Математические методы проектирования топологии СБИС

А.М.Марченко

Содержание

- Сведения о КМОП технологии и методология проектирования СБИС
- Задача разбиения электрической схемы
- Задача размещения модулей СБИС
- Задача трассировки соединений

Литература

1. T. Lengauer. Combinatorial algorithms for integrated circuit layout. Wiley, 1990, 694 p.
2. Naveed Sherwani. Algorithms for VLSI physical design automation. Kluwer academic publishers, 1995, 538p.
3. Г.Г.Казеннов, В.М.Щемелинин. Топологическое проектирование нерегулярных БИС. М., Высшая школа, 1990, 109с.
4. Introduction to Algorithms, T. Cormen, C. Lesierson, R. Rivest, The MIT Press, Second Printing, 1996.
5. Handbook of Algorithms for Physical design Automation, Edited by Charles J. Alpert Dinesh P. Mehta Sachin S. Sapatnekar, 2009.
6. Andrew B. Kahng, Jens Lienig, Igor L. Markov, Jin Нию VLSI Physical Design: From Graph Partitioning to Timing Closure, 2011

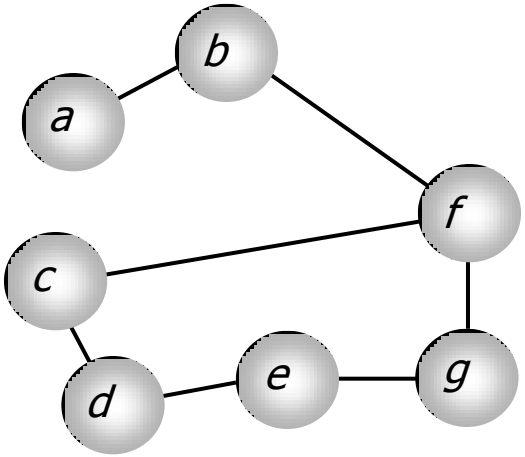
Сведения из теории графов

Определение 1: *Конечный граф* $G=\langle V,E\rangle$ состоит из конечного множества вершин $V=\{v_1,\ldots,v_n\}$ и конечного множества ребер $E=\{e_1,\ldots,e_m\}$, причем $E\subseteq V\times V$ для ориентированного графа и $E\subseteq\{(x,y):x,y\in V\wedge x\neq y\}$ для неориентированного.

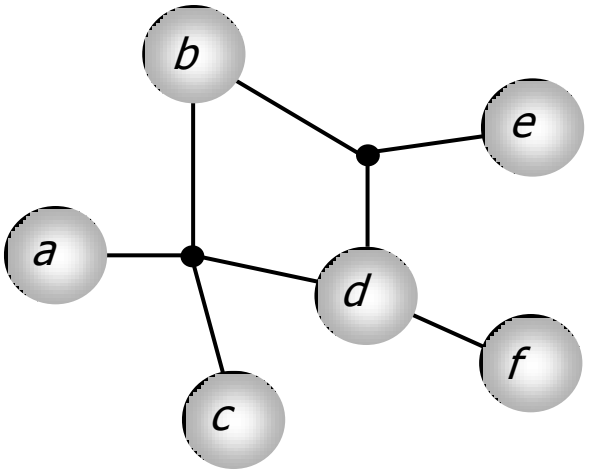
Способы представления графов:

- Матрица инциденций;
- Списки ребер (пар вершин, соединенных ребром);
- Матрица смежностей (весов);
- Списки инцидентности;
- Кодовые схемы

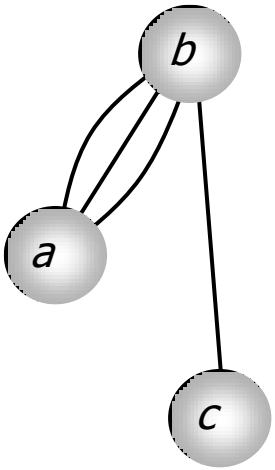
Графы



Гиперграфы

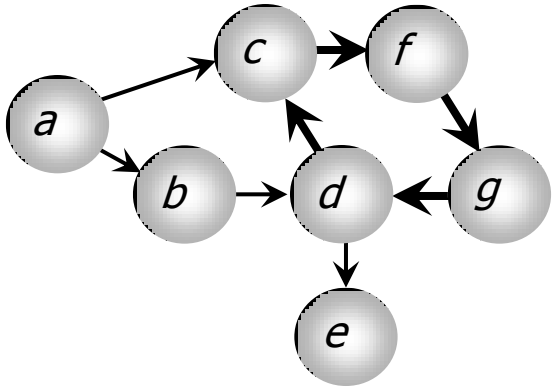


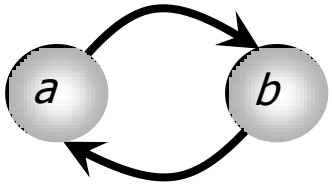
Мультиграфы

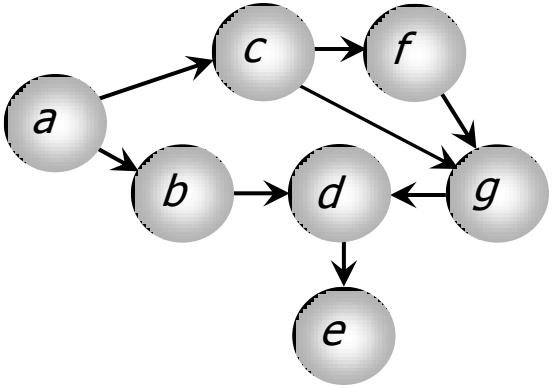


Ориентированные графы с циклами

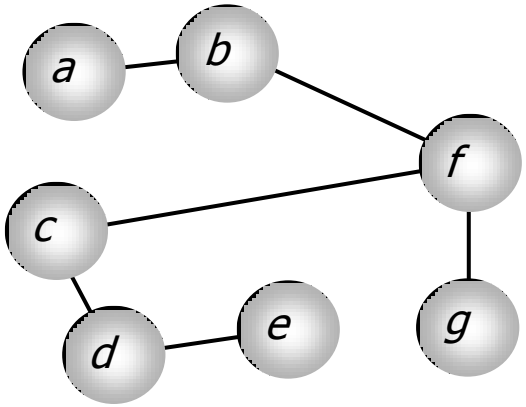
Ориентированные графы без циклов



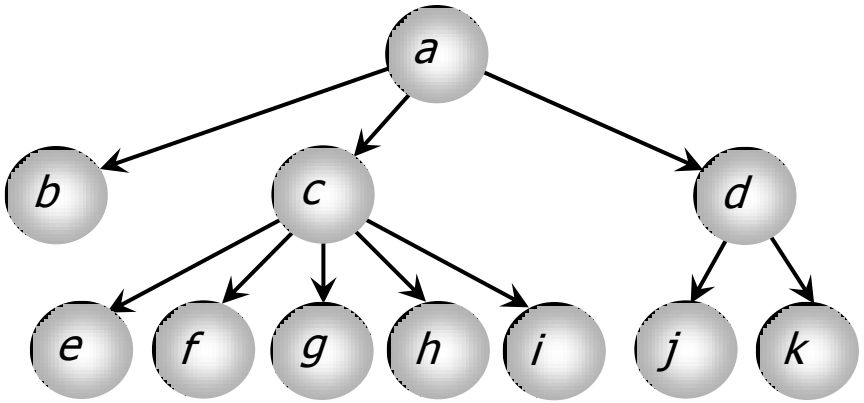




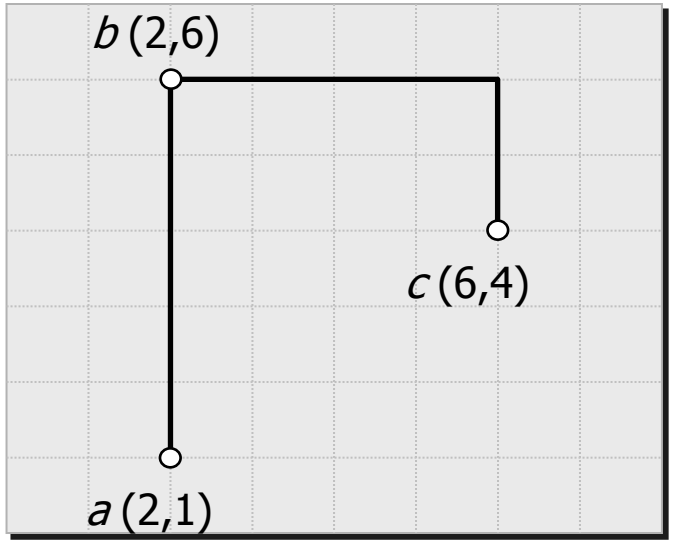
Неориентированные графы



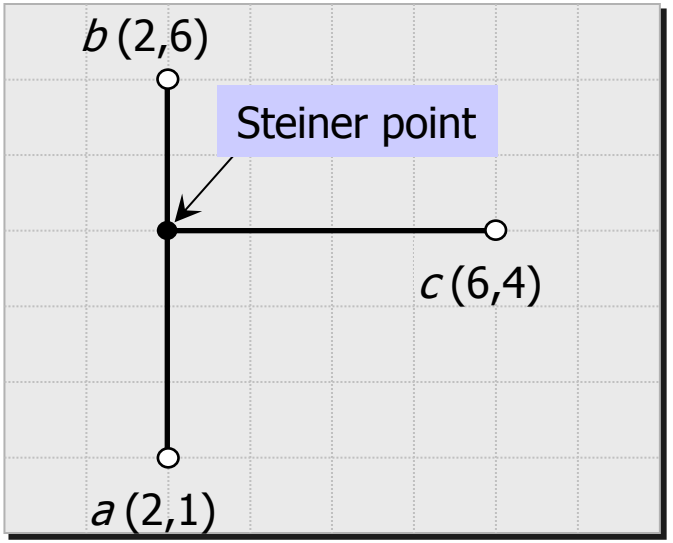
Ориентированные деревья



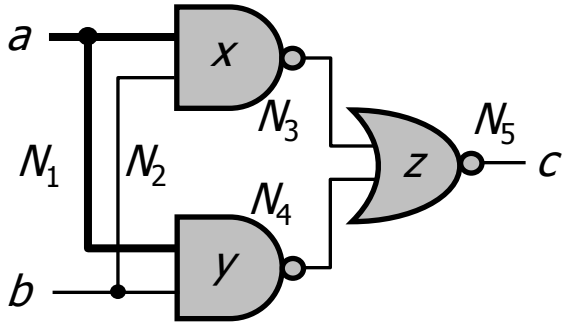
Rectilinear minimum spanning tree (RMST)



Rectilinear Steiner minimum tree (RSMT)



Netlist



(***a*: *N*$_1$**)
(*b*: *N*$_2$)
(*c*: *N*$_5$)
(***x*: *IN*1 *N*$_1$**, *IN*2 *N*$_2$, *OUT N*$_3$)
(***y*: *IN*1 *N*$_1$**, *IN*2 *N*$_2$, *OUT N*$_4$)
(*z*: *IN*1 *N*$_3$, *IN*2 *N*$_4$, *OUT N*$_5$)

Pin-Oriented Netlist

(***N*$_1$: *a*, *x.IN*1, *y.IN*1**)
(*N*$_2$: *b*, *x.IN*2, *y.IN*2)
(*N*$_3$: *x.OUT*, *z.IN*1)
(*N*$_4$: *y.OUT*, *z.IN*2)
(*N*$_5$: *z.OUT*, *c*)

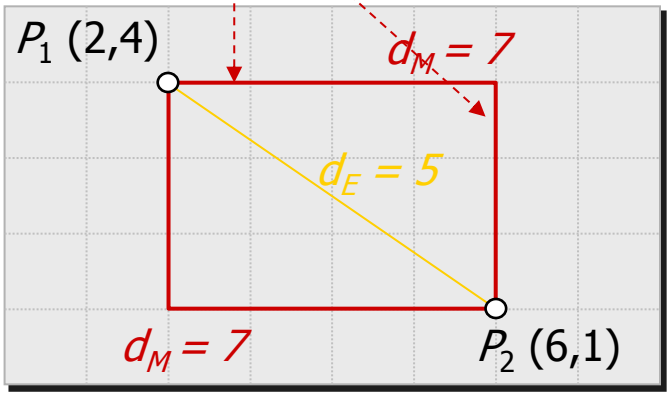
Net-Oriented Netlist

Расстояние между двумя точками $P_1(x_1,y_1)$ и $P_2(x_2,y_2)$

$$d = \sqrt[n]{|x_2 - x_1|^n + |y_2 - y_1|^n}$$

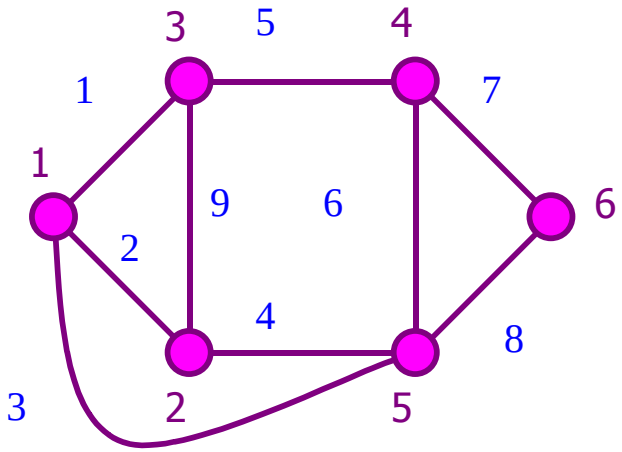
with $n = 2$: Euclidean distance $d_E(P_1,P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

$n = 1$: Manhattan distance $d_M(P_1,P_2) = |x_2 - x_1| + |y_2 - y_1|$



Способы представления графов

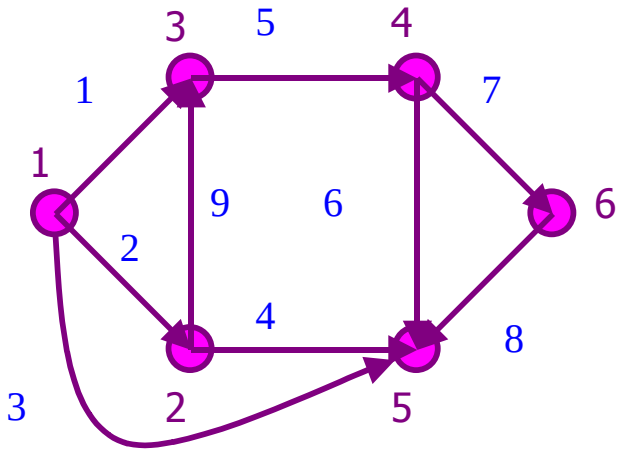
- Матрица инциденций



	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	1	1	1	0	0	0	0	0	0
2	0	1	0	1	0	0	0	0	1
3	1	0	0	0	1	0	0	0	1
4	0	0	0	0	1	1	1	0	0
5	0	0	1	1	0	1	0	1	0
6	0	0	0	0	0	0	1	1	0

Способы представления графов

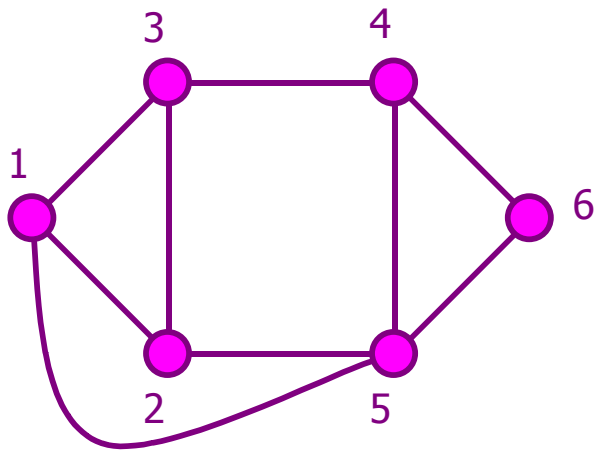
- Списки ребер



{1,1,3}, {2,1,2}, {3,1,5},
{4,2,5}, {5,3,4}, {6,4,5},
{7,4,6}, {8,6,5}, {9,2,3}

Способы представления графов

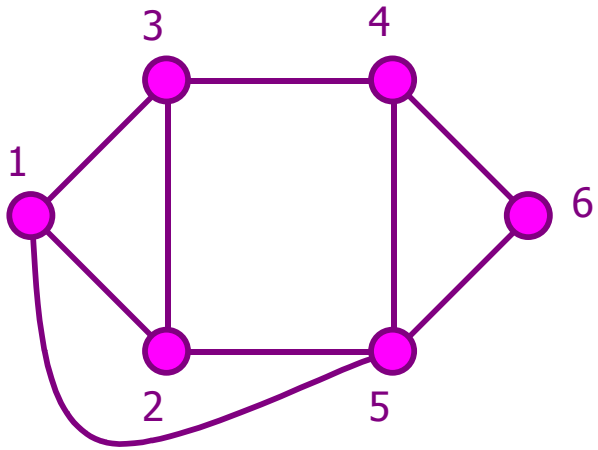
- Матрица смежностей (весов)

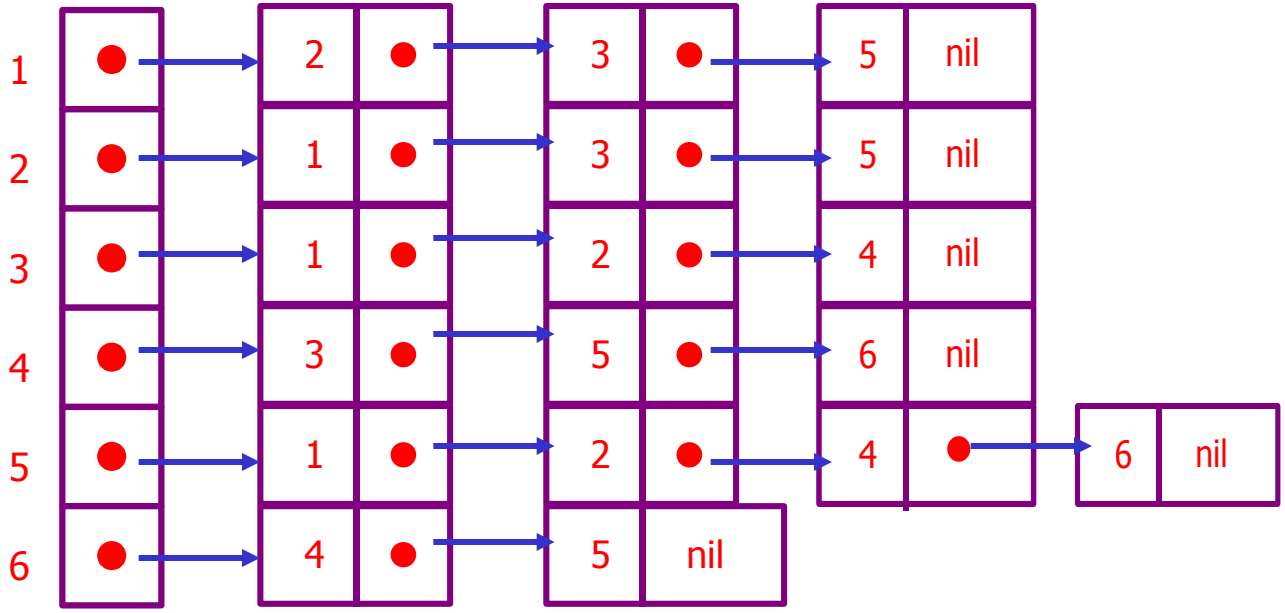


	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	0	1	0	1	0	1
6	0	0	0	1	1	0

Способы представления графов

- Списки инцидентности ЗАПИСЬ[v]





Способы представления графов

- Кодовые схемы – код Харари

	1	*2*	*3*	*4*	*5*	*6*
1	0	1	1	0	0	0
2	1	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	0	1	0	1	0	1
6	0	0	0	1	1	0

110001010100111

Оцените сложность алгоритма перевода одного описания графа в другое.

Поиск в глубину (Depth First Search)

Поиск в глубину из вершины v

- *procedure* WG(v)
- *begin* рассмотреть v
  - НовыйСмежный[v] ← *false*
  - *for* u ∈ ЗАПИСЬ[v] *do*
    - *If* НовыйСмежный[u] *then* WG(u)
- *end*

Поиск в глубину в графе

- *begin*
  - *for* v ∈ V *do*
    - НовыйСмежный[v] ← *true*
  - *for* v ∈ V *do*
    - *if* НовыйСмежный[v] *then*
      - WG(v)
- *end*

Связные компоненты графа можно вычислить за время O(n+m)

Поиск в ширину (Bread First Search)

Поиск в ширину из вершины v

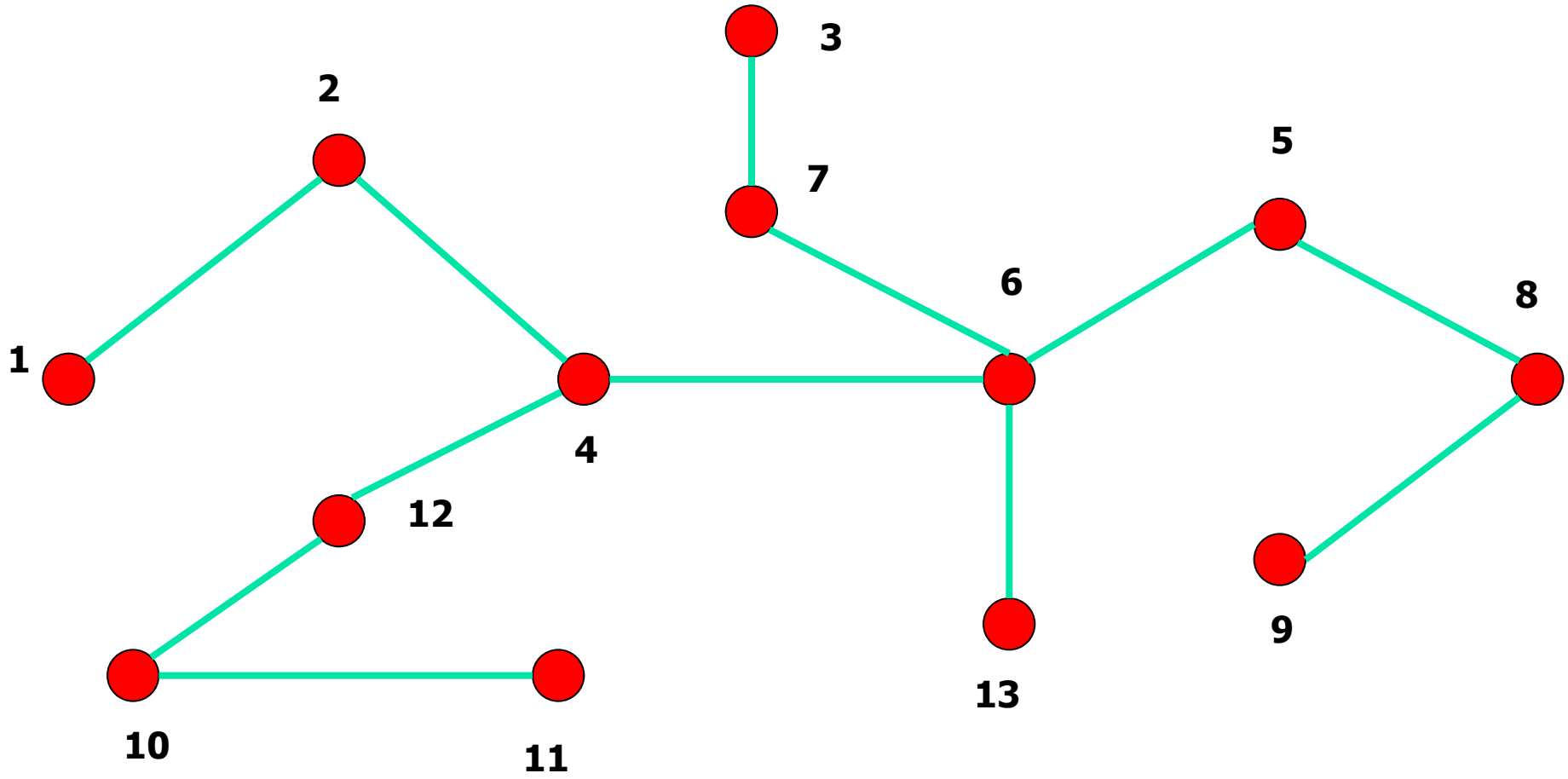
```
procedure WS(v)
begin рассмотреть v
    Очередь ← v
    НовыйСмежный[v] ← false
    while Очередь ≠ ∅ do
        begin p ← Очередь
            for u ∈ ЗАПИСЬ[p] do
                if НовыйСмежный[u] then
                    begin Очередь ← u
                        НовыйСмежный[u] ← false
                    end
        end
end
```

Стягивающие деревья

- Определение 2: *Деревом* называется произвольный неориентированный граф без циклов.

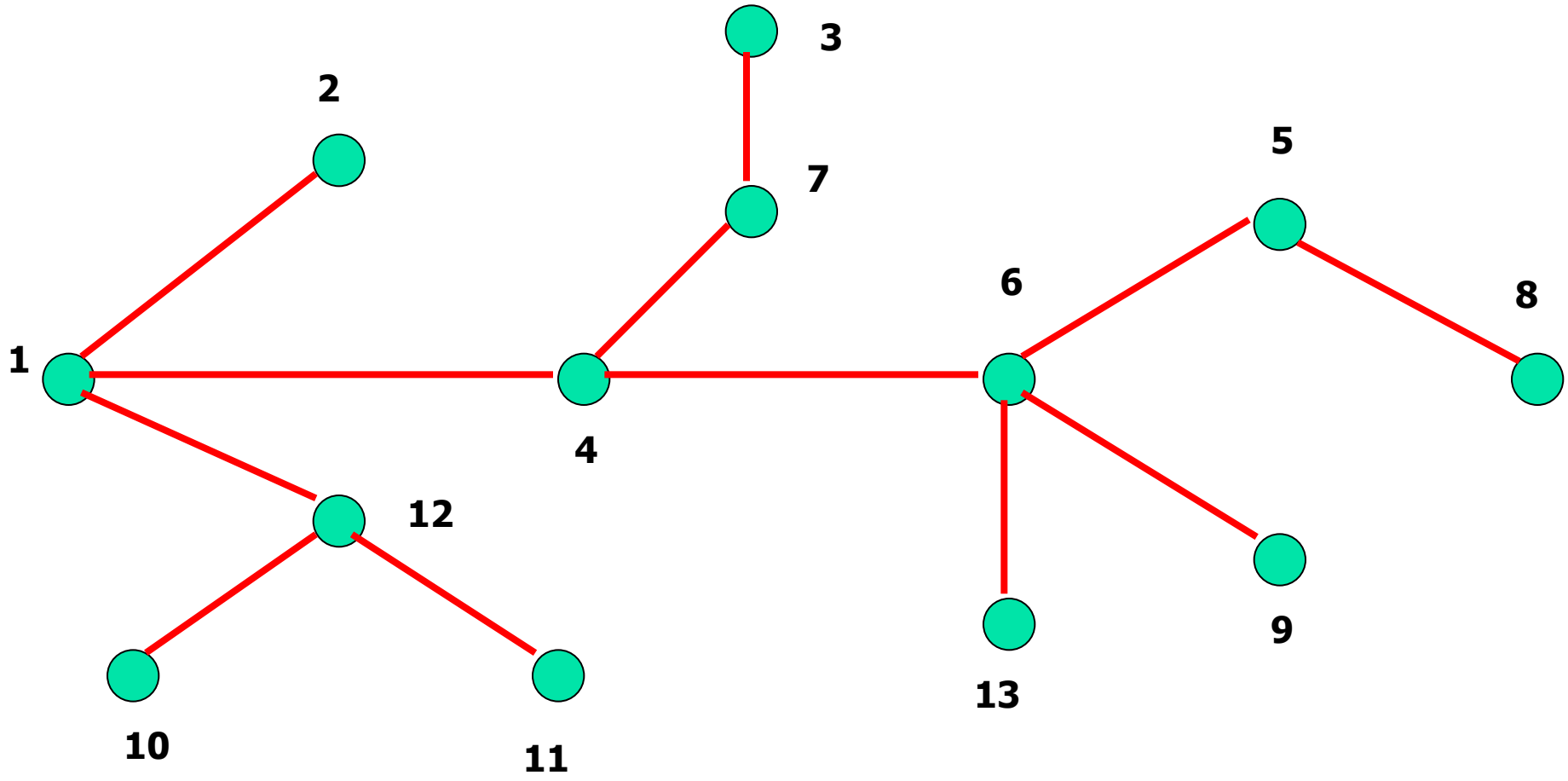
```
procedure WGD(v)
begin НовыйСмежный[v] ← false
    for u ∈ ЗАПИСЬ[v] do
        if НовыйСмежный[u] then
            begin
                T ← T ∪ {v,u}
                WGD(u)
            end
end
```

```
begin  //главная программа
    for u ∈ V do
        НовыйСмежный[v] ← true
        T ← ∅ // множество
              // найденных ребер
        WGD(r) // произвольная
               // вершина графа
end
```



Математические методы проектирования
топологии СБИС

Пример



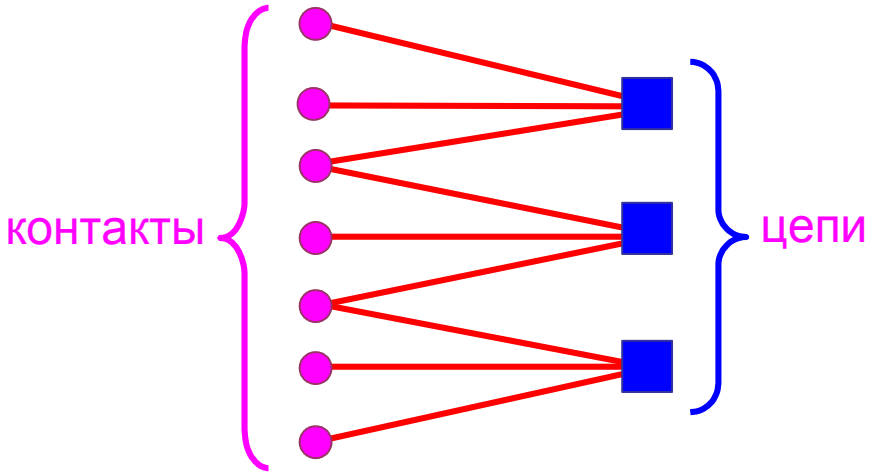
Гиперграфы

- Определение 10: Конечный граф $G=\langle V,E\rangle$ называется *гиперграфом*, если элементами множества E являются не обязательно двухэлементные, а любые подмножества множества V.

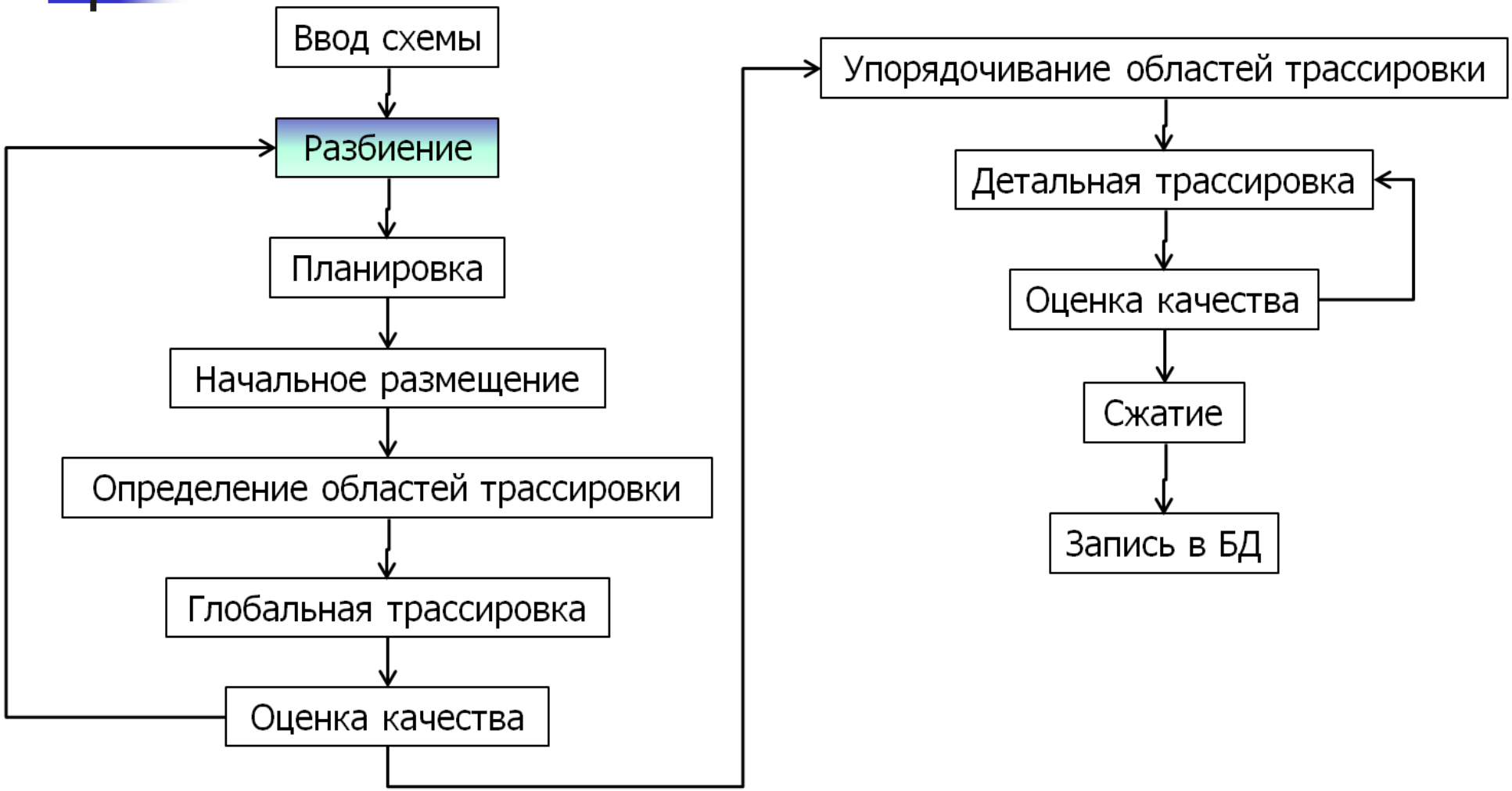
- Определение 11: *Двудольный граф* – это граф $G=\langle V,E\rangle$, такой что $V=V_1\cup V_2$ и $V_1\cap V_2=\varnothing$, причем всякое ребро из E соединяет вершину из V_1 с вершиной из V_2.

Представление гиперграфов

- Утверждение: Гиперграф $G=\langle V,E\rangle$ может быть интерпретирован как двудольный граф $G'=\langle V\cup E,U\rangle$, где $\forall u\in U$: $u=(v,e)$, v – вершина гиперграфа, а e – его гиперребро ■.



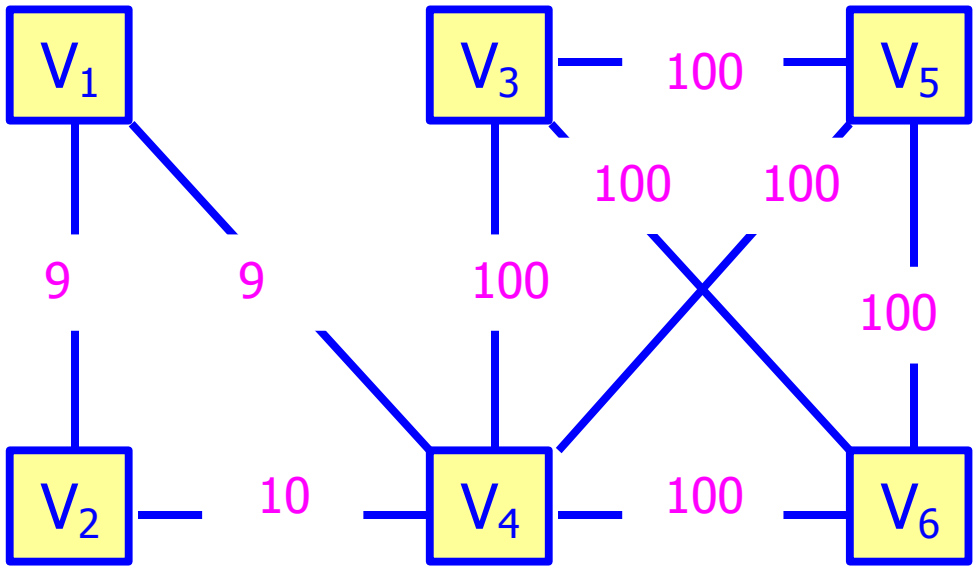
Маршрут проектирования топологии СБИС



Постановка задачи

- Определение 1: Пусть дано множество модулей $V=\{v_1,\ldots,v_n\}$. *K-partitioning* $P^k=\{C_1,\ldots,C_k\}$ состоит из k взаимно непересекающихся кластеров таких, что $C_1 \cup C_2 \cup \ldots C_k = V$.
- Определение 2: *Min-cut bipartitioning* – минимизировать $F(P^2)=|E(C_1)|=|E(C_2)|$ при $C_1 \neq 0$, $C_2 \neq 0$.
- Определение 3: *Min-cut bisection* - минимизировать $F(P^2)=|E(C_1)|$ при $|w(C_1)-w(C_2)| \leq \varepsilon$.
- Определение 4: *Size-constrained min-cut bipartitioning* – минимизировать $F(P^2)=|E(C_1)|$ при $L \leq w(C_i) \leq U$, для i=1,2.
- Определение 5: *Minimum ratio cut bipartitioning* – минимизировать $F(P^2)=|E(C_1)|/(w(C_1) \times w(C_2))$

Примеры



Min-cut bipartitioning:
$\{(v_1),(v_2,v_3,v_4,v_5,v_6)\}$;

cut size=18

Min-cut bisection:
$\{(v_1,v_2,v_4),(v_3,v_5,v_6)\}$;

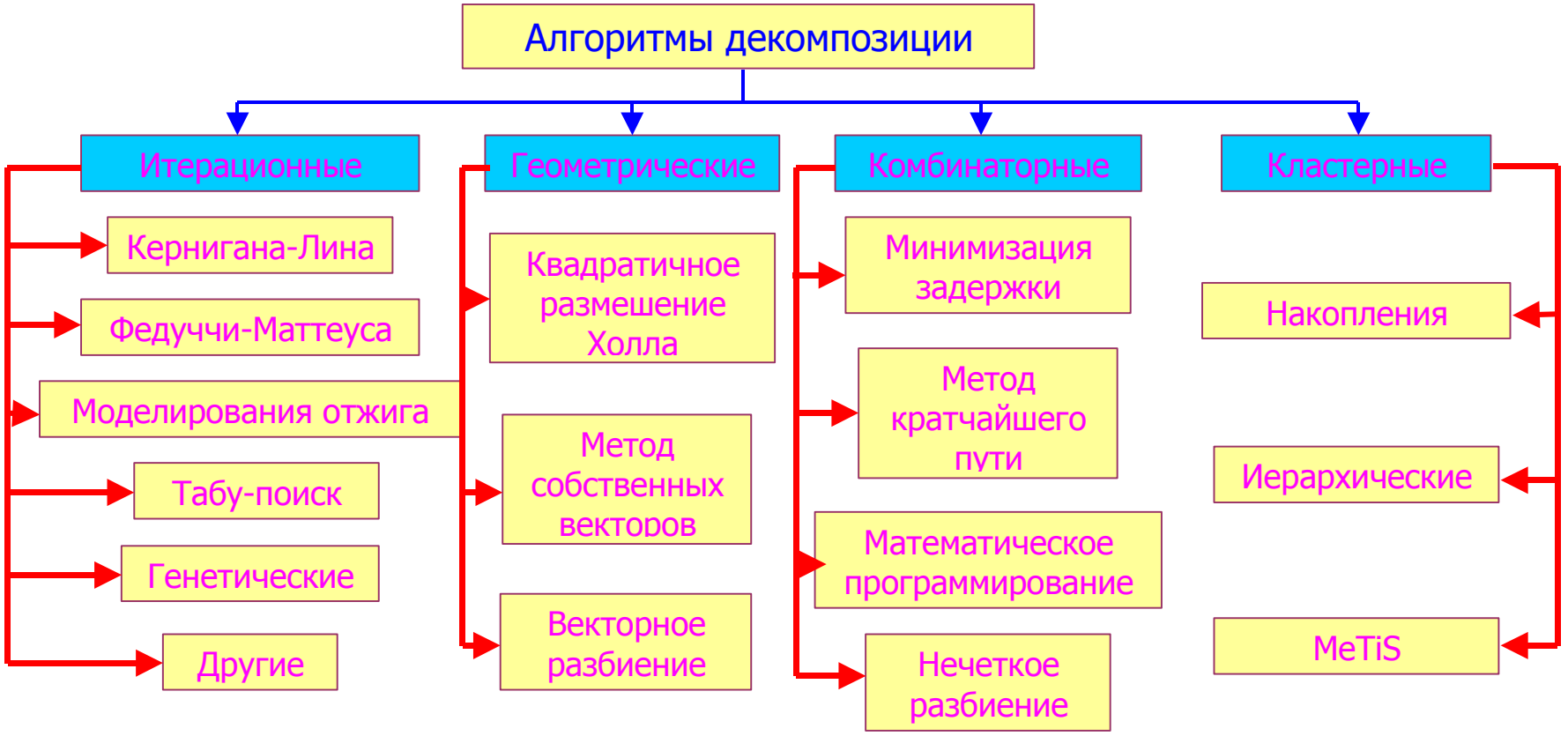
cut size=300

Оптимальный разрез:

$\{(v_1,v_2),(v_3,v_4,v_5,v_6)\}$;

cut size=19

Классификация алгоритмов декомпозиции электрической схемы



Итерационные (moved-based) алгоритмы

Основные парадигмы:

i. *Структура соседства* (neighborhood structure) – определяет способы локального изменения текущего решения;

ii. *Предистория* – может быть использована для выбора лучшего продолжения.

Преимущества:

- естественность
- простота

Алгоритм Кернигана-Лина (KL, 1970)

- Каждый модуль двигается строго один раз. KL итеративно меняет местами пару модулей с максимальным значением целевой функции

Сложность алгоритма – $O(n^3)$

Алгоритм Кернигана-Лина (определения)

- Определение 6: *External edge cost* - для вершины $v_i \in A$:
- A,B – начальное разбиение

$$E(i) = \sum_{\substack{e=\{v_i, v_j\} \in E \\ v_j \in B}} c(e)$$

- Определение 7: *Internal edge cost* - для вершины $v_i \in A$:

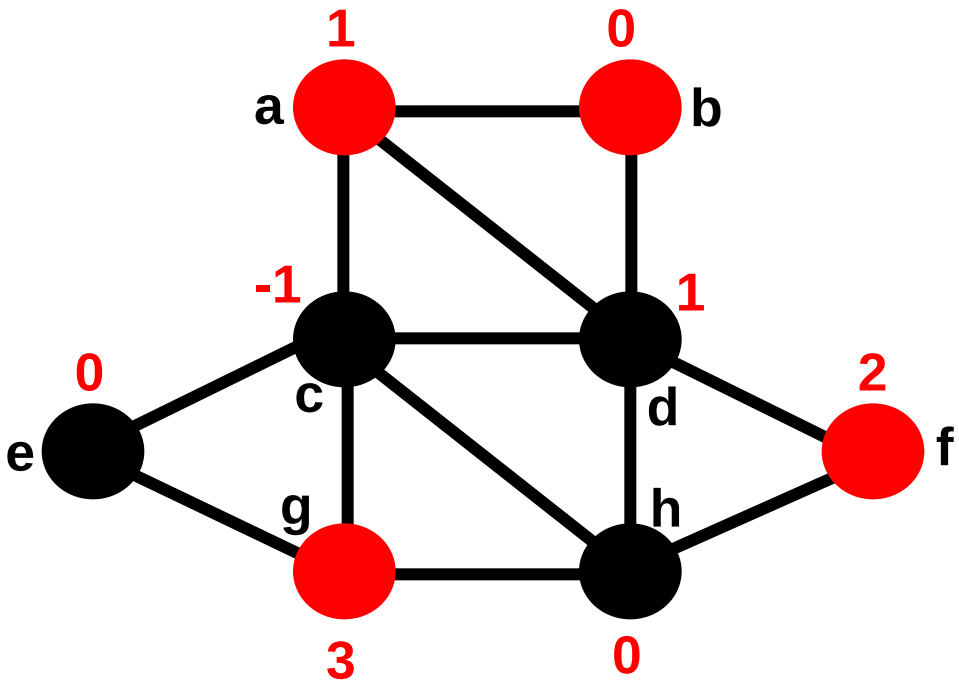
$$I(i) = \sum_{\substack{e=\{v_i, v_j\} \in E \\ v_j \in A}} c(e)$$

- Определение 8: *Gain* - для вершины $v_i \in A$:

$$D(i) = E(i) - I(i)$$

Алгоритм Кернигана-Лина

procedure KL
begin
- Pair:*array*[1:n/2] *of pair of* [1:n];
- Cost:*array*[0:n/2] *of integer*;
- Locked:*array*[1:n] *of Boolean*;
- D:*array*[1:n] *of integer*;
- C:*array*[1:n,1:n] *of integer*;
- BestChange: [1:n/2];
- BestCost: *integer*;
- imin, jmin: [1:n];
- Compute the C and D values;
- *for* i *from* 1 *to* n *do*
  - Locked[i] ← *false* ;
- BestChange ← 0;
- BestCost ← Cost[0] ← cutsize(A,B);
- *for* s *from* 1 *to* n/2 *do*
  - Cost[s] ←∞;
  - *for* i,j *from* 1 *to* n *such that* $v_i \in A$ *and* Locked[i]== *false and* $v_j \in B$ *and* Locked[j]== *false do*
    - *if* 2C[i,j]-D[i]-D[j]< Cost[s] *then*
      - Pair[s] ← (i,j);
      - Cost[0] ←2C[i,j]-D[i]-D[j];
    - (imin, jmin) ← Pair[s];
    - Locked[imin] ← Locked[jmin] ← *true*;
    - *for* i,j *from* 1 *to* n *such that* Locked[i]== *false do*
      - *if* $v_i \in A$ *then* D[i]←D[i]–c[i,jmin]+ c[i,imin]
      - *else* D[i]←D[i]–c[i,imin]+ c[i,jmin];
    - Cost[s] ← Cost[s-1] + Cost[s];
    - *if* Cost[s] < BestCost *then*
      - BestChange ←s;
      - BestCost ← Cost[s];
    - *for* s *from* 1 *to* BestChange *do*
      - Exchange Pair[s];
  - *end*
- *end*



Все возможные пары: 2c[i,j]-D[i]-D[j], c – вес ребра

(a,c,**2**), (a,d,**0**), (a,e,**-1**), (a,h,**-1**);

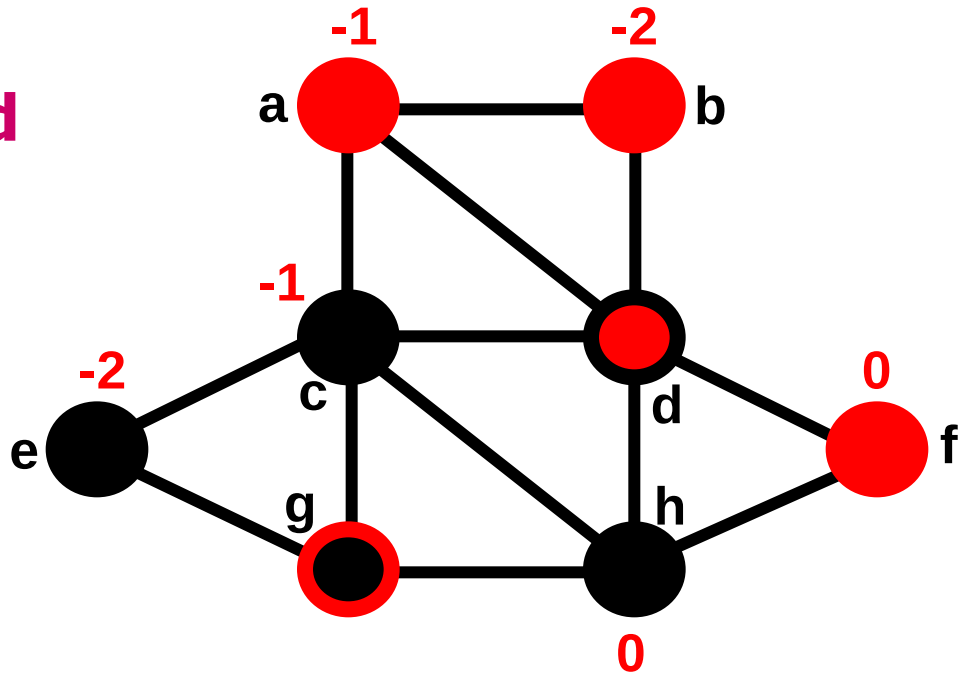
(b,c,**1**), (b,d,**1**), (b,e,**0**), (b,h,**0**);

(g,c,**0**), (g,d,**-4**), (g,e,**-1**), (g,h,**-1**);

(f,c,**-1**), (f,d,**-1**), (f,e,**-2**), (f,h,**0**).

C=8

Обмен: **g ↔ d**



Все возможные пары: 2c[i,j]-D[i]-D[j]

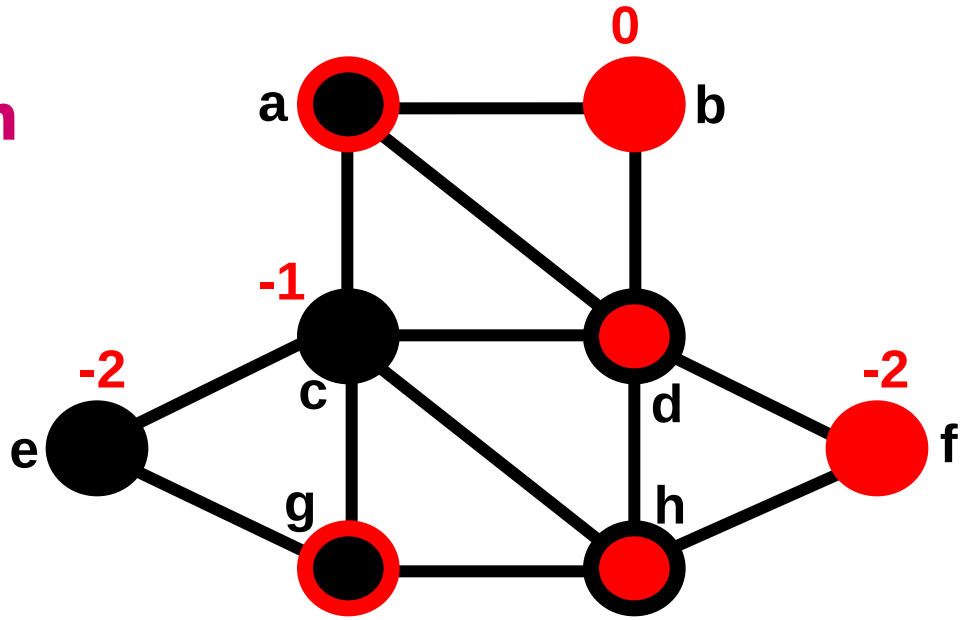
C=4

(a,c,**4**), (a,e,**3**), (a,h,**1**);

(b,c,**3**), (b,e,**4**), (b,h,**2**);

(f,c,**1**), (f,e,**2**), (f,h,**2**).

Обмен: **a ↔ h**



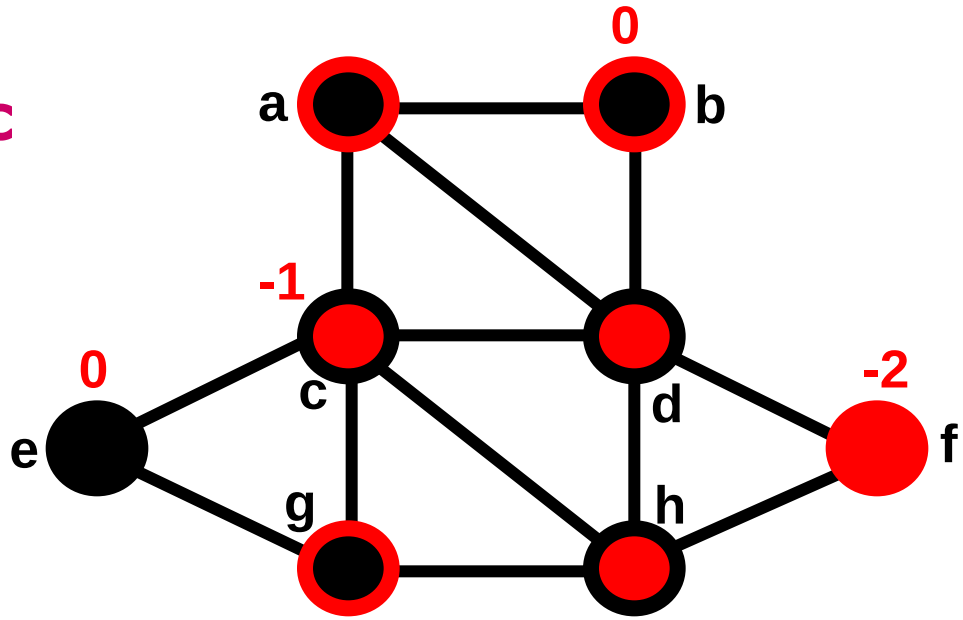
Все возможные пары: 2c[i,j]-D[i]-D[j]

(b,c,**1**), (b,e,**2**);

(f,c,**3**), (f,e,**4**).

C=5

Обмен: **b ↔ c**



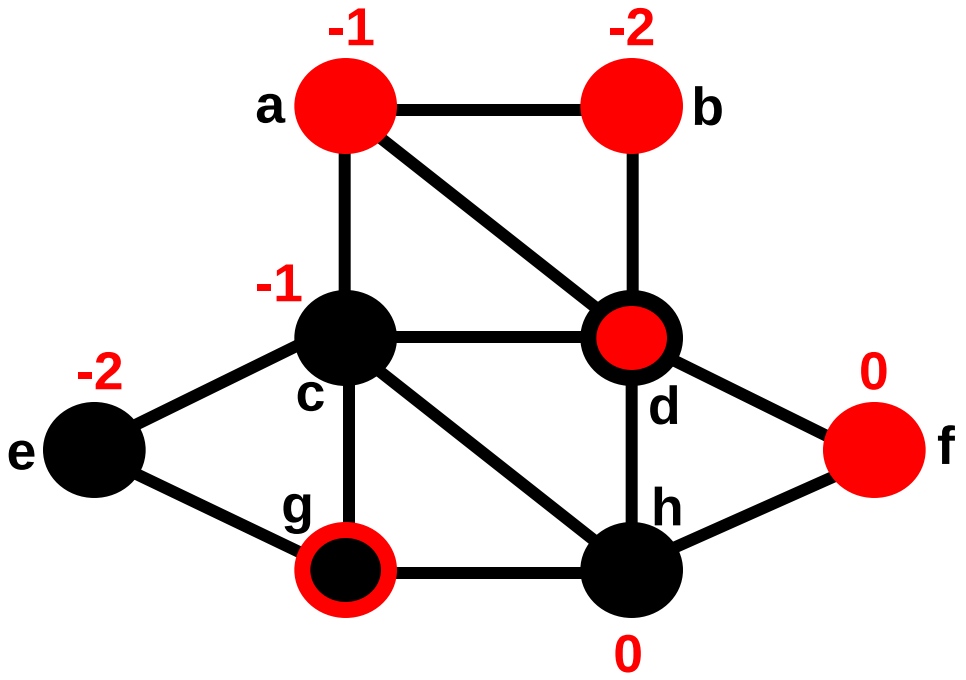
Все возможные пары: 2c[i,j]-D[i]-D[j]

(f,e,**2**).

C=6

Лучшее разбиение

Следующий этап: повторить алгоритм для другого начального разбиения



C=4

Анализ алгоритма

1. Все вершины графа должны быть единичного веса. Этого недостаточно для применения к схемам, т.к. размеры блоков неодинаковы.
2. Разбиение должно быть строго сбалансировано. Для расширения нужно ввести фиктивные изолированные вершины.
3. Алгоритм не может быть применен к гиперграфам.
4. Сложность алгоритма слишком высока.
5. Эвристика включает в себя неопределенность, что может приводить к выбору плохих обменов и попаданию в локальный минимум.
6. Алгоритм эффективен для плотных графов (с большим числом ребер у вершины).

Алгоритм Федуччи-Маттеуса (FM, 1982)

Главные отличия FM от KL:

- Перемещается один модуль за одну итерацию, что позволяет получать несбалансированные решения
- Понятие *gain* расширено до гиперграфов
- Специальный способ для выбора вершин снижает вычислительную сложность алгоритма

Сложность алгоритма – O(n)

Алгоритм Федуччи-Маттеуса (определения)

- Определение 9: *External hyperedge cost* - для вершины $v_i \in A$:

$$E(i) = \sum_{e \in E_{ext,i}} c(e)$$

- Определение 10: *Internal hyperedge cost* - для вершины $v_i \in A$:

$$I(i) = \sum_{e \in E_{int,i}} c(e)$$

- $E_{ext,i} = \{e \in E / \{v_i\} = e \cap F | A\}$ – множество гиперребер, которые будут удалены после переноса вершины;
- $E_{int,i} = \{e \in E / v_i \in e \text{ и } e \cap B = \varnothing\}$ – множество добавленных гиперребер

Алгоритм Федуччи-Маттеуса

Алгоритм

procedure FM

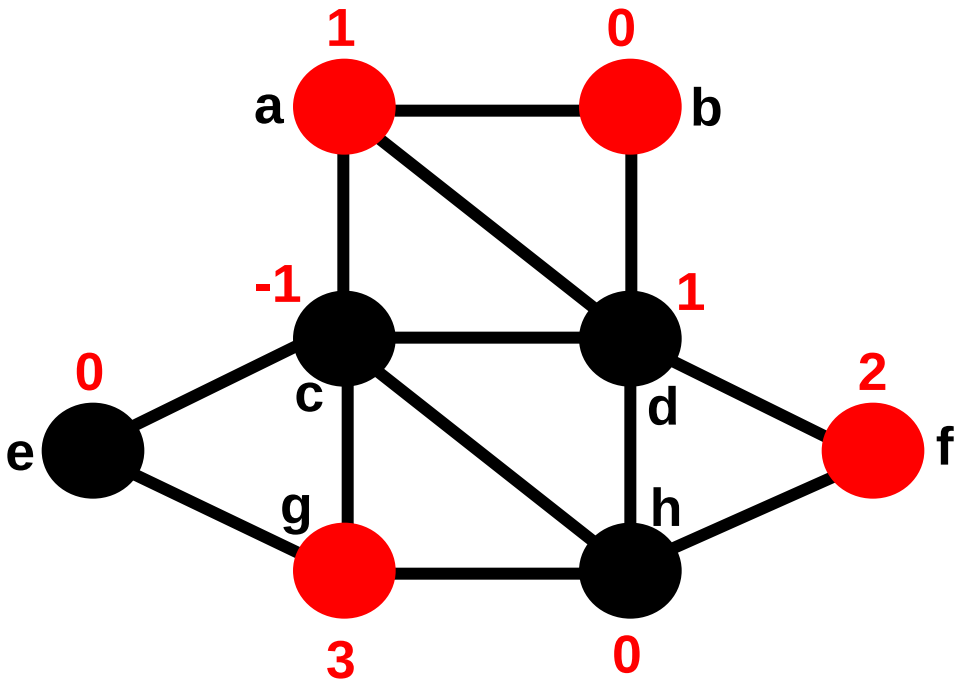
begin

- Инициализировать *D*
- *while* перемещения вершин возможны *do*
  - Выбрать разрешенную вершину *v*
  - Обновить структуру данных с учетом перемещения *v*
  - Расширить список перемещений
- Вычислить префикс последовательности перемещений, достигающих минимального сечения

end

Пример (1)

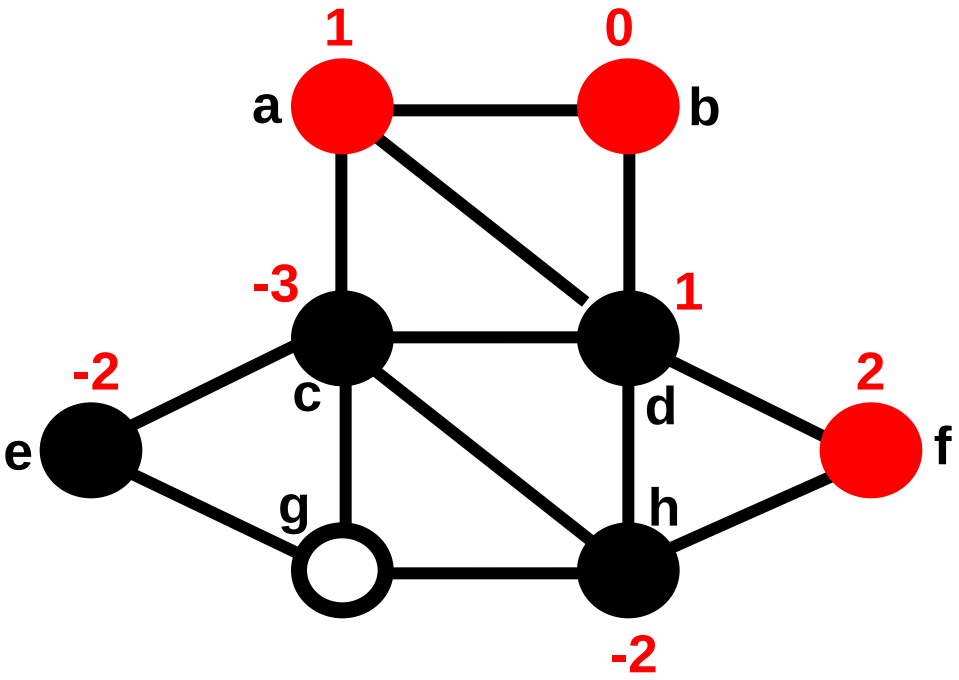
Начальное разбиение: {**A,B**};



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8);

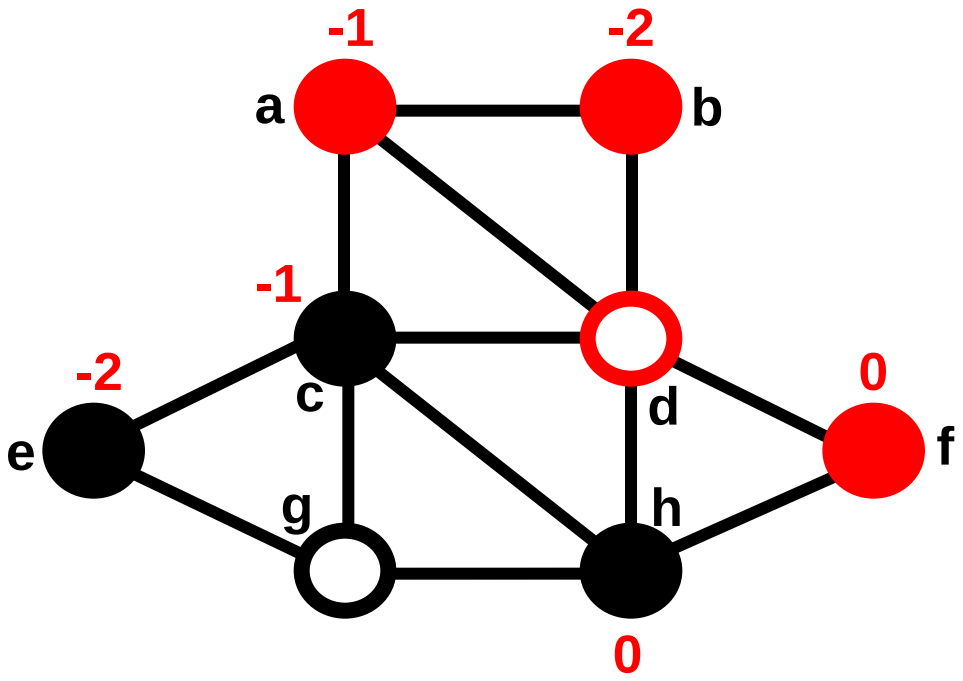
Пример(2)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g

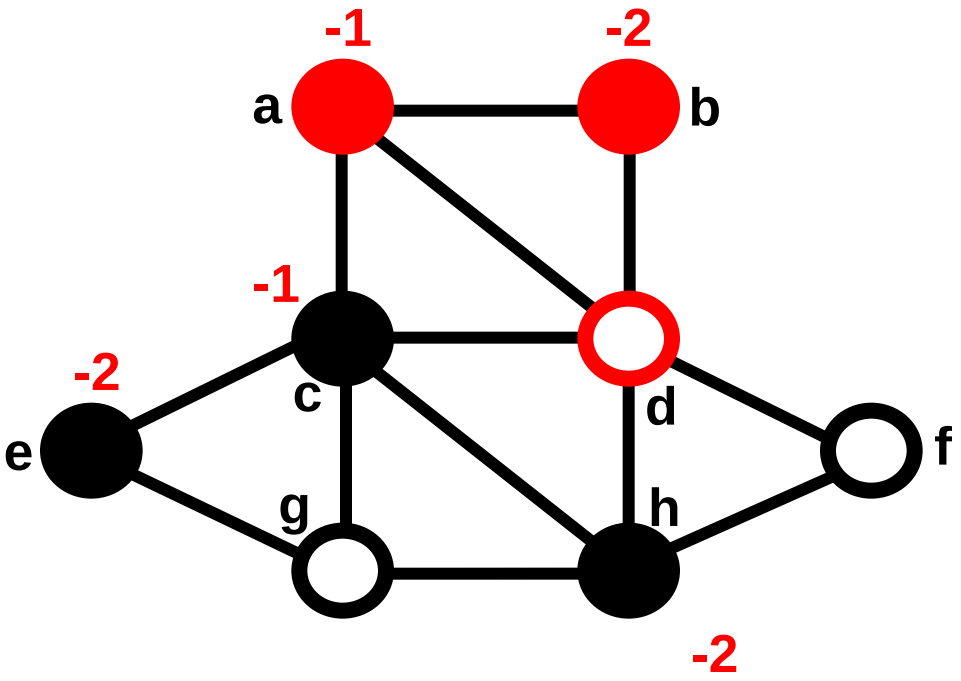
Пример(3)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4);

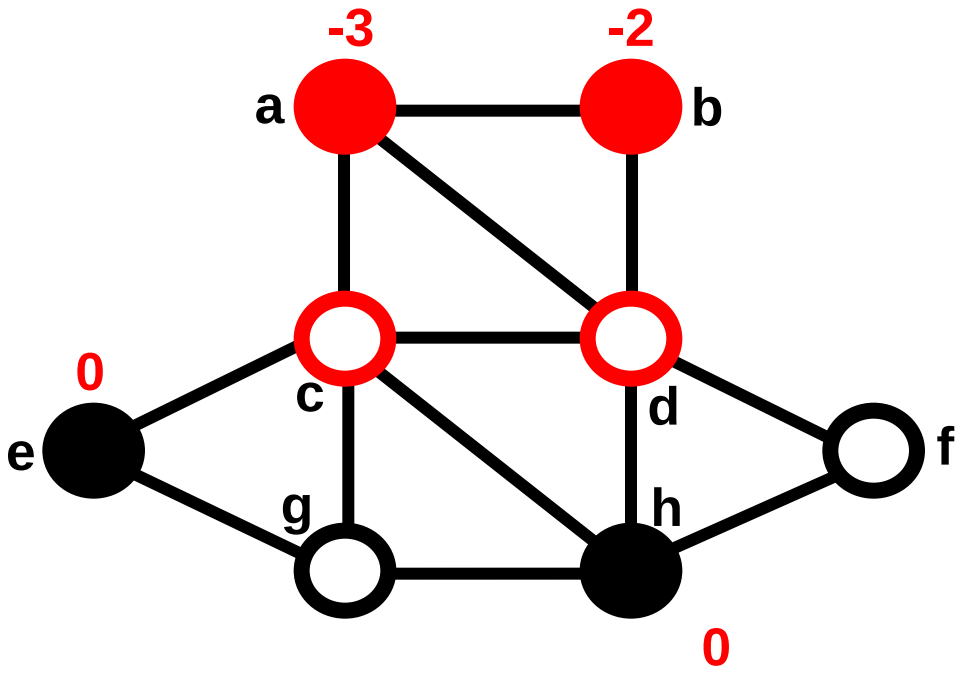
Пример(4)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f

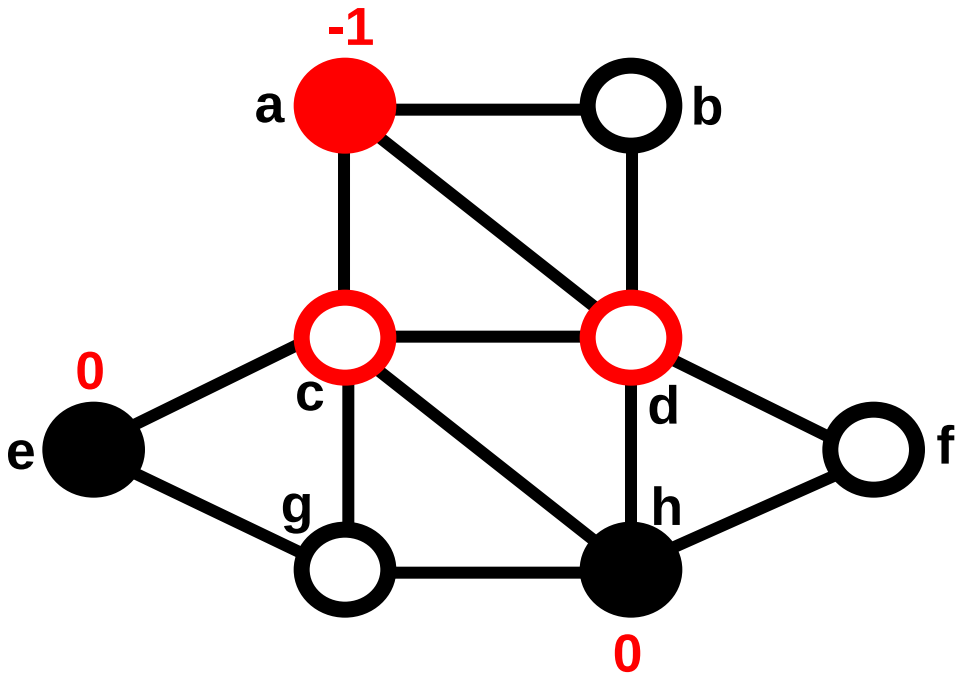
Пример(5)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f, c (5);

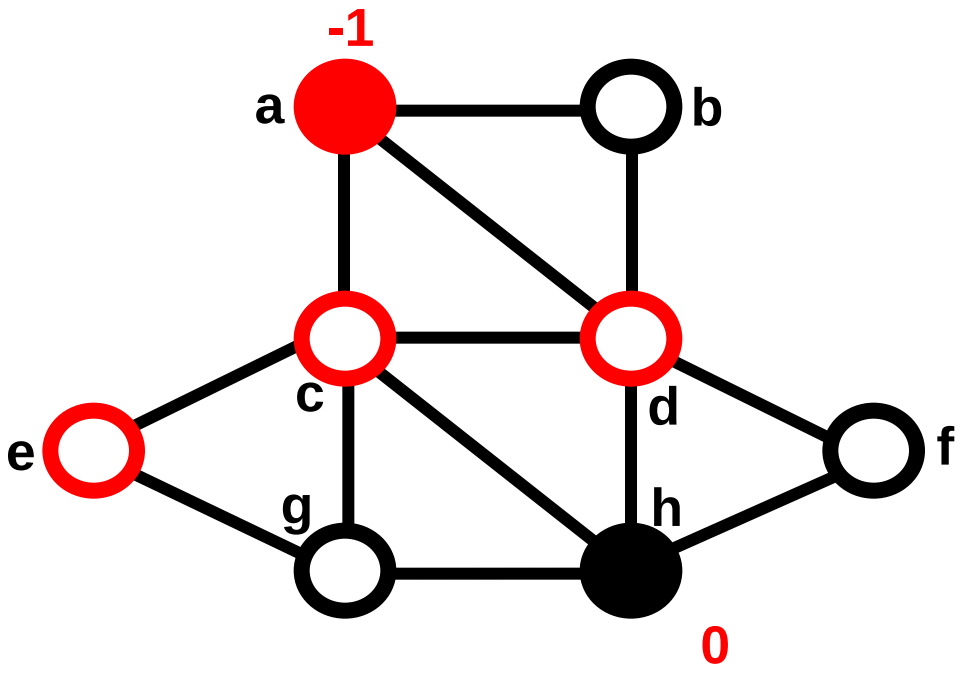
Пример(6)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f, c (5); b

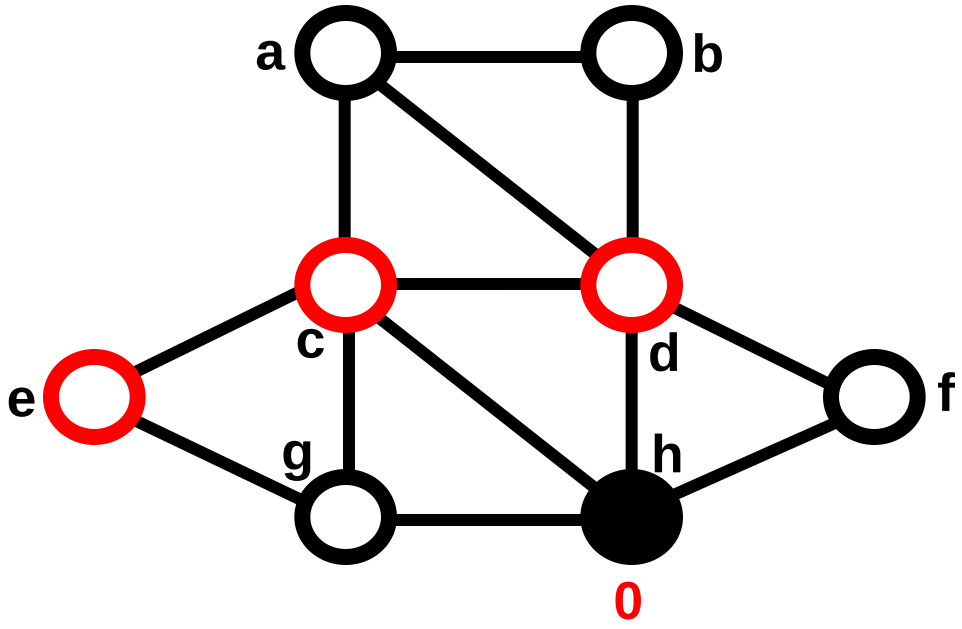
Пример(7)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f, c (5); b, e (7);

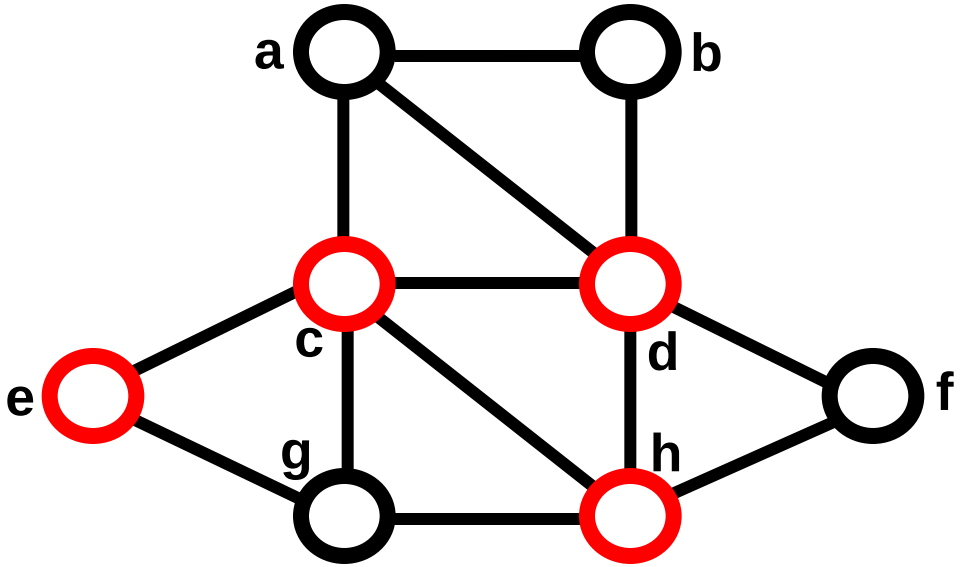
Пример(8)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f, c (5); b, e (7); a

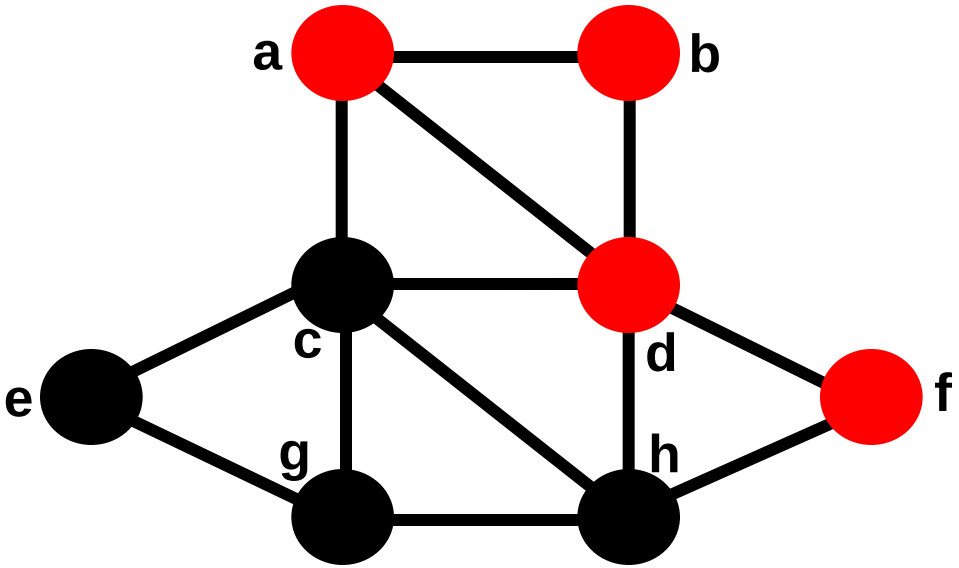
Пример(9)



Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); g, d (4); f, c (5); b, e (7); a, h (8)

Пример(10)



Выбирается наилучший вариант

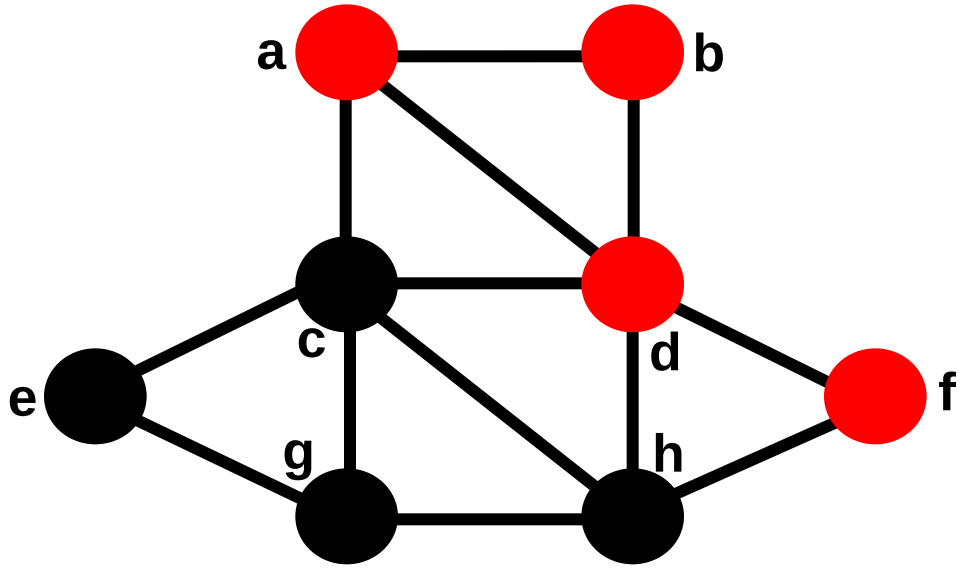
Перемещенные вершины (и стоимость разреза после перемещения каждой пары):

none (8); **g, d (4);** f, c (5); b, e (7); a, h (8)

Пример(11)

Процесс итеративного улучшения повторяется.

Улучшение не найдено.



FM: недостатки алгоритма

FM порождает много неопределенных вариантов, также как и *KL*.

Вершины, которые *FM* не может различить: v1 ? v2

Какую вершину выгоднее переместить?