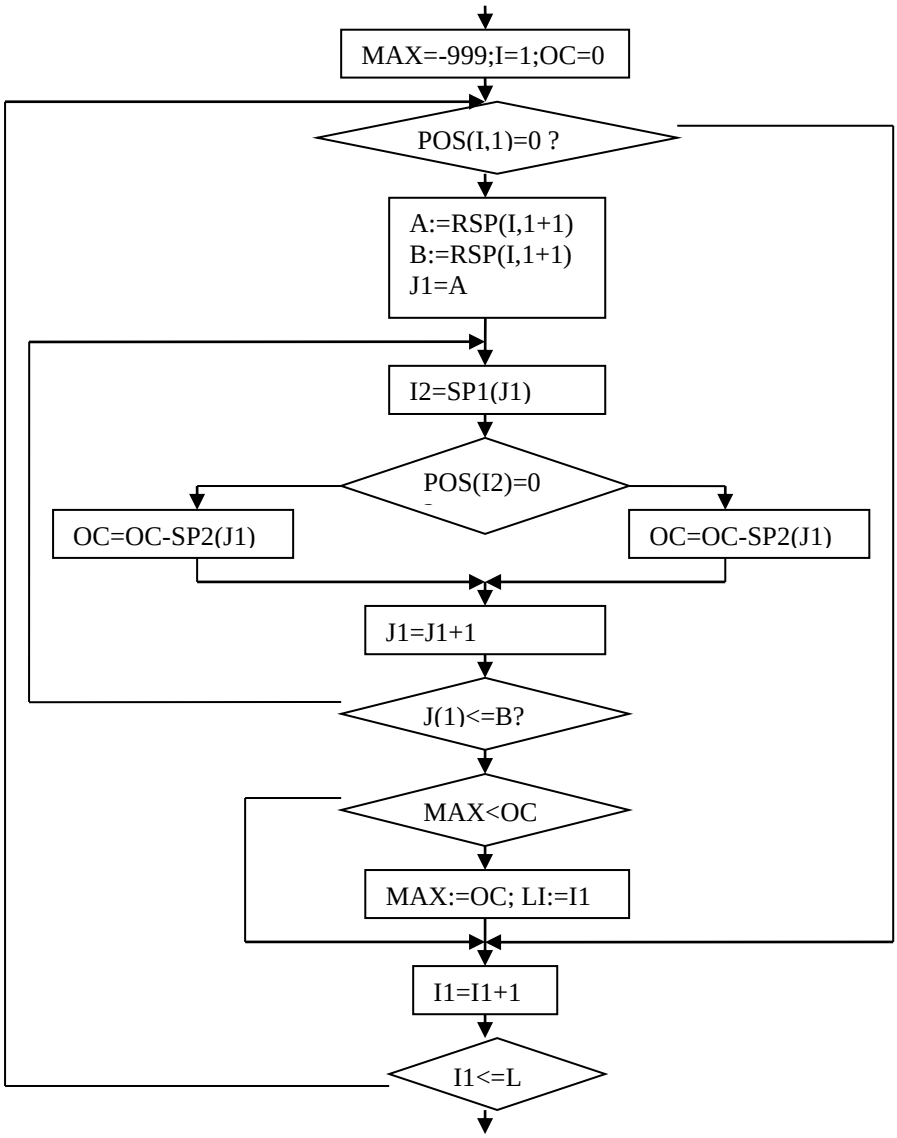
Замечание: соседние позиции.

Приведем краткий перечень идентификаторов для реализации последовательного алгоритма размещения.

L – число элементов;
D – число директивных элементов;
M – число позиций по горизонтали;
N – число позиций по вертикали;
E(I) – массив директивных элементов;
P(I) – массив директивных позиций;
POS(LI) – номер позиции, в которую размещается элемент с номером ***LI***;
Z(I) – массив соседних позиций;

SP1 – список элементов, связанных с данным элементом;
SP2 – список количества связей между элементами;
RSP – список, устанавливающий связность ***SP1*** и ***SP2***.

Приведем фрагмент развернутой блок схемы алгоритма позволяющей определить элемент или максимальную оценку.



Учтем, что списки имеют вид SP1=(2,4,9), SP2=(1,1), RSP(1)=0, RSP(1+1)=1, RSP(2)=3 от E(1) переходим к E(4).
При выборе позиции элемента ei часто используется не модель ВНГ, а модель ГГ. В этом случае при определении позиции можно учесть взаимное расположение элемента в цепи, а также использовать в качестве оценки для позиции критерий минимизации многоугольника охватывающего элементы цепи.

Тема Параллельно-последовательное размещение
Метод обратного размещения.

Для каждого из неразмещенных элементов $e_i \in E, i = \overline{1,n}$, вычисляется некоторая оценка.

Вычисляется также некоторая оценка и для каждого посадочного места. Все элементы и посадочные места упорядочиваются, и осуществляется одновременное размещение всех элементов в позиции.

Пусть матрица $C = \|c_{ij}\|_{m\times n}$ - матрица смежности ВНГ, $D = \|d_{ij}\|_{n\times n}$ - матрица расстояний между позициями.

В соответствии с указанным методом для каждого элемента ***e_i*** рассчитывается суммарное число связей i-го элемента с остальными частями схемы $C_i = \sum_{\substack{ei\in E \\ j\neq i}} C_{ij}$ **(1)**

Для каждого посадочного места вычисляется суммарная длина расстояний j-ого посадочного места со всеми остальными позициями $D_j = \sum_{\substack{p_i\in P \\ i\neq j}} d_{ji}$, где P – число посадочных мест на плате.

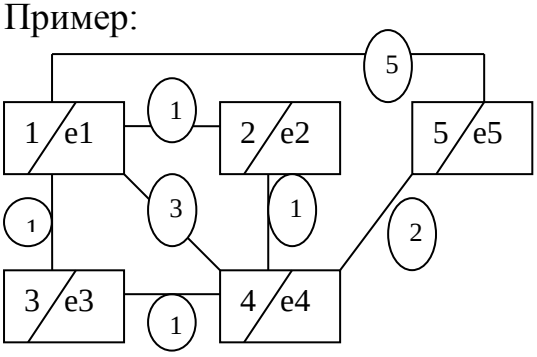
Все оценки связанности C_i упорядочиваются по возрастанию, а оценки длины d_{ji} - по убыванию:

$c_{i1} \le c_{i2} \le c_{i3} \le ... \le c_{in}$

$d_{j1} \ge d_{j2} \ge d_{j3} \ge ... \ge d_{jn}$

Элемент e_{I1} устанавливается в позицию $P_{j(1)}$, $e_{I2} \to P_{j(2)}$ и т.д. Это связано с тем, что min скалярное умножение двух векторов будет тогда, когда компоненты первого вектора упорядочены по возрастанию, а элементы другого по убыванию.

Пример:



Распишем матрицы C и D:

$$C = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \left\|\begin{array}{ccccc} 0 & 1 & 1 & 3 & 5 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 2 \\ 5 & 0 & 0 & 2 & 0 \end{array}\right\| \end{array} \qquad D = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{array} \begin{array}{c} \begin{array}{ccccc} P_1 & P_2 & P_3 & P_4 & P_5 \end{array} \\ \left\|\begin{array}{ccccc} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 2 & 1 & 1 \\ 1 & 2 & 0 & 1 & 3 \\ 2 & 1 & 1 & 0 & 2 \\ 2 & 1 & 3 & 2 & 0 \end{array}\right\| \end{array}$$

$d_1 = 6, d_2 = 5, d_3 = 7, d_4 = 6, d_5 = 8$

L_начальное=1+10+6+4+1+1+1=24

c_1э=10, c_2э=2, c_3э=2, c_4э=7, c_5э=7

d_1n=6, d_2n=5, d_3n=7, d_4n=6, d_5n=8

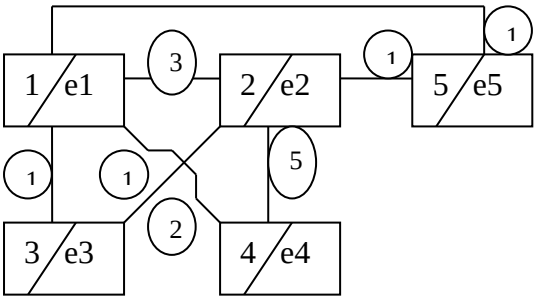
Упорядочим ***c_i*** по возрастанию, ***d_i*** по убыванию:

c_2э=2, c_3э=2, c_4э=7, c_5э=7, c_1э=10

d_5n=8, d_3n=7, d_1n=6, d_4n=6, d_2n=5

Таким образом, второй элемент размещаем в 5-ю позицию, 3-ий - в 3-ю позицию, 4-й - в 1-ю, 5-й - в 4-ю, 1-й - в 2-ю.

Окончательный вариант размещения приведен на рисунке:



$L_{суммарная}$=18 (18<24!), что и требовалось доказать.

Итерационные алгоритмы размещения

Эти алгоритмы предполагают заданным некоторое начальное размещение, которое может быть получено некоторым конструктивным алгоритмом размещения. Исследуются некоторое множество возможных вариантов размещения, близких в некотором смысле к начальному, и находящие размещение с меньшим значением целевой функции $L_{суммарное}$. Найденное размещение принимается за исходное, и процесс повторяется. Алгоритм заканчивает работу, если в окрестности полученного размещения отсутствуют варианты с меньшими значениями целевой функции.

Различают 2 группы итерационных алгоритмов:

1. алгоритмы парных перестановок
2. алгоритмы групповых перестановок

Алгоритм парных перестановок.

Суть: Последовательное целенаправленное улучшение произвольного начального размещения модулей на плате по выбранному критерию путем парных перестановок. С этой целью на каждой итерации алгоритм осуществляет вычисление приращенной суммарной длины всех связей для всевозможных парных перестановок модулей. Из всего множества перестановок, дающих отрицательное приращение, выбирается подмножество, которое удовлетворяет следующим требованиям:

1. Выбранное подмножество перестановок позволяет максимально уменьшить суммарную длину всех связей.
2. Подмножество образует лишь независимые перестановки, в которых модули не связаны с модулями других переставляющихся пар.

Далее осуществляются перестановки выделенных таким образом пар модулей и переход к следующей итерации.

Описанный итерационный процесс сходится к локальнооптимальному размещению модулей на плате. Выведем формулы для суммарной длины L всех связей и ее приращение ΔL при перемещении модулей.

Пусть имеется плоское или объемное плато с узлами, предназначенными для установки модулей. Задана матрица расстояний $D = \|d_{ij}\|_{n\times n}$, где $\boldsymbol{d_{ij}}$ определяется в обычной или ортогональной метриках.

Даны некоторые совокупности модулей, подлежащих размещению, и матрица $C = \|c_{ij}\|_{m\times n}$ числа связей этих модулей между собой.

Пусть на некоторой итерации имеется следующее размещение модулей на плате:

узлы	1	2	3	…	e	…	n
модули	t_1	t_2	t_3	…	t_e	…	t_n

где t_e - номер модуля, оказавшегося размещенным в узле платы.

Поставим в соответствие этому варианту размещения матрицу связей $R = \|r_{ij}\|_{n\times n}$.

Элемент r_{ij} равен числу связей между модулями, t_i и t_j находятся на данной итерации в узлах i и j соответственно.

Так как расстояние между узлами i и k равно d_{ik}, то суммарная длина связей между модулями t_i и t_k:

$l_{ik}=r_{ik}*d_{ik}$, **(1)**

отсюда суммарная длина связей t_i модуля, расположенного в i-ом узле, связаного со всеми модульными схемами, равна:

$$L_i = \sum_{k=1}^{n}(r_{ik} * d_{ik}) \quad \textbf{(2)}$$

Для суммарной длины всех связей при данном варианте получаем формулу:

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}(r_{ik} * d_{ik}) \quad \textbf{(3)}$$

Примечание: пусть $i=k=\overline{1,3}$ тогда

$$L = r_{11} * d_{11} + r_{12} * d_{12} + r_{13} * d_{13} + r_{21} * d_{21} + r_{22} * d_{22} + r_{23} * d_{23} + r_{32} * d_{32} + r_{33} * d_{33}$$

Найдем формулу для приращения суммарной длины всех связей при перестановке модулей t_i и t_j, расположенных в узлах i и j соответственно.

Для суммарной длины связей t_i и t_j со всеми остальными модулями имеем:

$$L_{ij} = \sum_{k=1}^{n}(r_{ik} * d_{ik}) + \sum_{k=1}^{n}(r_{jk} * d_{jk}) - 2 * r_{ij} * d_{ij} \quad \textbf{(4)}$$

Замечание: пусть $n=4, i=2, j=3$

$$L_{23} = r_{21} * d_{21} + r_{22} * d_{22} + r_{23} * d_{23} + r_{24} * d_{24} + r_{31} * d_{31} + r_{32} * d_{32} + r_{33} * d_{33} + r_{34} * d_{34} - 2 * r_{23} * d_{23}$$

Найдем формулу для приращения суммарной длины всех связей при перестановке модулей t_i и t_j, расположенных в узлах i и j соответственно. Для суммарной длины связей t_i и t_j со всеми оставшимися модулями имеем:

$$L_{ij} = \sum_{k=1}^{n}(r_{jk} * d_{ik}) + \sum_{k=1}^{n}(r_{ik} * d_{jk}) \quad \textbf{(5)}$$

Заметим, что взаимная перестановка модулей t_i и t_j соответствует перестановке строк и столбцов с номером i и j в матрице R. Вычитая из **(5)** выражение **(4),** определим приращение суммарной длины всех связей после перестановки модулей с номером t_i и t_j .

$$\Delta L_{ij} = 2 * r_{ij} * d_{ij} - \sum_{k=1}^{n}(r_{ik} * r_{jk})(d_{ik} * d_{jk}) \quad \text{i,j}=\overline{1,n} \quad \textbf{(6)}$$

Введем в рассмотрение матрицу $P = R \times D$, элементы которой определяются по формуле:

$$p_{ij} = \sum_{k=1}^{n} r_{ik} \times d_{kj} \qquad (7).$$

Полусумма диагональных элементов матрицы P (полуслед) равна суммарной длине всех связей, определяемой формулой (3). С помощью элементов матрицы P могут быть легко вычислены элементы ΔL_{ij} матрицы приращений для всех парных перестановок. С учетом симметричности матрицы D выражение (6) преобразуется к виду:

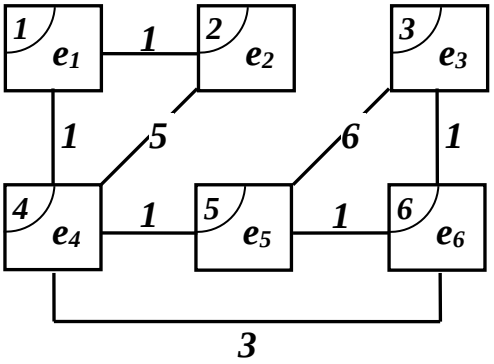
$$\Delta L_{ij} = 2 r i_j d_{ij} + q_{ij} \qquad (8) \qquad i, j = \overline{1,n};$$

где $q_{ij} = (p_{ij} - p_{ii}) + (p_{ji} - p_{jj}) = \gamma_{ij} + \gamma^{*}_{ij}$ (9)

Вычисляя по выражениям (8) и (9) элементы матрицы приращений ΔL, можно выбрать подмножество перестановок, удовлетворяющих вышеперечисленным требованиям.

Пример.

Пусть начальное размещение связанных с системой модулей на плате с 6-ю узлами имеет вид, представленный на рисунке:



$L_{\Sigma}=33$

Во втором узле размещен разъем, который запрещено переносить на другую позицию. Если расстояния d_{ij} между узлами i и j определены в ортогональной метрике, то матрица D будет иметь вид:

$D =$

	p_1	p_2	p_3	p_4	p_5	p_6
p_1	0	1	2	1	2	3
p_2	1	0	1	2	1	2
p_3	2	1	0	3	2	1
p_4	1	2	3	0	1	2
p_5	2	1	2	1	0	1
p_6	3	2	1	2	1	0

$R =$

	e_1	e_2	e_3	e_4	e_5	e_6
e_1	0	1	0	1	0	0
e_2	1	0	0	5	0	0
e_3	0	0	0	0	6	1
e_4	1	5	0	0	1	3
e_5	0	0	6	1	0	1
e_6	0	0	1	3	1	0

Составим соответствующую начальному размещению матрицу $P=R\text{x}D$. Элемент p_{11} определяется из выражения: $p_{11}=r_{11}d_{11}+r_{12}d_{21}+r_{13}d_{31}+r_{14}d_{41}+r_{15}d_{51}+$ $+r_{16}d_{61}=0+1\cdot1+0+1\cdot1+0+0=2$ и т. д. и т. п.

$P =$

	1	2	3	4	5	6
1	2	2	4	2	2	4
2	5	11	17	8	7	13
3	15	8	13	8	1	6
4	16	8	12	18	10	14
5	16	10	4	20	14	8
6	7	8	11	4	5	8

Суммарная длина всех связей в начальном размещении равна полусуму матрицы P: $L_\Sigma = \frac{1}{2}(2+11+13+18+14+8)=33$.

Вычислим далее последовательно матрицы γ, γ^*, Q и ΔL. Отметим, что матрица γ^* есть матрица γ, у которой строки заменены на столбцы, а $Q = \gamma+\gamma^*$.

$\gamma = \|p_{ij}-p_{ii}\|$ =

	1	2	3	4	5	6
1	0	0	2	0	0	2
2	-6	0	6	-3	-4	2
3	2	-5	0	-5	-12	-7
4	-2	-10	-6	0	-8	-4
5	2	-4	-10	6	0	-6
6	-1	0	3	-4	-3	0

$\gamma^* = \|p_{ij}-p_{ii}\| = \gamma^T =$

	1	2	3	4	5	6
1	0	-6	2	-2	2	-1
2	0	0	-5	-10	-4	0
3	2	6	0	-6	-10	3
4	0	-3	-5	0	6	-4
5	0	-4	-12	-8	0	-3
6	2	2	-7	-4	-6	0

$Q = \gamma+\gamma^*$ = симметр.

	1	2	3	4	5	6
1	0	-6	4	-2	2	1
2		0	1	-20	-8	1
3			0	-11	-22	-4
4				0	-2	-8
5					0	-9
6						0

$\Delta L =$ симметр.

	1	2	3	4	5	6
1	0	—	—	0	—	—
2		0	—	—	—	—
3			0	-11	2	-2
4				0	0	0
5					0	-7
6						0

$\Delta L_{ij} = 2r_{ij}d_{ij}(>0) + q_{ij}(>, <, =0)$

В матрице ΔL прочерком указаны элементы, которые заведомо являются положительными в силу положительности соответствующих элементов матрицы Q, а также элементы, отвечающие недопустимому переразмещению модуля 2 (разъема). Из рассмотрения матрицы ΔL видно, что к уменьшению суммарной длины всех связей приводят взаимные перестановки пар модулей, расположенных в начальном размещении в следующих узлах: 3 и 4 (ΔL=-11), 3 и 6 (ΔL=-2), 5 и 6 (ΔL=-7).

Примечания.

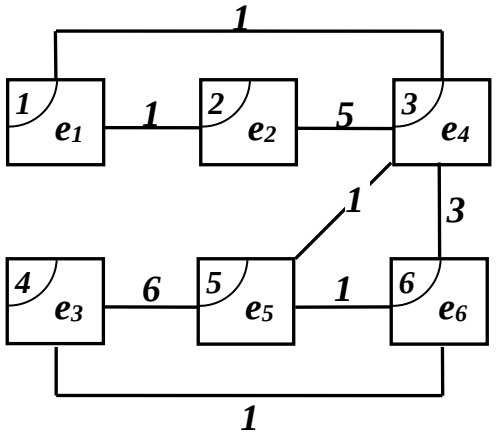
1. Пары модулей $\{e_i$ и $e_j\}$ и $\{e_k$ и $e_q\}$ являются независимыми, если элементы матрицы связей R равны нулю. Для нашего примера допустимой является лишь одна перестановка (пара 3 и 4), обеспечивающая $|\Delta L_{max}|$=11.

2. Пары модулей, относящиеся к 1-й строке или 1-му столбцу $\Delta \boldsymbol{L}$, являются зависимыми.

(3,4), (3,6) – зависимы;

(3,4), (5, 6) – зависимы.

Производим перестановку элементов $\boldsymbol{e_3}$ и $\boldsymbol{e_4}$ (см. рис.):



$\boldsymbol{L_\Sigma=22}$

Для нового варианта размещения находим матрицы $\boldsymbol{R}$ и $\boldsymbol{P}$. С этой целью необходимо в матрице $\boldsymbol{R}$, соответствующей начальному варианту размещения, поменять местами элементы 3-го и 4-го столбцов и 3-ей и 4-ей строк. Матрица $\boldsymbol{P}$ получается умножением новой матрицы $\boldsymbol{R}$ на старую $\boldsymbol{D}$.

$\boldsymbol{P}$ =

	1	2	3	4	5	6
1	3	1	1	5	3	3
2	10	6	2	16	12	8
3	16	8	12	18	10	14
4	15	8	13	8	1	6
5	11	15	19	5	9	18
6	9	6	5	10	7	6

Суммарная длина всех связей нового варианта

$$\boldsymbol{L_\Sigma} = \frac{1}{2}(3+6+12+8+9+6)=22<33.$$

Выполним следующую итерацию.

$\boldsymbol{\gamma}$ =

	1	2	3	4	5	6
1	0	-2	-2	2	0	0
2	4	0	-4	10	6	2
3	4	-4	0	6	-2	2
4	7	0	5	0	-7	2
5	2	6	10	-4	0	4
6	3	0	-1	4	1	0

$\boldsymbol{Q}$ =

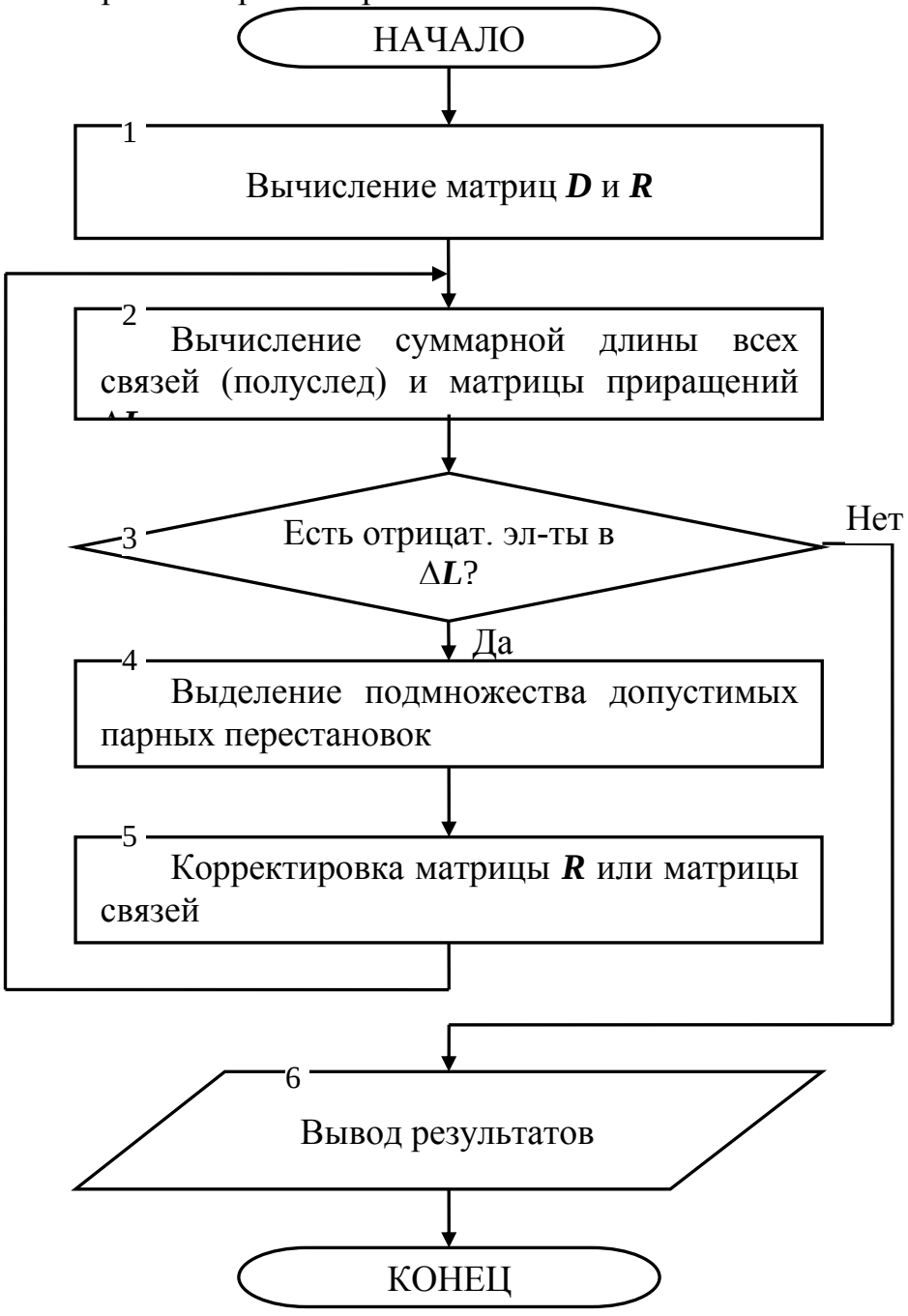
	1	2	3	4	5	6
1	0	2	2	0	2	3
2		0	-8	10	12	2
3			0	11	8	1
4				0	-11	2
5					0	5
6						0

1 2 3 4 5 6

$$\Delta \boldsymbol{L} = \begin{array}{c|cccccc} 1 & 0 & — & — & — & — & — \\ 2 & & 0 & — & — & — & — \\ 3 & & & 0 & — & — & — \\ 4 & & & & 0 & 1 & — \\ 5 & & & & & 0 & — \\ 6 & & & & & & 0 \end{array}$$

Матрица $\Delta \boldsymbol{L}$ не имеет ни одного отрицательного элемента, поэтому процесс улучшения начального размещения закончен. Т. о., размещение, изображенное на последнем рисунке, соответствует локальному оптимуму.

Блок-схема алгоритма парных перестановок.



Алгоритмические методы трассировки монтажных соединений

Трассировка заключается в определении конкретной геометрии печатного или проводного монтажа, реализующего соединения между элементами схемы. Исходными данными для трассировки являются список цепей, метрические параметры и топологические свойства типовой конструкции и её элементов и результаты

решения задачи размещения, по которым находятся координаты выводов элементов. Формальная постановка задачи трассировки и методы её решения в значительной степени зависят от вида монтажа (проводной или печатный) и конструктивно-технологических ограничений, определяющих метрические параметры и топологические свойства монтажного пространства.

ТРАССИРОВКА ПРОВОДНОГО МОНТАЖА(провода с изоляцией)

Трассировка проводного монтажа может осуществляться по прямым, соединяющим выводы элементов (монтаж в навал) или с помощью жгутов, которые прокладываются в специальных каналах. Основные ограничения – количество проводников, которые можно подсоединить к 1 выводу, и число проводов в каждом жгуте (пропускная способность каналов).

Трассировка проводного монтажа заключается в определении порядка соединения выводов в соответствии с принципиальной электрической схемой или с учетом заданных ограничений. Критерием качества обычно является минимальная суммарная длина соединения. Нахождение порядка соединения выводов модулей внутри цепи сводится к задаче построения на вершинах кратчайшего связывающего дерева (КСД).

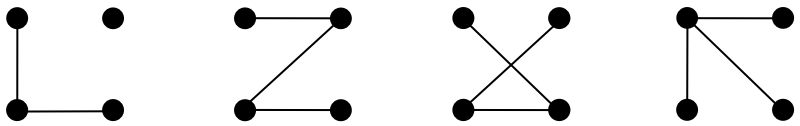
Будем использовать модель схемы (цепи) в виде графа, в котором выводам элементов сопоставлены вершины, и на них строится полный граф цепи. При этом, если число вершин графа равняется ***n***, то число ребер полного графа $r = n \cdot (n-1)/2$.

При таком подходе каждая элементарная цепь может быть представлена отдельной компонентой связности.

Ставится задача построения КСД для схемы на тех компонентах связности, число вершин в которых больше 2.

Отметим, что расстояния между каждой парой вершин может быть определено в обычной или ортогональной метрике. На n вершинах полного графа можно построить t_n деревьев, где $t_n = n^{n-2}$.

Пример: ***n***=4



При $n \geq 10$ точное решение задачи построения КСД методом полного перебора не целесообразно. Существуют приближенные алгоритмы решения этой задачи, дающие результаты, достаточно близкие к оптимальным.

Алгоритм Красккала

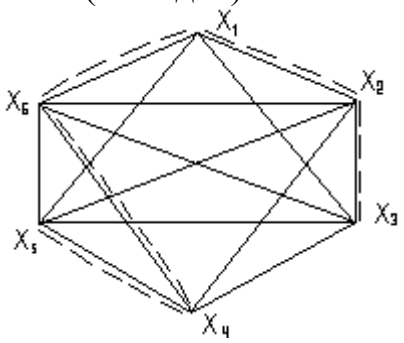
Пусть задан полный граф $G = (X, U)$. Каждому ребру графа поставлено в соответствие число U_{ij} – вес или длина данного ребра. Также $i \neq j;\ i, j = \overline{1,n}$. Такой граф называется взвешенным. Ставится задача: среди всех деревьев ***(n-(n-2))***, которые можно выделить в данном графе, требуется найти дерево с минимальной длиной. В дереве ***(n-1)*** ребро.

Пусть $C = \|c_{ij}\|_{n \times n}$ – матрица длины ребер графа.

1. Все ребра графа ***n(n-1)/2*** упорядочиваются в порядке убывания длины, начиная с ребра $C_1 = \min\limits_{i,j} c_{ij}$ и заканчивая $C_n = \max\limits_{i,j} c_{ij}$.
2. Последовательно просматривается упорядоченное множество U' длин ребер, на каждом шаге в дерево включается одно ребро. В качестве такого ребра выбирается ребро среди не включенных в дерево, имеющее минимальную длину и не образующее циклов с ребрами, уже вошедшими в дерево.

Отметим, что при работе этого алгоритма возможно появление несвязных поддеревьев, которые затем соединяются, образуя одну компоненту связности.

Пример: требуется построить КСД для цепи, содержащей 6 эквипотенциальных контактов (выводов). Полный граф имеет вид:



Пусть матрица длин ребер графа цепи имеет вид:

	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6
*x*1	0	3	7	2	6	1
*x*2	3	0	1	3	9	6
U= *x*3	7	1	0	3	5	9
*x*4	2	3	3	0	1	1
*x*5	6	9	5	1	0	2
*x*6	1	6	9	1	2	0

n = 6; n*(n-1) = 15

рис.1

1. Упорядочивание длин ребер полного графа

$U_{16}^{1}, U_{23}^{1}, U_{45}^{1}, U_{46}^{1}, U_{14}^{2}, U_{56}^{2}, U_{12}^{3}, U_{24}^{3}, U_{34}^{3}, U_{34}^{3}, U_{35}^{5}, U_{15}^{6}, U_{26}^{6}, U_{13}^{7}, U_{25}^{9}, U_{36}^{9}$ - 15 элементов.

2.1. Включаем в U_{16}^{1} в дерево и получаем 1-ую изолированную компоненту связности.

2.2. Включаем в дерево U_{23}, получаем 2-ую изолированную компоненту связности (2,3).

2.3. Включаем U_{45}, получаем 3-ю изолированную компоненту связности (4,5).

2.4. Включ. U_{46}, получаем 4-ю изолированную компоненту связности (1,6,4,5).

Ребро U_{14} образовывает цикл - брать нельзя.

Ребро U_{56} тоже образовывает цикл - брать нельзя.

2.5. Включаем в дерево U_{12}. КСД построено. Суммарная длина – 7.

Недостатки:

1) Необходимо на каждом шаге проверять условия необразования цикла, а также наблюдения за различными компонентами связности.
2) Большое время уходит на упорядочивание ребер полного графа.

АЛГОРИТМ ПРИМА

Использует тот же принцип соединения ближайших вершин, что и алгоритм Краскала, но на каждом шаге к строящемуся дереву присоединяется ближайшая изолированная вершина. В основе алгоритма лежат 2 теоремы:

Т.1. Каждая вершина КСД непосредственно связана, по крайней мере, с 1 ближайшей вершиной.

Т.2. Каждый измеряемый фрагмент поддерева связан, по крайней мере, с 1 из излированных фрагментов кратчайшим ребром.

В соответствии с Т.1. построение КСД может быть начато с произвольной вершины графа:

1) Выбираем вершину, например, x_1 и находим кратчайшее ребро, инцидентное этой вершине. Это ребро U_{16}.

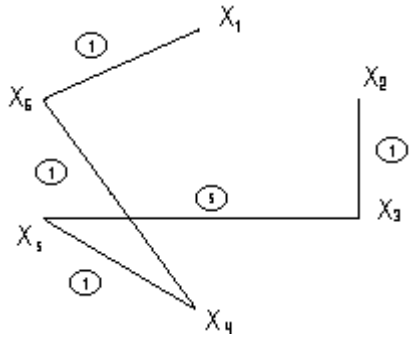
2) Включаем в КСД это ребро, вычеркиваем 1 и 6 столбцы (чтобы не было циклов), помечаем 6 строку и выбираем из нее минимальный элемент $U_{64}=1$.

3) Включаем в КСД U_{64}, вычеркиваем 4 столбец, помечаем 4 строку и выбираем из нее минимальный элемент U_{45}=1.

4) Включаем в КСД U_{45}, помечаем строку 5, вычеркиваем 5 столбец, выбираем U_{53}=5.

5) Включаем в КСД, вычеркиваем 3 столбец, помечаем 3 строку и выбираем из неё U_{32}=1.

КСД построено, суммарная длина ребер равняется 9.



Отметим, что вычеркивание столбцов исключает возможность образования циклов. Т.о. на Т-м шаге алгоритма используется теорема 1, и далее теорема 2.

В этом алгоритме всегда формируется 1 компонента связности, т.е. 1 разрастается по дереву.

Алгоритм Прима построения КСД при ограничении на локальные степени вершин

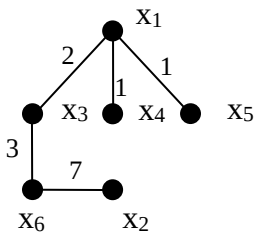
Локальная степень вершины $\rho(x_i)$ - число ребер графа, инцидентных этой вершине.

Задача построения КСД с ограниченными локальными степенями возникает при проектировании проводных соединений, когда ограничено число паек к 1 контакту. Чаще всего количество паек к контакту равняется 2.

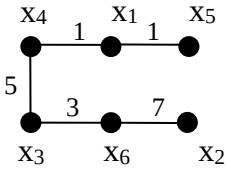
Пример: пусть задана матрица $C=\|c_{ij}\|_{n\times n}$, где ***n*** – число контактов цепи. Пусть необходимо построить КСД при $\rho(x_i)\le m \quad (i=\overline{1,n})$. Для решения можно применить алгоритм Прима с вычеркиванием строки, соответствующей вершине, локальные степени которой становятся равны ***m*.**

	x1	x2	x3	x4	x5	x6
x1	0	3	2	1	1	5
x2	3	0	4	6	8	7
C= x3	2	4	0	5	4	3
x4	1	6	5	0	6	10
x5	1	8	4	6	0	9
x6	5	7	3	10	9	0

КСД без ограничений на $\rho(x_i)$ будет иметь вид:



Локальная степень вершины $\rho(x_1)=3, L_\Sigma=14$.



$L_\Sigma=17$

Алгоритм является приближенным, результат зависит от выбора исходной вершины и от вершины, имеющей на текущем шаге минимальную оценку.

АЛГОРИТМИЧЕСКИЕ МЕТОДЫ МОНТИРОВКИ ПРОВОДНОГО МОНТАЖА

Матрицы Штейна

$\rho(x_i)<2$

Пример:

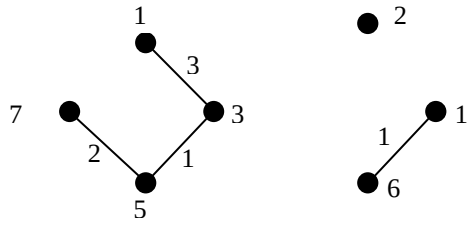
$C=$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0	10	3	5	4	5	4
x_2	10	0	4	6	3	7	8
x_3	3	4	0	3	1	4	3
x_4	5	6	3	0	3	1	4
x_5	4	3	1	3	0	5	2
x_6	5	7	4	1	5	0	4
x_7	4	8	3	4	2	4	0

$C_j^1=\min\limits_{i=1,n} C_{ij}$

Пункт 1: определить в каждом столбце минимальный элемент. Если в ***j***-м столбце минимальным является C_{ij}, а в столбце L минимальным является C_{jl}, и $l>j$, то вместо элемента C_{jl} в столбце l выбирается следующий по величине элемент. Т.о. имеем:

$C_{31}=3, C_{52}=3, C_{53}=1, C_{64}=1, C_{75}=2, C_{36}=4, C_{73}=3, C_{53}=C_{35}=1$

Пункт 2: упорядочить выделенные ребра в порядке невозрастания их длины. Ребра, приводящие к превышению локальной степени $\rho(x)<2$, вычеркнуть: U_{52}, U_{36}, U_{37}. Т.о. после 1-й итерации будут получены 2 фрагмента строящегося КСД – ***a*** и ***b***.



a) b)

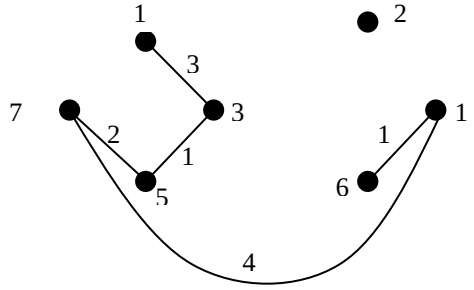
Пункт 3: при переходе к следующей итерации из исходной матрицы ***C*** вычеркнуть столбцы, соответствующие вершинам 3 и 5, локальные степени которых на 1-й итерации уже были равны 2. Полагаем $|l_{71}=\infty|, C_{17}=C_{71}=\infty$. Тогда матрица ***C'***:

$$C'=\begin{array}{c|ccccc} & x_1 & x_2 & x_4 & x_6 & x_7 \\ x_1 & 0 & 10 & 5 & 5 & \infty \\ x_2 & 10 & 0 & 6 & 7 & 8 \\ x_4 & 5 & 6 & 0 & \infty & 4 \\ x_6 & 5 & 7 & \infty & 0 & 4 \\ x_7 & \infty & 8 & 4 & 4 & 0 \end{array}$$

Пункт 4: определить минимальный элемент каждого столбца:

$C_{41}=5, C_{42}=6, C_{74}=4, C_{76}=4, C_{27}=8$

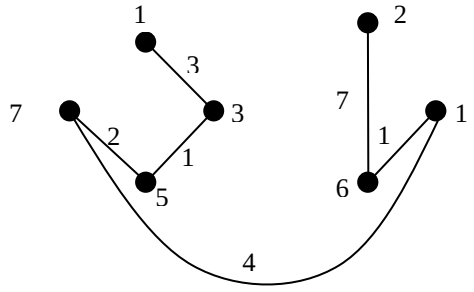
К дереву добавляется ребро $C_{74}=4$, тогда КСД принимает следующий вид:



При переходе к следующей итерации из ***C'*** вычеркнуть столбцы, соответствующие вершинам 4 и 7, локальные степени которых равняются 2. Тогда матрица ***C''***:

$$C''=\begin{array}{c|ccc} & x_1 & x_2 & x_6 \\ x_1 & 0 & 10 & \infty \\ x_2 & 10 & 0 & 7 \\ x_6 & \infty & 7 & 0 \end{array}$$

Минимальное ребро $U_{62}=7$. Окончательный вид КСД на следующем рисунке:



ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ПОСТРОЕНИЯ КСД ПРИ $\rho(x_i)\le 2$

Такие алгоритмы основаны на приближенном решении задачи о коммивояжере с последующим отбрасыванием ребра с максимальным весом.

Пункт 1: выбираем минимальное ребро и строим цикл через 2 вершины.

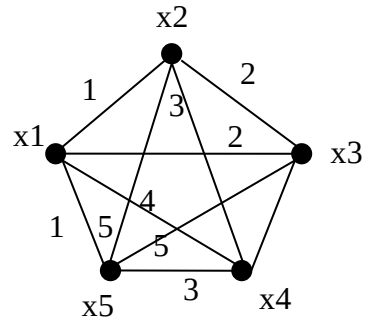
Пункт 2: к этому циклу добавляем одну из оставшихся вершин таким образом, чтобы длина цикла через 3 вершины была минимальной.

Пункт 3: пусть на текущем ***k***-м шаге построен цикл, охватывающий ***r*** вершин. В качестве вершины, включаемой в цикл на текущем шаге, выбирается такая вершина, включение которой приводит к минимальному приращению длины цикла.

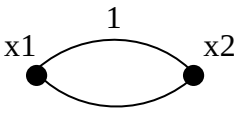
Пункт 4: из гамильтонова цикла исключаем ребро максимальной длины.

Пример:

Дан полный граф цепи. Требуется найти КСД при $\rho(x_i) \le 2$**.**

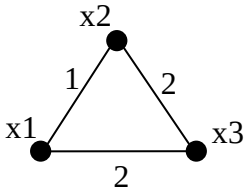


Пункт 1: Строим цепь ***x₁x₂***:



$L_\Sigma = 2$

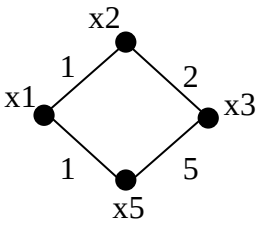
Пункт 2: Строим цикл ***x₁x₂x₃***:



$L_\Sigma = 5$

$\Delta L_\Sigma = 5 - 2 = 3$

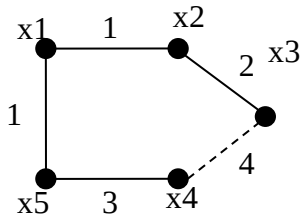
Пункт 3: Строим цикл ***x₁x₂x₃x₅***:



$L_\Sigma = 9$

$\Delta L_\Sigma = 9 - 5 = 4$

Пункт 4: Строим цепь ***x₄x₅x₁x₂x₃***:



КСД $x_4x_5x_1x_2x_3$ построено. Локальные степени вершин не превышают 2.

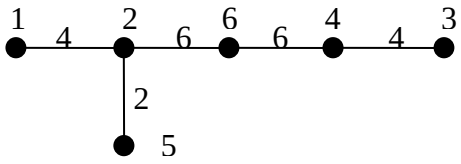
ТОЧНЫЕ МЕТОДЫ ПОСТРОЕНИЯ КСД ПРИ $\rho(x_i) \le 2$

Метод ветвей и границ

Пусть матрица длин ребер полного графа, построенного на площади вывода электрической цепи ***C*** имеет следующий вид:

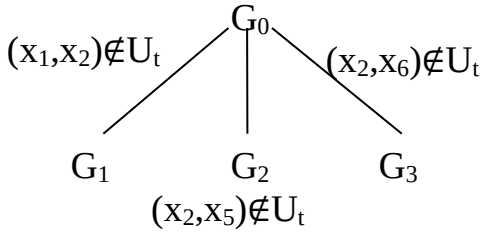
$$C = \begin{array}{c} \\ * \\ \\ *** \\ \\ ** \end{array} \left\| \begin{array}{cccccc} 0 & 4 & 10 & 18 & 5 & 10 \\ 4 & 0 & 12 & 8 & 2 & 6 \\ 10 & 12 & 0 & 4 & 18 & 16 \\ 18 & 8 & 4 & 0 & 14 & 6 \\ 5 & 2 & 18 & 14 & 0 & 16 \\ 10 & 6 & 16 & 6 & 16 & 0 \end{array} \right\|$$

Пусть G_0 – множество всех деревьев, локальные степени вершин которых не превышают 2. Среди элементов множества ρ_0 необходимо выбрать такое дерево, которое имеет минимальное L_Σ. В соответствии с методом ветвей и границ необходимо получить оценку ξ, которая заведомо меньше оценки ξ^* любого из деревьев ***t***, и множество дополнительных решений G_0. Для получения снимаем ограничения на $\rho(x_i)$ и строим КСД. Суммарная длина ребер может быть выбрана в качестве оценки. КСД имеет вид:



$\xi_{G_0} = 22$

Рисуем дерево ветвей, где U^T – это множество ребер дерева t.



Решение необходимо искать среди деревьев при $\rho(x_i) \le 2$. Для того, чтобы получить оценку $\xi(G_1)$, необходимо воспользоваться алгоритмом Прима без ограничений на локальные степени $\rho(x_i)$, полагая в матрице ***C*** элементы $C_{21} = C_{12} = \infty$.

Получим матрицу ***C'***, из которой может быть построено КСД, как на рисунке.

$$C_1 = \begin{array}{c} \\ ** \\ \\ \\ * \\ *** \end{array} \left\| \begin{array}{cccccc} 0 & \infty & 10 & 18 & 5 & 10 \\ \infty & 0 & 12 & 8 & 2 & 6 \\ 10 & 12 & 0 & 4 & 18 & 16 \\ 18 & 8 & 4 & 0 & 14 & 6 \\ 5 & 2 & 18 & 14 & 0 & 16 \\ 10 & 6 & 16 & 6 & 16 & 0 \end{array} \right\|$$

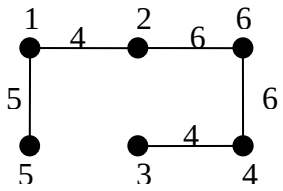
$L_\Sigma = 23, \rho(x_i) \le 2$

$\xi(\mathrm{G}_1) = 23$

Для получения оценки $\xi(G_2)$ полагаем в матрице ***C*** элементы $C_{25} = C_{52} = \infty$:

$$C_2 = \begin{array}{c} \\ * \\ \\ *** \\ \\ ** \end{array} \left\| \begin{array}{cccccc} 0 & 4 & 10 & 18 & 5 & 10 \\ 4 & 0 & 12 & 8 & \infty & 6 \\ 10 & 12 & 0 & 4 & 18 & 16 \\ 18 & 8 & 4 & 0 & 14 & 6 \\ 5 & \infty & 18 & 14 & 0 & 16 \\ 10 & 6 & 16 & 6 & 16 & 0 \end{array} \right\|$$

Строим КСД:

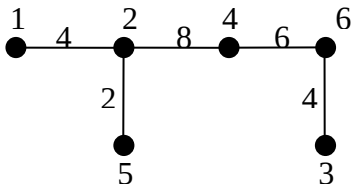


$L_\Sigma = 25, \rho(x_i) \le 2$

$\xi(G_2) = 25$

Для получения оценки $\xi(G_3)$ полагаем $C_{26} = C_{62} = \infty$. Получаем матрицу ***C₃,*** из которой формируется КСД, приведенное на рисунке:

$$C_2 = \begin{array}{c} \\ * \\ \\ ** \\ \\ \\ \end{array} \left\| \begin{array}{cccccc} 0 & 4 & 10 & 18 & 5 & 10 \\ 4 & 0 & 12 & 8 & 2 & \infty \\ 10 & 12 & 0 & 4 & 18 & 16 \\ 18 & 8 & 4 & 0 & 14 & 6 \\ 5 & 2 & 18 & 14 & 0 & 16 \\ 10 & \infty & 16 & 6 & 16 & 0 \end{array} \right\|$$



Таким образом, дерево ***G₁***, приведенное на рисунке, является оптимальным деревом при $\rho(x_i) \le 2$.

ТРАССИРОВКА ПЕЧАТНЫХ СОЕДИНЕНИЙ

Исходными данными для решения задачи трассировки печатного монтажа являются: принципиальная схема соединений; таблица размещения элементов; конструкторско-технологические ограничения:

1. размеры печатной платы
2. количество слоёв многослойной печатной платы (МПП)
3. области, запрещенные для прокладки проводников
4. минимальная ширина проводников
5. минимальное расстояние между проводниками
6. типы и размеры контактных площадок
7. шаг координатной сетки

При этих исходных данных необходимо реализовать все соединения так, чтобы соединения, принадлежащие разным цепям, в одном слое не имели пересечений. Дополнительно необходимо оптимизировать следующие показатели качества:

1. минимум суммарной длины проводников (из-за $\tau_{зад}$)
2. минимум числа переходных отверстий (электрическая надежность и механическая прочность)
3. минимальная длина параллельных участков соседних проводников
4. максимальная удаленность проводников друг от друга ($L_{пар}$ и $C_{пар}$ должны быть минимальными)
5. равномерное распределение проводников в монтажной плоскости

Трассировка печатных соединений предполагает выполнение следующих этапов:

1. Определение порядка соединения выводов внутри цепи (построение кратчайшего связывающего дерева - КСД).
2. Распределение соединений по слоям МПП.
3. Нахождение последовательности проведения соединений в каждом слое.
4. Получение конфигурации проводников.

Первый этап: Определение порядка соединения выводов внутри цепи.

Задача сводится к построению КСД. При печатном монтаже соединения можно выполнять не только по эквипотенциальным выводам, но и в любой точке проводника, поэтому построение КСД здесь формируется как задача Штейнера:

К множеству $P=\{P_1,P_2,...,P_n\}, i=\overline{1,n}$ основных точек добавить множество $S=\{S_1,S_2,...,S_m\}$ дополнительных точек и построить КСД. Множество ***P*** основных точек сопоставлено выводам цепи, а дополнительные точки представляют собой места соединений типа проводник-проводник. При определении положения дополнительных точек можно рассматривать только узлы координатной сетки, построенной на ***n*** заданных точках. Тогда число таких точек $Q \le n$.

Метод точного решения задачи Штейнера для реальных цепей требует больших затрат машинного времени.

Второй этап: Распределение соединений по слоям.

В результате выполнения первого этапа трассировки электрическая цепь представляется КСД, являющимся плоским графом. Но совокупность КСД (МС) может иметь пересечения между ребрами, принадлежащими разным деревьям, так как последние строятся на фиксированных вершинах, и существуют ограничения на

размер монтажного поля, ширину проводников и зазоры между ними, в то же время в каждом слое печатные проводники не должны пересекаться.
При ортогональной трассировке возможно распределение соединений по двум слоям, при этом каждая цепь представляется в виде ортогонального покрывающего дерева, вертикальные ветви которого находятся в одном слое, а горизонтальные – в другом. На рисунках показано КСД (1а) и ортогональное Штейнерово дерево (1б).

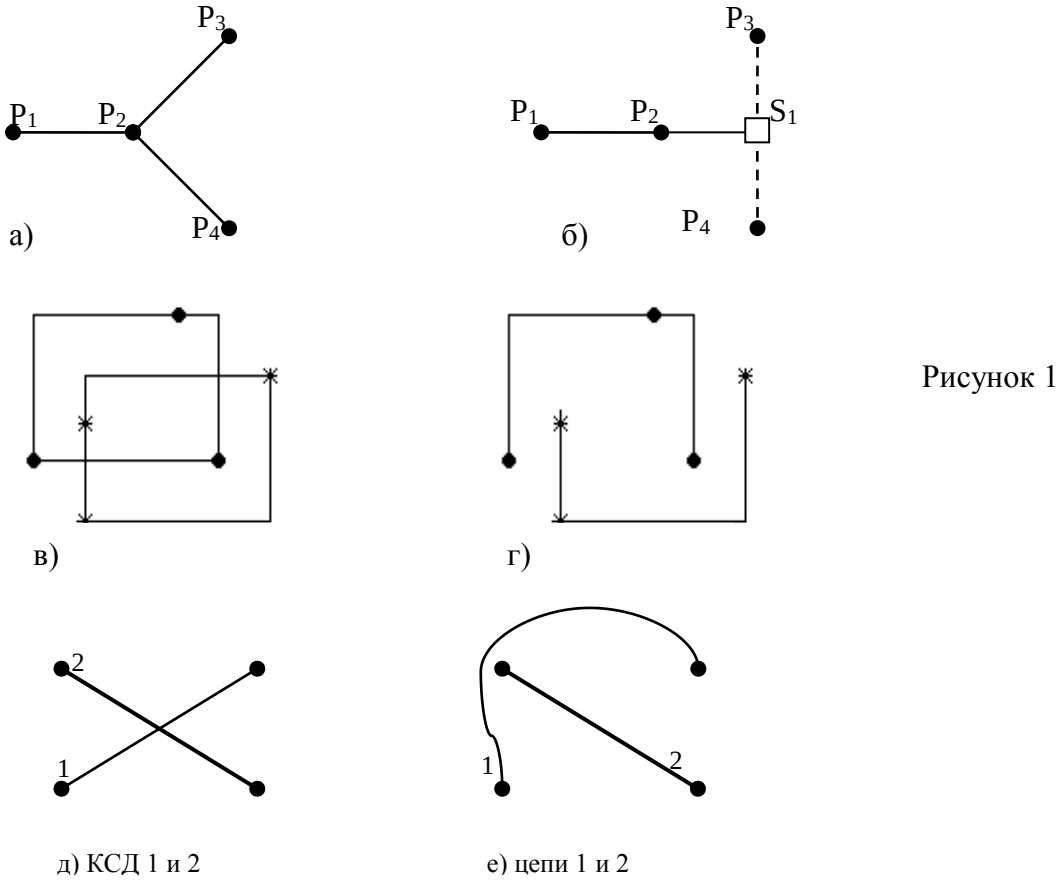


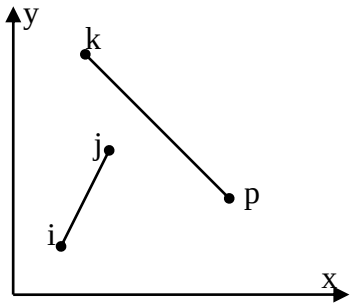
Рисунок 1

д) КСД 1 и 2

е) цепи 1 и 2

В узлах дерева Штейнера необходимо делать межслойные переходы, при этом количество переходов может быть весьма большим, что ухудшает механические параметры печатной платы и снижает надёжность схемы.
В общем случае распределение соединений по слоям может быть сформулировано как задача правильной раскраски вершин графа пересечений (см. ниже).
Предполагается, что соединение полностью выполняется в 1 слое. При ортогональной трассировке на вершинах каждой цепи строится минимальный охватывающий прямоугольник (рис. 1в). Считается, что два соединения пересекаются, если перекрываются соответствующие им многоугольники.
При представлении цепи КСД необходимо определять пересекается ли каждая пара ветвей этих деревьев. Для пары ветвей при известных координатах вершин составляются уравнения прямых линий и исследуются эти уравнения, методами аналитической геометрии определяют возможность пересечения соответствующих соединений. Вершины графа пересечений сопоставляются соседям, ребра устанавливают возможность пересечения. Раскраска будет правильной, если никакие смежные вершины не окрашены одним цветом. Минимальное число цветов, которое необходимо для правильной раскраски, определяет требуемое число слоев платы.

Отметим, что перекрытие многоугольников, построенных на вершинах цепей или пересечений КСД, ещё не означает, что соответствующие цепи нельзя трассировать на одном слое без пересечений.
На рис.1г показаны перекрывающиеся прямоугольники и пересекающиеся цепи, соединяющие охватываемые ими вершины, пересекающиеся КСД (1д) и не пересекающиеся трассы, соединяющие те же вершины.
При учете возможности проведения конфликтующих проводников без пересечения за счет огибания, распределение соединений по слоям может быть сделано путем объединения проводников, идущим под некоторым углом друг к другу в группы. Каждая такая группа затем трассируется в своем слое. Например, при ортогональной трассировке ребро U_{ij} КСД относится к группе горизонтальных, если $|x_i - x_j| \geq |y_i - y_j|$, и к группе вертикальных в противном случае.
Для МПП проводники могут группироваться в соответствии с их принадлежностью некоторым секторам, заданным для каждого слоя.
Третий этап: Задача нахождения последовательности проведения соединений в каждом слое.
Трассировка соединений выполняется последовательно, и каждая проложенная трасса является препятствием для всех не проведенных.
Сформируем условия отсутствия пересечения двух ребер и методику определения последовательности их проведения.



Рассмотрим два ребра U_{ij} и $U_{кд}$. Их уравнение в параметрической форме:

$$\left.\begin{aligned} x_{пер} &= \lambda x_i + (1-\lambda) x_j \\ y_{пер} &= \lambda y_i + (1-\lambda) y_j \end{aligned}\right\}$$

$$\left.\begin{aligned} x_{пер} &= \mu x_к + (1-\mu) x_к \\ y_{пер} &= \mu y_д + (1-\mu) y_д \end{aligned}\right\},$$

где

$$\lambda = \frac{(x_к - x_д)(y_j - y_д) - (y_к - y_д)(x_j - x_д)}{A}$$

$$\mu = \frac{(x_j - x_д)(y_j - y_i) - (y_j - y_д)(x_j - x_i)}{A}$$

$$A = (x_к - x_д)(y_j - y_i) - (y_к - y_д)(x_j - x_i)$$

Ребра пересекаются, если $0 \leq \lambda \leq 1,\ 0 \leq \mu \leq 1$.

Пример: 1) $x_i=1,\ y_i=1,\ x_k=3,\ y_k=6,$
$x_j=8,\ y_j=8,\ x_p=7,\ y_p=4,$

λ =3/7; μ =0,5 – прямые пересекаются

2) x_i=1, y_i=1, x_k= 3, y_k=6,

x_j=4, y_j=4, x_p=7, y_p=4,

λ = -1/3 (<0); μ =0,5 – прямые не пересекаются

На основании выражения (1) определяется список пересекающихся ребер. Непересекающиеся ребра можно трассировать в произвольном порядке. Для определения последовательности проведения пересекающихся ребер составляют уравнения удлинения при огибании, считая, что огибающий проводник может проходить сколь угодно близко от вершины. Эти уравнения составляются для всех пар пересекающихся ребер. Для каждого ребра подсчитывается число огибаний и удлинение. Список ребер ранжируется в порядке возрастания числа огибаний. Если у некоторых групп ребер число огибаний одинаково, то первыми проводятся ребра с меньшим удлинением.

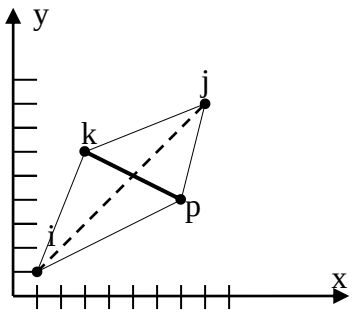


Рис.2

На рисунке 2 изображены два ребра. Если первым провести ребро H_{ij}, то удлинение второго ребра l_{kp} при огибании вершин i и j будет определяться:

$$\Delta l_{kp}^{i} = l_{ki} + l_{ip} - l_{kp},$$

$$\Delta l_{kp}^{j} = l_{kj} + l_{jp} - l_{kp};$$

Так как пересечение рассматривается только для пары ребер, то необходимо дополнительно проверять отсутствие пересечений с другими близлежащими ребрами.

Четвертый этап: Трассировка отдельных соединений.

Чаще всего эта задача решается в два этапа. В начале осуществляется предварительная приближенная трассировка, а затем окончательная.

Приближенная трассировка представляет собой стадию планирования окончательной трассировки, на которой определяется, по каким областям (каналам) должны следовать отдельные соединения. Предварительная трассировка осуществляется на макромодели.

Цель её – равномерно распределить соединения без превышения пропускной способности областей (каналов). Чаще всего она осуществляется за два прохода.

На первом проходе допускается превышение пропускной способности, и в расчет принимается только кратчайшие расстояния между соединениями. После первого выявляются критические каналы области, и в расчет принимаются только кратчайшие расстояния между соединениями. После 1-го прохода выявляются критические каналы (области) и регистрируются превышения пропускной способности.

Второй этап обычно состоит из нескольких итераций, на каждой из которых вводятся прогрессирующие штрафы за превышение пропускной способности.

В результате выполнения приближенной трассировки окончательная трассировка распадается на ряд боле простых задач, имеющих меньшую размерность. Методы окончательной трассировки могут быть разбиты на три группы:

1. Волновой алгоритм и его модификации.
2. Алгоритм трассировки по магистральным каналам.
3. Комбинированные алгоритмы.

Математические модели монтажного пространства

Под монтажной областью типовой конструкции понимается метрическое пространство, в котором устанавливаются входящие в неё типовые конструкции предыдущих уровней, и выполняется электронное соединение выводов. Формальная постановка и решение задач в топологическом конструировании невозможны без получения математической модели монтажного пространства. К которой предъявляются требования высокой степени формализации и наиболее точного и полного отображения метрических параметров и топологических свойств конструкции.

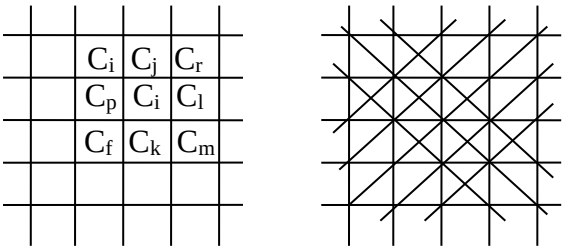
Метрическими параметрами являются габаритные размеры зоны монтажа, допустимая ширина проводников и зазоры между ними, координаты и размеры внешних контактных площадок, шаг установки и размер модулей, координаты и размеры полей их контактов.

К топологическим свойствам относятся: число слоев МПП и переходов со слоя на слой;

наличие замкнутых областей, запрещенных для проведения соединений; ограничение на взаимное расположение соединений; количество монтажных проводов, проводимых к одному выводу цепи.

В качестве математической модели монтажного пространства используется неориентированный топологический граф G_r (граф решетки). При этом плоскость монтажного пространства разбивается на элементарные площадки, стороны которых равны шагу проложения проводника по соответствующему направлению. Для печатного монтажа элементарная площадка – квадрат. Каждой элементарной площадке ставится в соответствие вершина графа решетки. Две вершины графа соединяются ребром, если между соответствующими площадками может быть проведено соединение с учетом метрических и топологических параметров конструкции.

Выделяют ортогональный монтаж (а) и 45°-монтаж (б):



а) б)

ВОЛНОВОЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ ТРАССИРОВКИ

Большинство алгоритмов конфигурации печатных проводников используют идею волнового алгоритма Ли, который представляет собой процедуру нахождения кратчайшего пути в графе.

В работе Ли плоскость монтажа разбивается на элементарные прямоугольники (ячейки), со стороной равной расстоянию между осями соседних печатных проводников. Минимальная величина прямоугольника (ячейки) определяется технологическим оборудованием.

При использовании ДРП, включение элементарной ячейки в путь означает проведение печатного проводника, так как показано на рисунке 1.

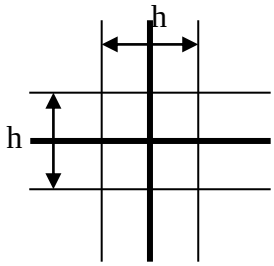


Рис. 1

Считаем, что основная координатная сетка смещена на h/2, чтобы пути следовали из ячейки в ячейку, а не по координатным линиям ДРП.

На каждом шаге алгоритма некоторые ячейки ДРП заняты. К ним относятся ячейки, попадающие в область запрещённых для трассировки: краевые поля платы, зоны размещения элементов и их выводы, ранее проведённые проводники.

Основа алгоритма Ли – процедура поиска оптимального, в смысле некоторого критерия, пути между двумя ячейками, при соблюдении ряда условий:

A

×

B

Первая часть алгоритма моделирует процесс распространения волны от элементарной площадки A по свободным ячейкам ДРП. При распространении волны от A, алгоритм последовательно строит $\Phi_1(A)$, $\Phi_2(A)$, ..., $\Phi_k(A)$ – её фронты. Множество ячеек, входящих в i-е фронты для всех $i \leq k$, называют k-й окрестностью ячейки A. $O_k(A)$-окрестность.

Если проведение пути возможно, то на $k+1$ шаге окажется, что $B \in O_{k+1}(A)$. Если в следующий фронт не удаётся включить ни одной свободной ячейки, т.е. $O_k(A) = O_{k+1}(A)$, то при данных условиях путь провести невозможно.

Т.о. первая часть алгоритма Ли определяет возможность проведения пути между ячейками A и B.

Во второй части алгоритма Ли, начиная с B, по определённым правилам выполняется переход от ячейки k-го фронта к ячейке k-1-го фронта, до тех пор, пока не будет достигнута A. Пройденные ячейки – искомый путь.

Условия, которые должны быть выполнены при проведении пути, и возможность оценки его оптимальности должны быть заложены в правила, по которым происходит движение фронта.

Для ячеек ДРП устанавливается отношения соседства. Ячейки ДРП считаются соседними, если они имеют общее ребро. Распространение волны заключается в присваивании соседним ячейкам значения весовой функции. Вес ячейки k-го фронта P_k является функцией веса ячейки k-1 фронта. В общем случае к весам ячеек предъявляются следующие требования: $P_{k-1} \neq P_k \neq P_{k+1}$, в большинстве модификаций алгоритма Ли на веса ячеек накладывается ограничение: $P_k > P_{k-1}$. В этом случае проведение пути заключается в переходе от B к A, т.о. чтобы P_k монотонно убывало. При этом возможен вариант, когда несколько ячеек, соседних с данной, имеют одинаковый вес. Для однозначного выбора при учёте критерия минимального

количества изгибов проводника следует сохранить направление движения. Если приходится делать поворот, то учитываем заранее заданный порядок предпочтений.

Например, пусть $P_k = P_{k-1}+1$ – т.е. равно расстоянию *k*-й ячейки от исходной ячейки *A* в ортогональной метрике (с учётом запрещённых ячеек). Работа алгоритма для этого случая приведена на рисунке:

	1	2	3	4	5	6	7	8	9	10	11	12
1	4	3	2	3	4	5	6	7	8	9	10	#
2	3	2	***1***	***2***	***3***	***4***	***5***	6	7	8	9	#
3	2	1	**A**	1	2	#	***6***	7	8	9	10	#
4	3	2	***1***	2	3	#	***7***	8	9	10	11	#
5	4	3	***2***	3	4	#	***8***	9	10	#	#	#
6	5	4	***3***	4	5	#	***9***	***10***	***11***	**B**		
7	6	5	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***		
8	7	6	5	6	7	8	9	10	11			

Рис. 2.

Волна распространяется из *A*, $P_A = 0$. Фронт волны доходит до B на 12-м шаге. В ходе построения пути из ячейки (*6,9)* с $P=11$, можно перейти в 3 соседние ячейки с $P=10$. здесь переход осуществляется сохраняя направление влево – вверх, аналогично из ячейки с $P=10$, у ячейки $P=9$, 2 соседних ячейки $P=8$, т.к. в этом случае приходится изменять направление движения, то переходим по предпочтённому направлению – вверх. Т.к. вес *k*-й ячейки в данном варианте равен расстоянию от *A*, то найденный путь – оптимальный, в смысле его длины. Т.к. алгоритм Ли представляет собой алгоритм нахождения кратчайшего пути в графе, то он легко распространяется на многослойный печатный монтаж, при использовании модификации в виде графа. При наличии ограничения на переходы со слоя в слой можно увеличить вес ребра, соединяющего две смежные вершины графа на соседних слоях, по сравнению с весом ребра графа соединяющего вершины в одном слое. В общем случае весовая функция может зависеть от параметров, учитывающих длину пути, числа переходов из слоя в слой, степень близости пути к другому и т.д. Например, в виде аддитивной функции:

$$P_k = \sum_{i=1}^{n} a_i P_i(k)$$

a_i – весовой коэффициент, учитывающий важность i-го параметра.
$P_i(k)$ – значение учитываемого параметра.

Усложнение функции веса увеличивает объём информации на одну ячейку ДРП и время работы 1-й части алгоритма. Кроме того, не представляется возможным строго обосновать выбор значений a_i. При практической реализации алгоритма Ли важная проблема – сокращение объёма памяти ЭВМ, необходимого для запоминания весов ячеек ДРП. При вычислении весов ячеек по выражению $P_k=P_{k-1}+1$ячейка может быть в следующих состояниях: свободна, занята или имеет вес от *1* до *L* (максимально возможная длина пути, определяемая как количество ячеек ДРП в пути).

Необходимое, для запоминания состояния одной ячейки, число 2-х разрядов памяти:

$$N = \lceil \log_2(L+2) \rceil_{б.б.ц.}$$

Наиболее эффективным способом кодирования состояния ячеек ДРП является метод путевых координат по модулю 3. При выборе последовательности ячеек на этапе построения пути по методу путевых координат для каждой ячейки, начиная с *B* (в случае соседства по рёбрам), достаточно знать, от какой соседней ячейки в неё пришла волна: ↓(3), →(2), ↑(1), ←(4). Т.о. ячейка здесь может иметь следующие признаки: свободна, занята, одна из путевых координат, следовательно, число разрядов для кодирования состояния ячеек равно $\log_2 6 = 3$.

Путевой координатой C_i фронта Φ_k является та соседняя с ней ячейка C_j фронта Φ_{k-1}, от которой C_i получила свой вес.

При использовании метода путевых координат соседним ячейкам присваиваются веса в соответствии с пожеланиями пользователя. При проведении пути следует двигаться направлении противоположном распространению волны, в соответствии с путевыми координатами.

Кодирование весов ячеек по mod3

Оно базируется на том, что $P_{k-1} \neq P_k \neq P_{k+1}$. Ячейкам, включённым в последующие фронты, можно присваивать не сами веса, а их значения по *mod3 (1,2,3,1,2,3…)*. Кол-во разрядов на кодирование состояний ячеек равно $N = \log_2 5$. На практике используется понятие вертикальной и горизонтальной полузанятости ячейки, эти состояния также нужно кодировать.

Проведение пути заключается в отслеживание отметок *(1,2,3)*. Если ячейка имеет несколько соседних ячеек с одинаковыми метками, то используется правило приоритетных направлений.

При движении от ячейки *B* на рис. 3 используется следующие правило приоритетов:←, ↑, →, ↓.

	1	2	3	4	5	6	7	8	9	10	11	12
1	2	1	3	2	3	1	2	3	1	2	3	#
2	1	3	2	***1***	***2***	***3***	***1***	***2***	***3***	1	2	#
3	3	2	1	**A**	1	2	#	3	***1***	2	3	#
4	1	3	2	1	2	3	#	1	***2***	3	1	#
5	2	1	3	2	3	1	#	2	***3***	#	#	#
6	3	2	1	3	1	2	#	3	***1***	**B**		
7	1	3	2	1	2	3	1	2	3	1		
8	2	1	3	2	3	1	2	3	1			

Рис. 3.

Методы ускорения работы волнового алгоритма.

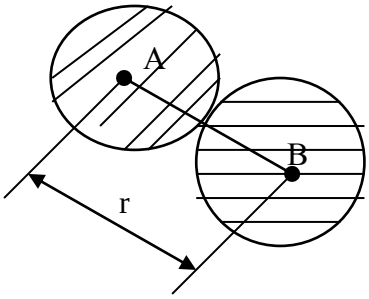
Волновой алгоритм характеризуется высокой эффективностью нахождения пути за счёт исследования всех свободных ячеек ДПР, но требует значительного времени t на распространение волны.

Используются различные методы ускорения выполнения 1-го этапа алгоритма. Одним из них является правильный выбор начала точки трассы (рис.1).

Рис.1.

Из рассмотрения рис.1 видно, что при выборе в качестве источника распространения волны ячейки А, максимально удалённой от центра платы, просматривается меньшее число свободных ячеек ДПР. Это становится очевидным по мере роста числа протрассированных цепей.

Более эффективен метод встречной волны (рис.2)



$M = \dfrac{\Pi r^2}{2(\dfrac{\Pi r^2}{4})} = 2$ - выигрыш в количестве ячеек.

Для реальных состояний ДРП выигрыш во времени будет отличаться от 2. Однако, в среднем, оценка является объективной. Использование данной идеи приводит к усложнению алгоритма.

Поле распространения волны можно уменьшить, ограничивая его охватывающим прямоугольником, внутри которого находятся ячейки А и В. Начальная площадь прямоугольника (печатной платы) обычно на 10-20% больше площади прямоугольника, проходящего через ячейки А и В. Если при этом соединение найти не удалось, то границы охватываемого прямоугольника расширяются. Данный метод обладает большей эффективностью ускорения работы алгоритма по сравнению с методами 1 и 2.

АЛГОРИТМ РАБИНА

Увеличение быстродействия в алгоритме Рабина достигается за счёт преимущественного распространения волны в направлении ячейки-цели.

В алгоритме Ли, который является реализацией метода динамического программирования, такая целенаправленность отсутствует.

Алгоритм Рабина, который реализует метод ветвей и границ, позволяет так же, как и алгоритм Ли, находить глобальный оптимум целевой функции (L_{min}).

Пусть требуется найти путь min длины между ячейками A и B.

Рассмотрим некоторую ячейку C_i , включаемую в очередной фронт волны. Значение весовой функции P_i , приписываемое C_i ячейкой фронта, определяется из выражения:

$P_i = L(i, A) + H(i, B); (1)$,

где $L(i,A)$ – длина пройденного волной пути из ячейки A в ячейку C_i , а

$H_{i,B} = | x_i - x_B | + | y_i - y_B |; (2)$ - расстояние между C_i и B в ортогональной метрике.

Нетрудно увидеть, что $H_{i,B}$ является нижней оценкой пути из C_i в B, полученной в предположении отсутствия препятствий на этом пути.

Отсюда, выражение (1) в целом представляет собой нижнюю оценку пути из A в B, проходящего через ячейку C_i.

Согласно методу ветвей и границ, оптимальное решение следует искать в подмножестве решений, имеющем наилучшую оценку. В соответствии с этим в сформированном очередном фронте волны выделяют подмножество ячеек с min оценкой 1, и распространение волны на следующем шаге осуществляется из ячеек этого подмножества.

Заметим, что для сокращения зоны поиска, распространение волны в алгоритме Рабина осуществляется в первую очередь, из тех ячеек с min оценкой, которые получили эту оценку последними.

Последовательность просмотра соседних ячеек та же, что и в алгоритме Ли; построение пути осуществляется с использованием путевых координат.

Эффективность алгоритма Рабина по сравнению с алгоритмом Ли показана на рис.3, на котором для каждой ячейки ДПР указано значение весовой функции P_i. Цифрами в правых углах ячеек обозначена последовательность рассмотрения ячеек на этапе распространения волны. Стрелками указаны путевые координаты. Волна распространяется от ячейки с минимальным значением весовой функции и большим порядковым номером.

8		15	13	11	11→	11→	11→	13		
7	15	←13↑	←11↑	←**A**↑ ↓→	9↑	11	11	13		
6	15	←13	←11	←9→	9→	9↑→ ↓	9↑→	11↑→	13	
5		#	#	#	#	9↓→	9→	11→	13	
4			15	←13	←11	←9→	9→	11→	13↓→	15
3				#	#	#	#	#	13↓→	15
2							13↓	←13↓	←↑13 ↓→	15
1							**B**	13	13	
	1	2	3	4	5	6	7	8	9	10

Рис.3.

АЛГОРИТМ СЛЕЖЕНИЯ ЗА ЦЕЛЬЮ

Возможно дальнейшее сокращение зоны поиска при трассировке в случае приближений реализации метода ветвей и границ, получивших название «алгоритм слежения за целью». В этом алгоритме в качестве значений весовой функции для ячейки C_i на этапе распространения волны используется выражение $H_{i,B} = |x_i - x_B| + |y_i - y_B|$

Примечание: волна распространяется от множества ячеек с минимальным весом, начиная с ячейки с большим/меньшим порядковым номером. Проведение пути с помощью алгоритма слежения за целью основано на использовании метода путевых координат.

Здесь распространение волны происходит из тех ячеек очередного фронта, которые ближе расположены к ячейке – зоне (запрещённые ячейки не учитываем). При этом сокращается зона поиска. Однако, из-за приближённого вычисления нижней

оценки для подмножества путей, ведущих из *A* в *B* через *C*-пути, возможна потеря глобального минимума.

Пример:

8				10	9	8				
7			10	←**A**↑ ↓→	↑8→	↑7↓ →	6			
6			9	←8→	7	6↓→	5			
5		#	#	#	#	5↓→	4	5	6	
4					5	←4→	3→	4↑→	5↑→ ↓	6
3				#	#	#	#	#	4↓→	5
2							1↓	←2↓	←3→ ↓	4
1							**B**	1	2	
	1	2	3	4	5	6	7	8	9	10

Примечание: алгоритм Ли требует просмотра 84 ячеек, алгоритм Рабина – 44 ячейки, алгоритм слежения за целью – 35 ячеек.

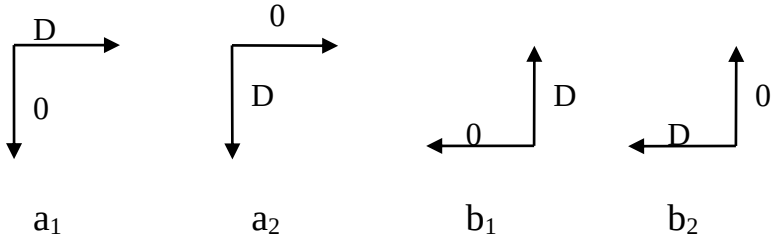
ЛУЧЕВОЙ АЛГОРИТМ ТРАССИРОВКИ АБРАЙТИСА

В данном алгоритме уменьшение временных затрат достигается за счет распространения волны не по всему полю платы, а вдоль определенных лучей. Идея этого алгоритма состоит в том, что при распространении волны фронт занимает не все свободные ячейки ДРП, а одну, выбранную по определенным правилам. Таким образом, последовательность фронтов представляет собой луч.

Построение пути в лучевом алгоритме осуществляется с использованием метода путевых координат. В двухлучевом алгоритме из начала *A* и конца *B* трассы распространяются навстречу друг другу 2 луча: a_1, a_2, b_1, b_2.

Лучи $a_1 b_2$ и b_1 a_2 называются разноименными. Лучи поочередно распространяются (шагают) до тех пор, пока разноименные лучи не пересекаются, или пока 4 луча не смогут продвинуться дальше (тупик). В первом случае осуществляется переход к проведению трассы. Во втором случае трассу провести невозможно.

Рассмотрим работу алгоритма на примере. Прежде всего можно выбрать некоторые направления движения лучей:



8										
7						↑	→	→	→	
6						↑	#	#	↑	

5		**A**	→^{A_2}	→	→	→↑	#	↓↑	↑	
4		↓^{A_1}				↓↑	#	↓←	**B**	
3		↓				↓↑	#	↓		
2		↓	#	#	#	#	#	↓		
1		↓	←	←	←	←	←	↓		
	1	2	3	4	5	6	7	8	9	10

РАСПРЕДЕЛЕНИЕ СОЕДИНЕНИЙ ПО СЛОЯМ МНОГОСЛОЙНОЙ ПЕЧАТНОЙ ПЛАТЫ

Многослойный монтаж печатных (плёночных) соединений между компонентами электронной схемы позволяют сократить суммарную длину соединений ($\tau_{зад.\ min}$), уменьшить вес и габариты проектируемой аппаратуры.

При разработке многослойных структур возникает задача распределения по отдельным слоям печатных плат схемы соединений, претендующих на одни и те же области монтажного пространства (конфликтующих между собой).

Целью решения этой задачи является обеспечение 100% разводки соединений, уменьшение числа слоёв и межслойных переходов, наиболее рациональное использование объёма монтажного пространства.

Алгоритмическое распределение соединений по слоям может выполняться до, после или в процессе трассировки отдельных соединений.

Расслоение до трассировки связано с анализом схемы соединений с целью выявления тех соединений или групп соединений, распределение которых в 1 слой неизбежно приведёт к возникновению пересечений на этапе трассировки.

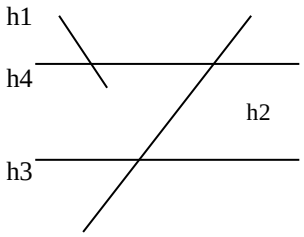
Наиболее общий подход к такому анализу предполагает использование топологических методов, позволяющих на основе исследования свойства планарности графа (гиперграфа) схемы выявить возможность её реализации без пересечений в заданном (минимальном) числе слоёв.

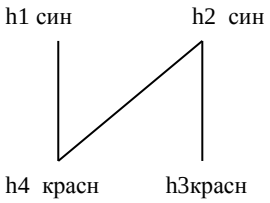
Несмотря на то, что требование планарности графа схемы является решающим условием её плоской реализации, тем не менее, ограничение метрического характера (ограниченные размеры конструкции, контроллера, длины проводников и т.д.) создают трудности при практическом использовании топологического подхода к рассмотрению платы.

Большинство приближенных алгоритмов расслоения до трассировки связана с введением определенных количественных оценок конфликтуемости цепей при их распределении в 1 слой. Эти оценки вычисляются на основе информации о расположении элементов и контактов цепей, полученных после решения задач размещения. Такого рода оценки могут зависеть от степени перекрытия минимальных прямоугольников, охватывающих контакты отдельных цепей, числа контактов цепей и т.д. После определения степени конфликтуемости каждой пары цепей строится взвешенный неор. граф конфликтов $G=(H,V)$, в котором каждой вершине $h\in H$ соответствует некоторая цепь, а каждому ребру $v_{i,j}\in V$ соответствует наличие конфликта между цепями H_i и H_j. Вес ребра принимается равным оценке конфликтуемости (числу пересечений) между соответствующими цепями. Далее осуществляется раскраска вершин графа G в заданное число цветов, при котором

сумма весов ребер, соединяющих вершины одного цвета, минимальна. Раскраска вершин графа конфликтов осуществляется различными эвристическими алгоритмами.

Пример:





К=2(требуемое число слоев)

1. Хроматическое число графа $\gamma(j)$ указывает наименьшее количество цветов, с помощью которых можно окрасить его вершины таким образом, чтобы в нем не было ни одной дуги, которая соединяла бы вершины одного и того же цвета.

Очевидно, что число слоев многослойной печатной платы равно хроматическому числу графа пересечений. Для примера $\gamma(j)=2$. Следовательно, для 100%-й трассировки достаточно 2 слоев. В общем случае для $\gamma(j)$ отсутствуют простые формулы, и возможна лишь оценка. Если число ребер m удовлетворяет условию $(n-1)\le m\le 0.5n(n-1)$, где n – число вершин графа, то оценка сверху от $\gamma(j)$ будет иметь вид: $\gamma(j)\le\dfrac{3+\sqrt{9+8(m-n)}}{2}$.

Теорема: пусть $\rho_{\max}$ - это максимальная степень вершин графа (число ребер, принадлежащих какой-то вершине). Для большинства графов $\gamma(j)\le\rho_{\max}$.

2. Расслоение после предварительной трассировки состоит в следующем. На 1-м этапе с помощью одного из алгоритмов построения кратчайших связанных деревьев (КСД) осуществляется предварительная трассировка соединений в одном слое. В процессе получения совмещенной топологии минимизируются такие геометрические размеры, как длина, число перегибов, число пересечений. Далее так же, как и в алгоритме расслоения до трассировки, строится граф конфликтов, но не между цепями, а между отдельными двухконтактными соединениями (отрезками проводников). При этом граф конфликтов *G=(X,V)* является неориентированным графом, наличие ребра в котором между x_i и y_j соответствует пересечению отрезков в совмещенной топологии.

Вершина ГК раскрашивается в заданное минимальное число цветов так, что вершина первого цвета не имеет ни 1 единого ребра. Каждое подмножество, соответствующее вершинам 1-го цвета, распределяется в 1 из слоев. Не вошедшие ни в 1 из подмножеств отрезки (что может иметь место при заданном К), последовательно распределяются в те слои, в которых они имеют минимальное число пересечений.

3. Расслоение в процессе трассировки осуществляется одновременно с прокладкой соединений в многослойном ДРП с помощью различных модификаций волнового алгоритма трассировки Ли. Рассмотрим математическую постановку задачи по слоям после предварительной трассировки. Будем считать, что число слоев – *k*. Пусть *G=(X,Y)* – граф пересечений, в котором каждой вершине x_i соответствует отрезок I, $i=\overline{1,n}$ а ребру (x_ix_j) соответствует *n* отрезков *i* и *j* в совмещенной топологии.

Требуется распределить отрезки по слоям, т.о., чтобы суммарное число пересечений было минимальным.

Введем в рассмотрение матрицу $Y=\|y_{ij}\|_{n\times k}$,

где $y_{ij} = \begin{matrix} 1, \text{ если отрезок i назначается в слой j} \\ 0, \text{ в противном случае} \end{matrix}$

При таком подходе задача расслоения может быть сформулирована:

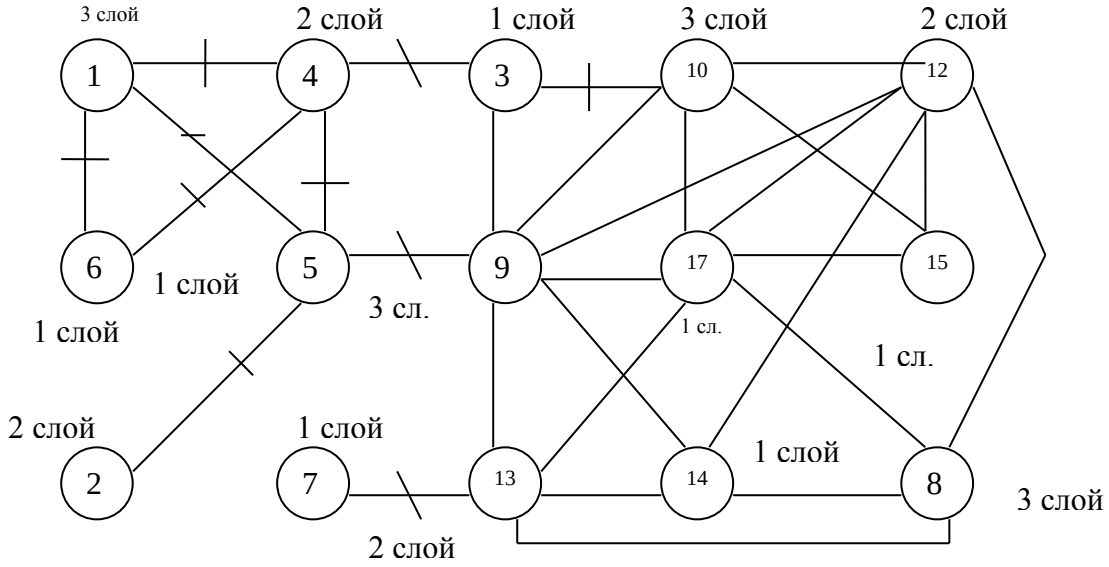
Требуется минимизировать $\hat{O}_{\min} = \sum_{j=1}^{k} \sum_{i=1}^{n-1} \sum_{p=1}^{n} c_{ip} \times y_{ij} \times y_{pj} (*)$, где c_{ij} – соответствующий элемент матрицы смежности графа G при ограничении $\sum_{j=1}^{k} y_{ij} = 1 \ (i = \overline{1,n})$ - каждый отрезок может быть расположен только в одном слое.

Данная задача является квадратичной задачей целочисленного программирования с булевыми переменными (y_{ij},y_{pj}) Рассмотрим приближенный алгоритм ее решения.

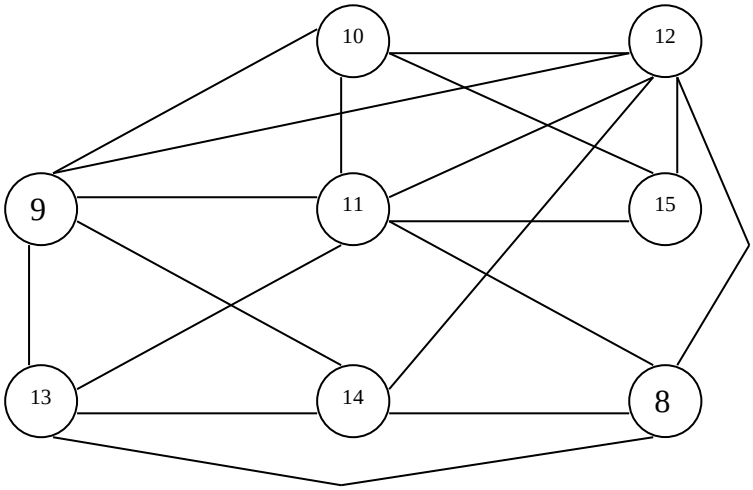
Алгоритм расслоения МПП

1. В соответствии с результатами предварительной трассировки строится граф конфликтов отрезков $G = (X, V)$, соответствующий топологии схемы.
2. Из графа конфликтов последовательно удаляются вершины, имеющие локальную степень (число ребер, инцидентных вершине) <K. Удаление вершин здесь и далее осуществляется вместе с инцидентными им ребрами. Формируется список таких вершин. Очевидно, что отрезки, соот-ие таким вершинам, могут быть реализованы в слоях К без пересечений.
3. Формируется список S_1 отрезков, распределенных в 1-й слой. С этой целью в графе $G_1=(X_1,V_1)$, где $X_1=X\backslash S$; $V_1=V\backslash V_s$, где V_s – множество ребер, инцидентных вершинам множества S, находится максимальное число не смежных между собой вершин. При этом используется приближенная процедура. Из графа G_1 последовательно удаляются вершины с максимальными локальными степенями до получения графа без ребер. Множество оставшихся при этом вершин графа включается в список S_1. Очевидно, что соответствующие этому списку отрезки между собой не конфликтуют и могут быть расположены в 1-м слое без пересечений.
4. Далее из графа G_1, из которого исключены вершины списка S_1, последовательно удаляются и заносятся в список S вершины, имеющие локальную степень $<(k-1)$. Отрезки, соответствующие этим вершинам, всегда могут быть расположены в $(k-1)$ слой (кроме 1-го) без пересечений. Получаем граф $G_2=(X_2,V_2)$ $X_2=X\backslash(SUS_1)$; $V2=\backslash(V_sUV_{s1})$
5. Аналогично формируются $S_1S_2,S_3,..,S_k$. При этом может остаться некоторое множество отрезков S^*, которое нельзя реализовать без пересечений в 1-м слое.
6. Из списка S последовательно в порядке, обратно порядку включения в список, выбирается очередной отрезок и включается в тот слой, в котором он не имеет пересечений. Так слой обязательно найдется, что следует из процедуры формирования списка S.
7. Из списка S^* последовательно выбирается очередной отрезок и распределяется в тот слой, где имеет минимальное число пересечений. Если слоев несколько, то отрезок распределяется в слой с меньшим общим числом отрезков.

Пример: Пусть $k=3$. Граф конфликтов:

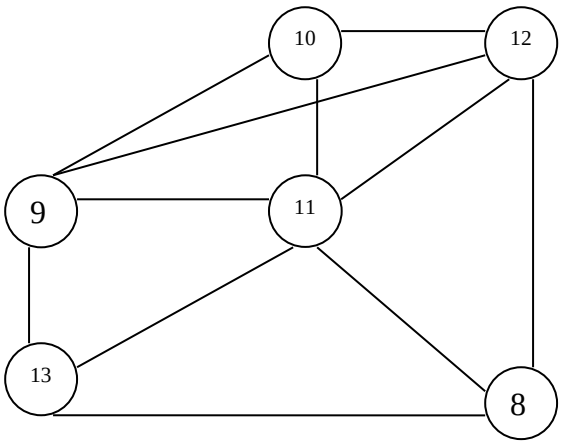


Последовательно удаляем вершины локальной степени, кот.<3. Формируем *S={2,6,1,4,3,5,7}*. Строим $G_1=(X_1,V_1)$, представленный на рисунке:



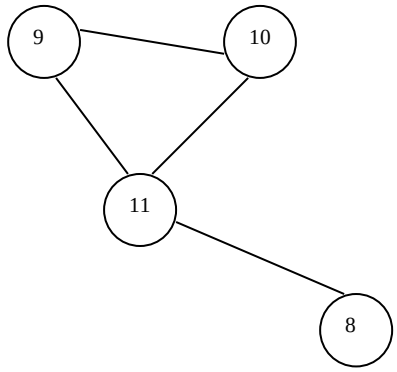
Из графа G1 последовательно удаляем вершины с максимальной локальной степенью. Сначала исключается вершина 11, затем 12,13,9,8,10. Вершины списка S_1={14,15} образуют внутреннее устойчивое множество, т.е. максимальное несмежное между собой число вершин.

Отрезки 14 и 15 списка S_1 помещаем в первый слой платы. Из графа G_1 удаляется вершина S_1; получаем граф G_2:



Список S здесь не дополняется, т.к. отсутствуют вершины локальной степени, которые <(K-1)=2 (п.4)

Из графа G_2 последовательно удаляются вершины с максимальной локальной степенью (11,9,8,10), и получаем $S_2=\{12,13\}$. Назначаем им 2-й слой. Строим G_3(без 12 и 13). Из G_3 находим $S_3=\{8,10\}$, которое распределяем в 3-й слой.



Вершина 9 и и11 включается в S*. Далее последовательно(в обратном порядке) распределяем по слоям отрезки из списка S.
7,5,3- в 1-й слой; 4 – 2-й слой; 1 – 3-й слой; 6 – 1-й слой; 2 – 2-й слой;
Распределяем изS*: 9 – 3-й слой(1-е пересечение); 11 – 1-й слой(1-е пересечение)

Т.о. расположение по слоям следующее:
S1={14,15,7,5,6,3,11}
S2={12,13,4,2,}
S3={8,10,1,9}

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Механизм генетического наследования. В каждой клетке любого животного содержится вся генетическая информация особи. Эта информация записана в виде набора молекул ДНК, каждая из которых представляет собой цепочку из молекул нуклеотидов четырех типов: А, Т, Ц, Ж. Собственно информацию несет порядок следования нуклеотидов в ДНК.

Таким образом, генетический код особи - это длинная строка, где используются четыре символа: А, Т, Ц, Ж. В животной клетке каждая молекула ДНК окружена оболочкой. Такое образование называется хромосомой.

Каждое врожденное качество особи (цвет глаз, наследственные болезни, тип волос и т. д.) кодируются определенной частью хромосомы, которая называется геном этого свойства.

Различие значения гена называется аллелями.

При размножении особи происходит слияние двух родительских клеток, и их ДНК взаимодействуют, образуя ДНК потомка.

Основной способ взаимодействия – кроссовер-скрещение. При кроссовере ДНК предков делится на две части, а затем обменивается своими половинами. При наследовании возможны мутации, в результате которых могут измениться некоторые гены в клетках одного из родителей. Измененные гены передаются потомку и придают ему новые свойства. Если эти новые свойства полезны они, скорее всего, сохранятся в данном виде. При этом произойдет скачкообразное повышение приспособляемости вида.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Генетические алгоритмы – это последовательность управляющих действий и операций моделирующие эволюционные процессы на основе аналогов механизмов информационного наследования и естественного отбора.

Генетические алгоритмы на поиске лучших решений с помощью наследования и усиления полезных свойств множества объектов определенного приложения в процессе имитации их эволюции.

В генетических алгоритмах свойства объектов значениями полей, введенными в запись, названной хромосомой.

Генетические алгоритмы оперируют хромосомами, относящимися к множествам объектов - популяции. Имитация генетических принципов - вероятностный выбор родителей среди членов популяции, скрещивание их хромосом, отбор потомков для включения в новое поколение объектов на основе оценки целевой функции, что ведет к эволюционному улучшению значению целевой функции F (функция полезности) от поколения к поколению.

Для поиска оптимального решения используются также методы, которые, в отличие от генетических алгоритмов, оперируют не с множеством хромосом, а с единственной хромосомой.

Так метод локального дискретного поиска основан на случайном изменении отдельных пар в значение генов в хромосоме, такие изменения называются мутацией.

После очередной мутации оценочное значение функции F, и результаты мутации сохраняются с некоторой вероятностью, зависящей от полученного значения F. Результат сохраняется в хромосоме, только если значение функции полезности улучшилось. В методе моделирования отжига результат мутации сохраняется с некоторой вероятностью, зависящей от полученных значений функции F.

ПОСТАНОВКА ЗАДАЧИ ПОИСКА ОПТИМАЛЬНЫХ РЕШЕНИЙ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Для применения генетических алгоритмов необходимо:

1. Выделить совокупность свойств объекта, характеризующихся внутренними параметрами и влияющими на его полезность, т. е. выделить множество управляющих параметров $X=(x_1, x_2, \ldots, x_n)$. Среди x типов могут быть величины различных типов. Наличие символьных величин обуславливает возможность решения задач не только параметрической, но и структурной оптимизации.
2. Сформулировать количественную оценку полезности вариантов объектов – функцию F.

   Если в исходном виде задача многокритериальная, то такая формулировка означает вывод скалярного критерия.
3. Разработать математическую модель объекта, представляющую собой алгоритм вычисления F для X.
4. Представить X в форме хромосомы – записи вида: $x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_n$

Используется следующая терминология:

Ген – управляемый параметр x;

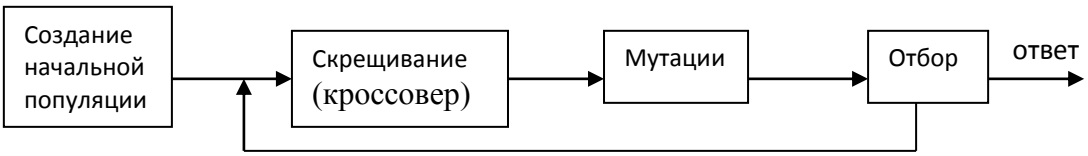
Аллель – значение гена;

Локус – позиция, занимая геном в хромосоме;

Генотип – экземпляр хромосомы, представляющий собой совокупность внутренних параметров проектируемого с помощью генетических алгоритмов объекта;

Генофонд – множество всех возможных генотипов;
Фенотип – совокупность генотипа и соответствующего значения F;
Под фенотипом понимают совокупность выходных параметров, синтезируемых с помощью генетических алгоритмов объектов.

ПРОСТОЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ



Вначале генерируются начальная популяция – несколько особей со случайным набором хромосом. Генетический алгоритм имитирует эволюцию этой популяции как циклический процесс скрещивания особей мутации и смены поколений (отбора). Вычислительный процесс начинается с генерации исходного поколения – множество, включенного N хромосом. N – размер популяции.

Генерация выполняется случайным выбором аллеями каждого гена. Далее организуется циклический процесс смены поколений.

for (k =0; k <G; k++)
for (j=0; j<N; j++)
{Выбор родительской пары хромосом}
Кроссовер
Мутации
Оценка F
Селекция
{Замена текущего поколения новым}
G – число повторений внешнего цикла.

Для каждого витка внешнего цикла генетического алгоритма выполняется внутренний цикл, на котором формируется экземпляры нового поколения.

На внутреннем цикле повторяется операция выбора родителей, кроссоверы, мутации, оценки приспособленности потомков, селекции хромосом.

Рассмотрим алгоритм выполнения операторов в простом г.а.

Выбор родителей

Этот оператор иллюстрирует естественный отбор, если отбор в родительскую пару хромосом с лучшими значениями F более вероятен.

Пр. Пусть F треб. min-т., тогда P_i выбора родителя с хромосомой C_i вычисляется:

$$P = (F_{\max} - F_i)\sum_{i=1}^{N}(F_{\max} - F_i) \qquad (1)$$

F_{max} – наихудшее значения F среди экземпляров текущего поколения.
F_i –значения функции i-ого экземпляра.
(1) называется Правило колеса рулетки.

Если в колесе рулетки выделены секторы пропорциональные (F_{max} – F_i), то вероятность попадания в них – суть F_i, определяется в соответствии с (1)

Пр. Пусть N = 4

F_i и P_i в таблице:

i	1	2	3	4
F_i	2	7	6	3
$F_{max} - F_i$	5	0	1	4
P_i	0,5	0	0,1	0,4

Скрещивание

Скрещивание заключается в передаче участков генов от родителей к потомкам. При простом (одноточечном кроссовере) хромосомы родителей разрываются в позиции одинаковой для обоих родителей. Место разрыва равновероятно. Далее рекомбинация, образовавшихся частей родительских хромосом, как в таблице 1, где разрыв подразделяется между 5 и 6 локусами.

Хромоссомы	Гены							
A	f	a	c	d	g	k	v	e
B	a	b	c	d	e	f	g	h
C	f	a	c	d	g	f	g	h
D	a	b	c	d	e	k	v	e

Мутации

Операция мутации выполняется с вероятностью P( ). Происходит замена аллели значением, выбираемым с равной вероятностью с области определенного гена. Благодаря мутации расширяется область ген. поиска.

Селекция

После каждого акта генерации пара потомков в новое поколение включается лучший экземпляр пары.

Внутренний цикл простого г.а заканчивается, когда число экземпляров нового поколения станет N.

Количество повторений G внешнего цикла чаще всего определяется автоматически, по выявлению признаков вырождения (стагнации), но с условием не превышения лимита машинного времени.

РАЗНОВИДНОСТИ ГЕН. ОПЕРАТОРОВ

Кроссовер:

В-первых, допустимы схемы многоточечного кроссовера.

Во-вторых, отметаем ситуации, когда на состав аллель наложены дополнительные условия.

Например. Пусть задача разбиение графа число вершин в подграфе A_1 и A_2 должно быть N_1 и N_2. Пусть к-тый аллель равен 1. Это означает, что К попадает в A_1, если же к-тый аллель равен 0, то в A_2.

Очевидно, что число 1 в хромосоме должно быть равно N_1, а число 0 N_2.

При рекомбинации левый участок хромосомы берется от одного из родителей без изменений, а в правом участке от другого родителя нужно согласовать число 1 с N_1.
Один из способов – метод PMX.

Для иллюстрации метода рассмотрим *пример* двухточечного кроссовера в задаче, когда в хромосоме должны присутствовать, причем только по одному разу, все значения генов из заданного набора.
Пусть в пр. набор включ-т от 1..9

1	2	3	4	5	6	7	8	9	Родите
3	7	1	9	2	4	8	6	5	ли
1	2	1	9	2	6	7	8	9	
1	2	3	9	5	6	7	8	4	

В таблице третья строка содержит хромосому первого из потомков, сгенерированного в результате применения первого из кроссоверов. Полученные хромосомы не относятся к числу допустимых, так как в них значения генов 1, 2, 9 встречаются второго рода, а значения 3, 2, 5 отсутствуют.

Четвертая строка показывает результат применения метода PMX. В этом методе выделяются сопряженные пары аллелей в локусе в одной из рекомбинированных частей. В нашем примере это пары 3–1, 4–9, 5–2. Хромосома потомка просматривается слева на право. Если встречаются повторяющиеся значения, то они заменяются сопряженным значением. В пр., повторно встречающиеся аллели заменены значениями 3, 5, 4.

Мутации

Бывают точечные мутации (в первом гене), макромутации (в нескольких генах) и хромосомные (появляются новые хромосомы)

Обычно вероятность появления мутаций указывается среди исходных данных, но можно авторегулировать число мутаций при их реализации только в такой ситуации, когда их родительские хромосомы различаются не более, чем в К генов.

Селекция

После определения и положительной оценки потомка он может быть сразу включен в текущую популяцию вместо худшего из родителей, при этом из алгоритма исключается внешний цикл.

Другой вариант селекции – отбор после каждого операции скрещивания двух лучших экземпляров среди двух потомков и двух родителей.

Часто именно потомки с F_{min} принужденно включаются в новое поколение, такой подход называется элипумом. Он способствует большей сходимости и локальному экстремуму, но в многоэкстремальной ситуации ограничивает возможность попадания в окрестности других локальных экстремумов.

Третий вариант селекции – отбор N экземпляров среди генов репродукции группы, которая состоит из родителей, потомков, мутантов, удовлетворяющих условию: $F_i < t$. t – порог значения функции полезности. Порог может быть равен ил среднему значению функции F в текущем поколении или значению F особи, занимающее определенное порядковое место. Суть мягкой схемы отбора – включение в новое поколение N лучших представителей репродукции группы.

Жесткая схема отбора – новое поколение, и экземпляры включаются с вероятностью, определяемой (1).

Четвертый вариант селекции – переупорядочивание – назначение переупорядочивание генов связывания со своими свойствами, носящими название эпистасис. Эпистасис имеет место, если F зависит не только от значений генов, но и от их позиционирования. Связывание эпистасиса говорит о нелинейности функции F, что приводит к усложнению оптимизационной задачи.

ОСОБЕННОСТИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Генетические алгоритмы – не единственный способ решения задач оптимизации.
1) Переборный метод наиболее прост в программировании, при этом необходимо вычислить значение цел. функции во всех точках, запоминая максимальные из них.
Минусы: Большая вычислительная сложность, но если перебор всех вариантов за разумное время возможен, то найденное решение является оптимальным.

2) Метод градиентного спуска.
Выбираются случайные значения параметров, а затем эти значения изменяются, добиваясь наибольшей скорости роста це…. функции.
Эти методы работают быстро, но не гарантируют оптимальности решения. Практические задачи ,как правило, мультимодальны и многомерны. Комбинируя переборные методы можно получить приблизительные решения, точность которых будет выше с увеличением t_n.
Генетические алгоритмы представляют собой именно такой комбинированный метод механизма скрещивания.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ТРАССИРОВКИ ДВУХСЛОЙНЫХ КАНАЛОВ

В настоящее время получили распространение канальные алгоритмы трассировки, которые требуют меньше машинных ресурсов и большинстве случаев обеспечивают стопроцентное разведение цепей.

Большое распространение получили алгоритмы моделирования эволюции. Это хорошо известная оптимизированная методология, основанная на аналогии процессов натуральной селекции в биологии.

Поиск в поле структурных модификаций адаптированных систем осуществляется генетическими алгоритмами.

Основная особенность генетических алгоритмов состоит в том, что анализируется не одно решение, а некоторое подмножество квазиопытных решений, названных хромосомами или стрингами.

Для каждой хромосомы должна быть цел. функция F(n), названная эволюционной.
n- число элементов в хромосоме. Функция F(n) вычисляет определенный вес каждой хромосомы. В каждой популяции хромосомы могут подвергаться действиям различных опер-в: кроссовера, инверсии, мутации, сегрегации, транслокации и т. п.

Генетические алгоритмы обладают возможностью выхода из локальных оптимумов, что позволяет получать лучшие решения, чем при решении задач канальной трассировки старыми алгоритмами.

ЗАДАЧА КАНАЛЬНОЙ ТРАССИРОВКИ В КЛАССИЧЕСКОЙ ПОСТАНОВКЕ

Канальные алгоритмы базируются на представлении о каналах и магистралях. Магистралью называется отрезок прямой, по которой проходит соединение в преимущественном напряжении.

Канал – область прямоугольной формы, на одной или нескольких сторонах которой расположены контакты с системой однонаправленных магистралей.

Каждая цепь – соединение эквипотенциальных контактов представлено как одиночный горизонтальный сегмент с несколькими вертикальными сегментами, которые соединяют горизонтальный сегмент с контактами цепи.

Горизонтальные сегменты располагаются в одном слое, вертикальные в другом. Соединения между горизонтальными и вертикальными делятся через переходные отверстия.

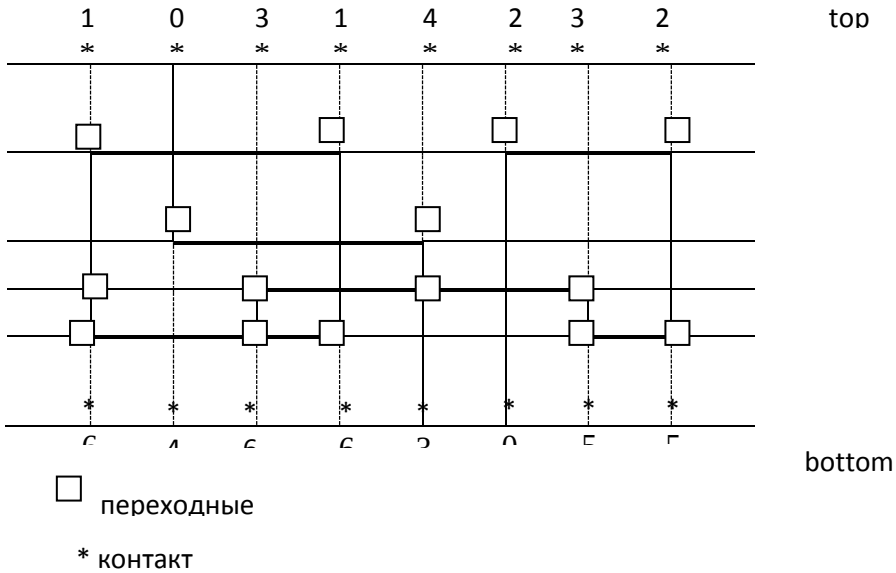
Основная задача канальной трассировки – выбор наименьшей трассировки канала, достаточной для размещения в нем всех соединений и назнач. соединений на магистралях.

Необходимо минимизировать суммарную длину соединений, число переходных отверстий и т. д.

Задача канальной трассировки в классической постановке основана трассировке двустороннего канала по верхней и нижней сторонам которого проходят линейки контактов.

Изломы (т. е. переходы) горизонтального участка с одной магистрали на другой не допускаются.

Описание каналов



Канал описывается двумя последовательностями top и bottom, в которых размещаются верхние и нижние линейки контактных площадок каналов. Размер обеих последовательностей равен C – число колонок в канале

Множество цепей определяется как Net = $\{N_1, N_2, \ldots, N_n\}$ (n – это число цепей)

Пример:

top = {1, 0, 3, 1, 4, 2, 3, 2}; bottom = {6, 4, 6, 6, 3, 0, 5, 5}; n = 6; C = 8

Горизонтальные и вертикальные ограничения

При канальной трассировке не допускается наложения вертикальных и горизонтальных сегментов цепей. Для решения этой задачи вводятся графы вертикальных и горизонтальных ограничений.

Вертикальные ограничения описываются ориентированным графом вертикальных ограничений (г.в.о.)

GV = {Enet, Ev}

Enet – множество вершин, соответствующих множеству цепей net

Ev – множество направляющих ребер.

Ребро (n, m) существует тогда, когда цепь n расположена выше цепи m для предотвращения наложения вертикальных сегментов цепей.

Для наших последовательностей г.в.о. на рис.:

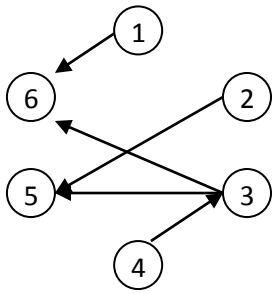


рис.

Например, на рисунке выше в г.в.о. существует путь из 1 в 6. Значит, что цепь 1 должна быть расположена выше цепи 6, чтобы не было наложений вертикальных сегментов на 1 и 6 контактов. Чтобы задача решалась в рамках классической постановки задачи, граф GV должен быть ацикличным, иначе задача может быть решена только с введением изломов, а это противоречит условиям классической постан. задачи.

Далее будем использовать расширенный г.в.о. (р г.в.о.)

GRV = {Enet, Erv}

Enet – множество вершин, соответствующих множеству цепей net

Erv – множество направляющих ребер.

Ребро (n, m), принадлежащее Erv, существует тогда, когда в г.в.о. существует путь из вершины n в m.

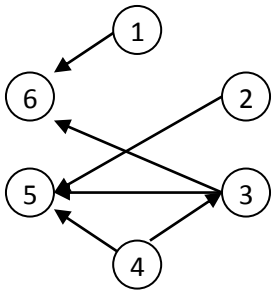


рис.

На рисунке 2а в г.в.о. есть путь из 4 в 5 через 3. Следовательно, в р.г.в.о. существует ребро (4, 5) и (4, 6)

Горизонтальные ограничения представлены не ориентированным графом горизонтальных ограниченицц (г.г.о)

GH = {Enet, Ен}

Ен – множество ребер.

Ребро (n, m), принадлежащее Ен, существует тогда, когда магистрали для n и m разнесены для исключения наложения горизонтальных сегментов n и m.

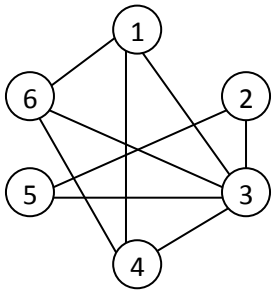


рис.

На рисунке 2в г.г.о. есть путь из 1 в 6, значит, что цепь 1 должны быть расположена выше 6, чтобы не было горизонтальных наложений. Цепи 1 и 5 могут быть расположены на одной магистрали. Цепь 1 и 2 тоже.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ КАНАЛЬНОЙ ТРАССИРОВКИ

Генетические алгоритмы – итерационная процедура, которая обрабатывает группу хромосом решений, названную популяцией.

Число хромосом в популяции – ее размер N. Каждая хромосома состоит из генов, каждый ген хромосомы может быть в нескольких состояниях – аллели.

Приведем соответствие терминов генетических алгоритмов и классические задачи.

Денотип – топология расположения цепей на кристалле.

Генотип – генетическая схема кодирования топологии.

Хромосома – кодированное представление первого варианта топологии.

Ген – элемент хромосомы, задающий некоторый фрагмент топологии.

Популяция – набор хромосом.

Фитнеса – целевая функция, определяющая качество решения задачи.

Генерация – первый цикл работы генетических алгоитмов.

Генетические алгоритмы требуют, чтобы хромосомы оценивались с точки зрения F задач. Функция F оценивает к-л. качества решения, соответствующего данной хромосоме (длину цепей, сило магистралей).

Генетические алгоритмы обрабатывают популяцию решений, закодированную в хромосому. В процессе обработки популяции к ней последовательно применяются генетические операторы, так как кроссовер и мутация, с заданными вершинами P(c) и P(m) и др.

Затем проводится редукция (анализ) увеличивающейся популяции для отбора лучших решений, которые составят следующее поколение, после чего цикл (популяция) повторяется.

Число таких циклов называется числом генерации T. В общем случае может быть построено большое число архитектур реализации генетического поиска.

СТАНДАРТНАЯ СХЕМА ГЕНЕТИЧЕСКОГО ПОИСКА. СТРУКТУРА Г.А.

1) Определение размера популяции M, числа генерации T, вероятности кроссовера P(C), вероятности мутации P(M).
2) Задание случайным образом начальной популяции П(0) размером M.
3) T положить равным 0.
4) Выбор случайным образом M пар хромосом из популяции П(t) и применение кроссовера к каждой паре с заданной вероятностью P(C)
5) Применение операции мутации к каждой хромосоме популяции П(t) с заданной вероятностью P(m).
6) Отбор M хромосом с наилучшим значением F из получившейся популяции П(t) в новую популяцию П (t + 1)
7) t = t + 1
8) Если t < T, то переход к пункту 4
9) Вывод хромосомы с наилучшими значениями.

ГЕНЕТИЧЕСКОЕ ОПЕР-И ПРИМ-Е В АЛГОРИТМЕ КАНАЛЬНОЙ ТРАССИРОВКИ.

КОДИРОВАНИЕ ХРОМОСОМЫ

Пусть хромосома описывает взаимное расположение цепей канала. Для этого каждой паре цепей (n, m) ставится в соответствие ген, который может принимать 3 состояния: 0 – если m должна располагаться выше n, 1 – если m должна располагаться ниже n и * – если их взаимное расположение не имеет значения или определяется из взаимного расположения остальных цепей (это происходит если цепи не имеют горизонтальных ограничений)

Для нашего примера хромосома может иметь вид, приведенный в таблице:

цепь n	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5
цепь m	2	3	4	5	6	3	4	5	6	4	5	6	5	6	6
ген	*	0	0	*	0	0	*	0	*	1	0	0	*	0	*

В строке «ген» записывается значение гена в хромосоме, задающий расположение цепей, показанное на рисунке 1.

Длина хромосомы L = N*(N–1)/2 = 6*5/2 = 15. длина хромосомы достаточно большая, что неприемлемо. Поэтому для понижения длины хромосомы используется анализ р.г.в.о и г.г.о (рис. 2)

Понижение длины хромосомы получаем за счет того, что взаимное расположение пар цепей уже зафиксировано р.г.в.о., и изменение этого расположения приводит к наложению вертикальных сегментов цепей в канале. Это ведет к образованию «мертвых» хромосом, то есть такой, из которой нельзя получить решение, удовлетворяющее условиям задачи трассировки.

Второй прием, позволяющий уменьшить длину хромосомы, основывается на том, если цепи не имеют горизонтальных ограничений, то их взаимное положение либо не важно, либо определяется из соотношения с остальными цепями. Такие состояния отмечены «*». Таким образом, длина хромосомы может быть определена: L = NGO – NRVO, где NGO – число горизонтальных ограничений в г.г.о.(9), а NRVO – число вертикальных ограничений в р.г.в.о.(6).

Цепи 4 и 5 не зависят друг от друга, как по горизонтали, так и по вертикали. Для пары цепей (1, 6) (2,5) (3,4) (3,5) (3,6) (4,6) взаимное расположение жестко задано р.г.в.о., а а взаиморасположение цепей в (1, 2) (1, 5) (2, 4) (2, 6) (4, 5) (5, 6) (см. г.г.о) не имеет значения, так как цепи в этих парах не конфликтуют по горизонтали.

В нашем примере хромосома будет выглядеть так:

цепь n	1	1	2
цепь m	3	4	3
ген	0	0	0

1 – должна быть выше 4, а 2 должна быть выше 3.

Получение из хромосомы эскиза канала с разведенными цепями

Для получения из хромосомы вида A=(a_1, a_2, a_3,), где a_i равна 0 или 1 эскиза канала с разведенными цепями используется граф топологии (ГТ). ГТ – ориентированный граф вида GT=(ENet, ET), где ENet – множество цепей, ET – множество направленных ребер, описывающих взаимное расположение цепей в канале.

Ребро $(m, n) \in ET$ существует тогда и только тогда, когда цепь m расположена в канале выше цепи n, т.е. на магистрали с меньшим номером. ГТ строится на основе РГВО, направленные ребра которого соответствуют параметрам цепей, взаимное расположение которых жестко задано с добавлением направленных ребер, соответствующих параметрам ветвей, образующих хромосому А. для нашего примера рис.1 ГТ будет иметь вид, приведенный на рис.3.

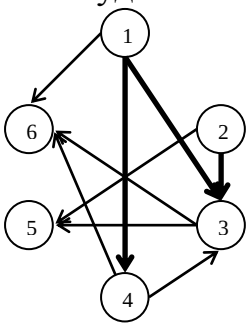


Рис.3

Цепь m	1	1	2
Цепь n	3	4	3
Ген	0	0	0

A=(0,0,0)

Граф топологии для хромосомы A=(0,0,0).

Канал восстанавливается из хромосомы следующим образом:

Шаг 1. Строим по хромосоме граф топологии.

Шаг 2. Полагаем i=0.

Шаг 3. Находим вершины (1 и 2 для нашего случая), у которых есть только исходящие дуги. Размещаем их на магистрали с номером i или на магистрали с

меньшим номером, если это возможно и удаляем эти вершины с инцидентными им ребрами из ГТ.

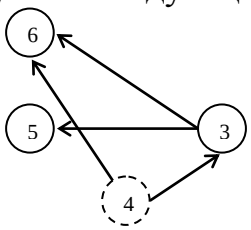
Шаг 4. i=i+1

Шаг 5. Если ГТ не пуст, то возвращаемся к шагу 3.

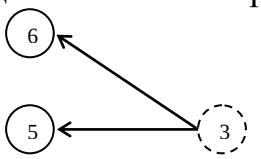
Шаг 6. Не нарушая взаимного расположения, смещаем цепи по магистрали так, чтобы минимизировать длину вертикальных сегментных цепей.

Шаг 7. Возвращаем полученный образец размещения цепей в канале шириной i (см. алгоритм расслоение МПП).

Для хромосомы A=(0,0,0) имеем: на первом шаге находим вершины 1 и 2, у которых есть только исходные дуги. Цепи 1 и 2 размещаем на магистрали m1, получаем следующий граф топологии:



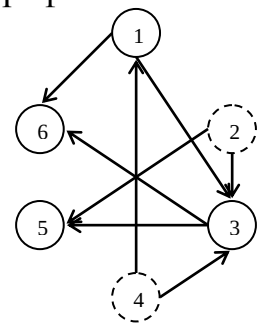
На втором шаге исключаем вершину 4, а цепь 4 размещаем на m2 получаем ГТ представленный на рис.5.



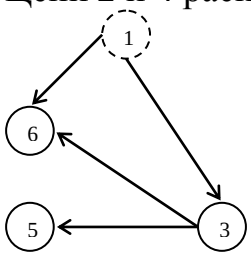
На третьем шаге исключаем вершину 3, а цепь 3 размещаем на магистрали 3.

На 4-ом шаге назначаем цепям 5 и 6 магистраль m4 (5 и 6 не конфликтуют ни по вертикали, ни по горизонтали).

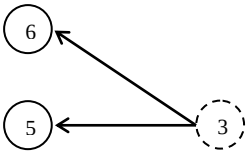
На рис.1 показан полученный образец трассировки, который является оптимальным решением для нашего примера. Если хромосома A=(0,1,0) (отличие от представленного решения заключается в том, что цепь 4 располагается выше цепи 1), то граф топологии будет иметь вид:



Цепи 2 и 4 располагаем на магистрали m1(удаляем 1 и 4).

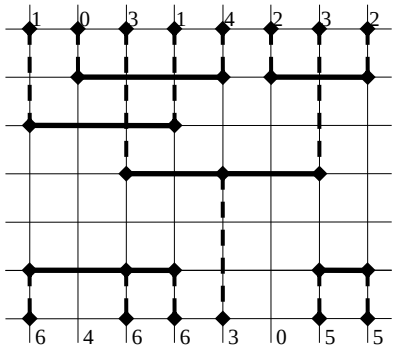


Цепь 1 располагаем на магистрали m2.



Цепь 3 располагаем на магистрали М3, 5 и 6 – на магистрали М4.

Эскиз канала с разведенными цепями для хромосомы А=(0,1,0) показан на рисунке:



Подсчитаем суммарную длину вертикального сегмента: $L_{\sum} = 3+4+4+3+3+1+4+2 = 24$

Целевая функция оценки хромосомы

Когда генетические алгоритмы применяются для решения оптимизационных задач, важнейшим требованием является разработка целевой функции (ЦФ) для полученных решений.

ГА обычно используют для минимизации (максимизации), но желательно, чтобы целевая функция была положительной. В нашем случае задача заключается в минимизации целевой функции. Принимая во внимание характеристики задач канальной трассировки, целевая функция может быть определена следующим образом:

F(A)=(Used Track(A)+2)*C + Total Vert Seg(A)

C – число контактов (8 линеек Top или Bottom)

2 – число слоев печатной платы

Программная функция Used Track(A) определяет число магистралей (в нашем примере 4) занимаемые каналом, полученным при восстановлении хромосомы А, а программная функция Total Vert Seg(A) определяет длину вертикальных сегментов цепи в полученном решении.

Длина вертикального сегмента цепи определяется как расстояние между контактами и переходными отверстиями, которые соединяют вертикальные и горизонтальные сегменты. Например для канала приведенного на рис.1 занятых магистралей равно 4, длина вертикального сегмента равна 22, т.о.целевая функция хромосомы F(A)=(4+2)*8+22=70, а для канала с хромосомой А=(0,1,0) (рис.7) F(A)=72.

Данная методика определения F(A) направлена в первую очередь на минимизацию ширины канала, а во вторую на минимизацию суммарной длины соединений.

Кроссовер и мутация

Согласно структуре ГА, все хромосомы (случайно полученные решения) оцениваются целевой функцией F(A). Далее, согласно процедуре селекции (отбора) производится построение списка упорядоченных хромосом. На основе анализа этого

списка производится подбор пар хромосом на основе элитной селекции(хромосома с наилучшим значением целевой функции и случайно выбранной хромосомой).

После подбора пар применяется оператор кроссовера – ОК с вероятностью P_c к каждой паре хромосом. Исходная хромосома называется родительской, а хромосомы, полученные после операторов ГА – потомки.

Алгоритм применения ОК реализуется следующим образом:

Шаг 1. i=0

Шаг 2. Выбрать хромосом с наилучшим значением целевой функции как первого родителя.

Шаг 3. Выбрать произвольно второго родителя.

Шаг 4. Сгенерировать случайное число $RND \in [0,1]$.

Шаг 5. Если RND<P_c, то применить оператор кроссовера к выбранным хромосомам и добавить получившихся потомков к популяции.

Шаг 6. i=i+1.

Шаг 7. Если i<Mm, переход к шагу 2 (Mm – различные популяции).

Под оператором кроссовера подразумевается обмен, скрещивание генетического материала между двумя различными хромосомами (потенциальными родителями). После окончания оператора кроссовера полученные потомки подвергаются мутации. В большинстве случаев может быть выбран стандартный двухточечный оператор кроссовера. Первая и вторая точки разрыва в таком операторе кроссовера выбираются случайно.

Часть первого родителя перед первой и после второй точки разреза копируется в аналогичные места в потомке 1. Место между первой и второй точкой разреза во втором родителе копируется в аналогичное место первого потомка. Второй потомок строится аналогично.

Рассмотрим пример реализации оператора кроссовера. Пусть вертикальная волнистая линия означает точку разреза оператора кроссовера.

Родитель 1	0	1	0	F(A)=72 (рис.7)
Родитель 2	1	0	1	F(A)=7
Потомок 1	0	0	0	F(A)=70 (рис.1)
Потомок 2	1	1	1	F(A)=77

Потомок 2 аналогичен родителю 2. В данном случае получен потомок 1 с лучшим значением целевой функции, чем значение целевой функции (72,77).

Мутация произвольно изменяет 1 ген выбранной хромосомы случайно изменяя значение гена с вероятностью P_M равной норме мутации. Алгоритм применения операции мутации выглядит следующим образом:

Шаг 1. i=i+1.

Шаг 2. Сгенерировать случайное число RND [0,1]

Шаг 3. Если RND< P_M, то применить ОМ к i-ой хромосоме М.

Шаг 4. i=i+1.

Шаг 5 если i<M, то к шагу 1.

Смысл ОМ введение добавочных изменений в популяцию. В простейшем случае может быть выбрана точечная мутация, она заключается в изменении произвольного

гена в хромосоме с вероятностью $P_{M.}$ Это происходит следующим образом: один из генов хромосомы выбирается случайно, а затем меняет свое значение на противоположное. Пример хромосомы (до применения ОМ): 0,(1),0. Значение целевой функции – 72.

Хромосома после применения ОМ: 0,(0),0, значение целевой функции – 70 – min. Выбранная точка мутации показана в скобках. После применения ОМ получается хромосома с лучшим значением целевой функции. Дальнейшее улучшение работы генетического алгоритма можно получить за счет сортировки гена в хромосоме или путем использования более сложной целевой функции. Кроме того, можно анализировать параллельные и последовательно – параллельные методы генетического поиска.

Теоретическая временная сложность рассмотренного генетического алгоритма составляет $t*(M*P_c*2)*2*P_M*O(N_2)$

$O(N_2)$ – временная сложность декодирования хромосомы;

M – число цепей;

t – число поколений.

Особенности генетических алгоритмов

Генетические алгоритмы – не единственный способ решения задач оптимизации. Сравним стандартные подходы с генетическими алгоритмами.

1. Переборный метод – наиболее прост в программировании. Для поиска оптимального решения требуется последовательно вычислить значения целевой функции во всех возможных х.т., запоминая максимальный из них. Недостаток этого метода – большая вычислительная сложность.
2. Метод градиентного спуска. В начале выбирается некоторое случайное значение переходов. А затем эти значения постепенно изменяют, добиваясь наибольшей скорости роста целевой функции. При достижении локального максимума такой метод останавливается. Для поиска глобального оптимума требуются дополнительные методы. Градиент работает быстрее, но не гарантирует оптимальности решения.

Градиентные методы могут получить приближенное решение, так как сложность будет возрастать с увеличением времени расчета. ГА представляет собой такой метод.