Классическая задача о назначениях

Задано n работ и n исполнителей, причем каждую из работ может выполнять любой из исполнителей. Стоимость выполнения i-ой работы j-ым исполнителем определяется величиной $c_{ij} \geq 0$. Требуется распределить исполнителей на выполнение работ так, чтобы все работы были выполнены, каждый исполнитель выполнял только одну работу, и стоимость выполнения всех работ была бы минимальной.

Построим математическую модель для данной задачи. Пусть

$$x_{ij} = \begin{cases} 1, \text{если } i \text{ - ый исполнитель назначен на } j \text{ - ую работу;} \\ 0, \text{в противном случае.} \end{cases}$$

Тогда задача может быть сформулирована в виде следующей задачи булевого программирования

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij} \to \min ,$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1,n} , \quad \sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,n} ,$$

$$x_{ij} = 0 \vee 1 .$$

Можно привести ещё одну интерпретацию постановки задачи. Построим полный двудольный граф $G=(V_1 \cup V_2, E)$, вершины первой доли которого соответствуют исполнителям, второй – работам. Припишем ребру (v_i, u_j), v_i из V_1, u_j из V_2, вес c_{ij}. Тогда задача заключается в поиске совершенного (покрывающего все вершины графа) паросочетания с минимальным весом.

Приведём алгоритм решения задачи, использующий в качестве вспомогательного алгоритм Форда-Фалкерсона нахождения максимального потока в сети.

Начальный шаг. Находим в каждом строке матрицы стоимостей C минимальный элемент и вычитаем его из всех элементов этой строки. Затем находим минимальный элемент в каждом столбце и вычитаем его из элементов данного столбца. Проделанные выше процедуры называются *приведением матрицы*, а полученная матрица *C'- приведенной*.

Легко показать, что решения задач для исходной и для приведенной матриц совпадают.

Нулевые элементы приведенной матрицы называются *допустимыми*. Если бы можно было выбрать по одному нулевому элементу в каждом столбце и каждой строке, то это и было бы оптимальным назначением.

Задачу выбора нулевых клеток можно решить с помощью алгоритма нахождения максимального потока.

Шаг 1. Для приведенной матрицы строим сеть с $(2n + 2)$-мя вершинами: источником s, множеством вершин $S = \{ s_1 , \dots , s_n \}$, соответствующими строкам матрицы стоимостей, множеством вершин $T = \{ t_1 , \dots , t_n \}$, соответствующими столбцам матрицы стоимостей, и стоком t . Источник s соединяем дугой с каждой вершиной s_i если $c'_{ij} = 0$, каждую вершину t_j соединяем дугой со стоком t . Добавляем дугу (s_i , t_j) тогда и только тогда, когда $c'_{ij} = 0$. Пропускные способности всех дуг полагаем равными 1.

Шаг 2. Находим максимальный поток в сети из s в t. Если величина найденного потока равна n, то решение задачи получено. В оптимальном назначении $x_{ij} = 1$ тогда и только тогда, когда дуга (s_i , t_j) существует и поток по ней равен 1.

Если максимальный поток меньше n, то переходим на шаг 3.

Шаг 3. Необходимо преобразовать сеть, добавив какую-либо новую дугу (или дуги). Естественно, следует добавить такую дугу, которая увеличивала бы поток, а соответствующий ей элемент c'_{ij} был бы наименьшей.

Пусть при нахождении максимального потока на последнем шаге алгоритма вершины $S' \subset S$ множества $T' \subset T$ оказались *помеченными*, а вершины множества *непомеченными*. Пусть S'' (T'') номера вершин, входящих в S' (T'). Находим

$$\bar{c} = \min_{i \in S'', j \in T''} (c_{ij})$$

Шаг 4. Вычитаем элемент $\bar{c}$ из всех элементов в строках, соответствующих вершинам из S' и добавляем к элементам столбцов соответствующих вершинам из $T \backslash T'$.

Шаг 5. Переходим к шагу 1 с матрицей, полученной на шаге 4.

В результате работы алгоритма в матрице стоимостей на каждой итерации появляется хотя бы один новый ноль (на месте элементов равных $\bar{c}$), т.е. в сети появляется хотя бы одна новая дуга.

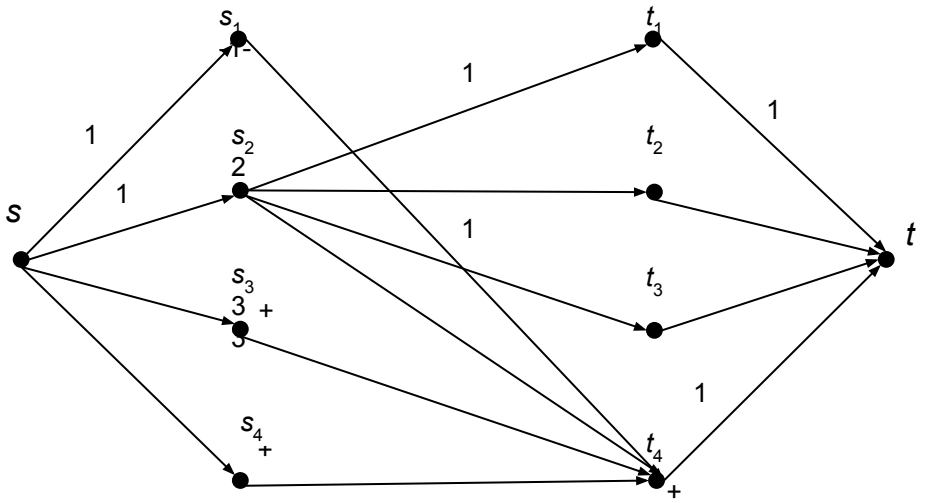
Пример. Решить задачу о назначении со следующей матрицей стоимостей:

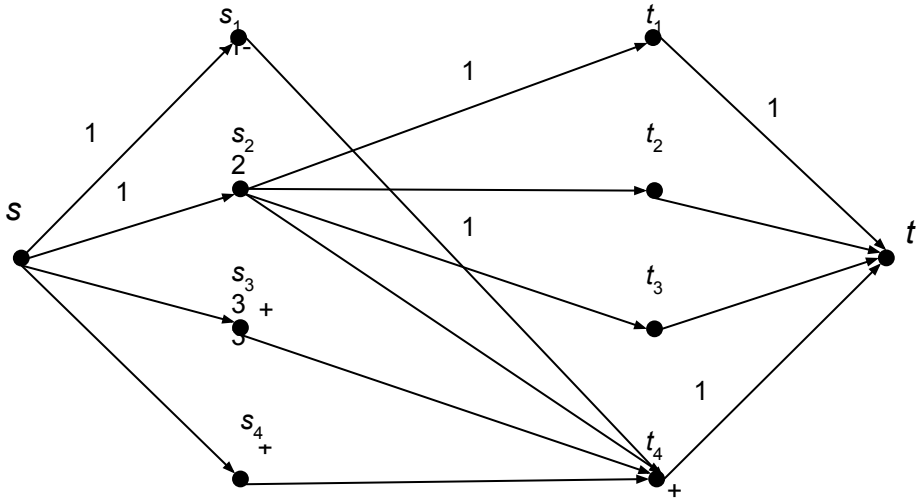
$$C=\begin{bmatrix}7 & 5 & 6 & 3\\ 2 & 1 & 2 & 1\\ 5 & 5 & 5 & 2\\ 4 & 4 & 5 & 2\end{bmatrix}$$

Начальный шаг. Приводим матрицу C сначала по строкам, затем − по столбцам. Приведенная матрица C' имеет вид

$$C'=\begin{bmatrix}3 & 2 & 2 & 0\\ 0 & 0 & 0 & 0\\ 2 & 3 & 2 & 0\\ 1 & 2 & 2 & 0\end{bmatrix}$$

Итерация 1. Строим для приведенной матрицы C' сеть





На последней итерации алгоритма Форда-Фалкерсона получим:

s	s_1	s_2	s_3	s_4	t_1	t_2	t_3	t_4	t	v
$(-,\infty)$	$(t_4^-,1)$		$(s^+,1)$	$(s^+,1)$				$(s_3^+,1)$		2

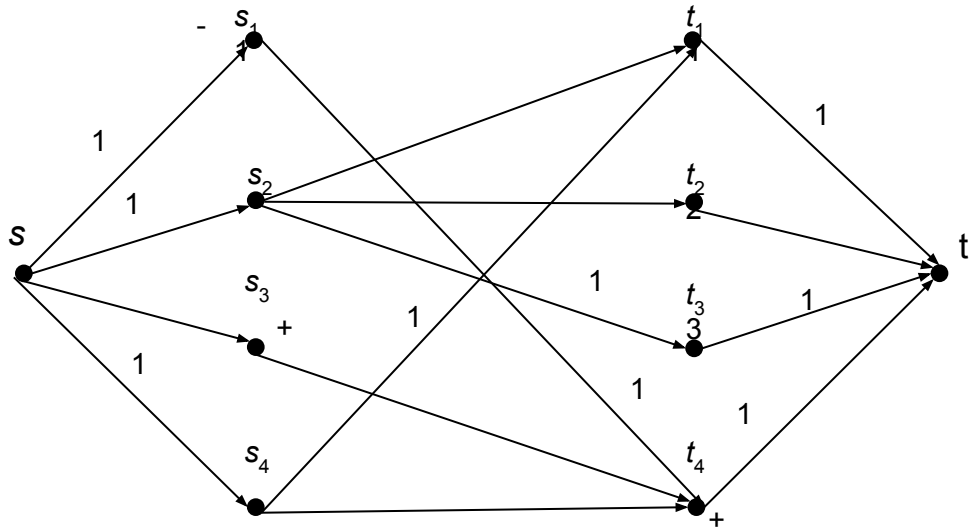
Значение максимального потока $v = 2$. Это меньше 4, поэтому необходимо преобразовать сеть. Имеем: $S'' = \{1, 3, 4\}$, $T'' = \{1, 2, 3\}$. Находим $\bar{c} = \min_{i \in S'', j \in T''} (c_{ij})$ 1.

Преобразуем приведенную матрицу, вычитая элемент из всех элементов в строках с номерами из S'' и добавляя к элементам столбцов с номерами, не вошедшими в T''.

Матрица C' примет вид

$$C' = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Итерация 2. Получим следующую сеть:



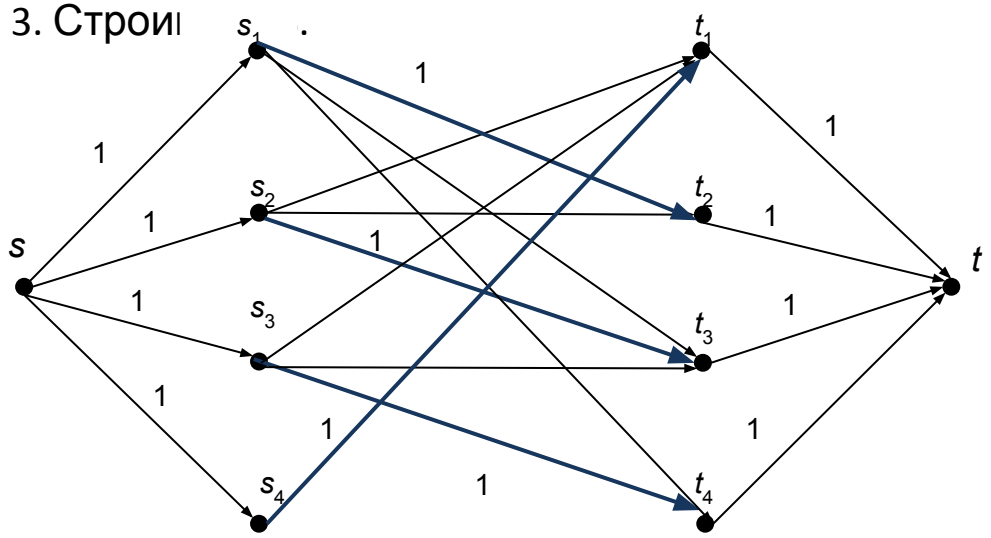
Находим максимальный поток в сети. На последней итерации алгоритма Форда Фалкерсона получим:

s	s_1	s_2	s_3	s_4	t_1	t_2	t_3	t_4	t	v
$(-,\infty)$	$(t_4^-,1)$		$(s^+,1)$					$(s_3^+,1)$		3

Имеем: v=3<4, S'' = {1,3}, T''= {1,2,3}. Находим = 1. После преобразования получим матрицу

$$C' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Итерация 3. Строи



Максимальный поток в сети равен 4. Следовательно, решение получено. На сети соответствующие дуги с единичным потоком выделены. Получаем назначение: x_{12} = 1, x_{23} = 1, x_{34} = 1, x_{41} = 1. Значение целевой функции равно 13.

Примечания. 1. Можно рассматривать задачу о назначениях, в которой количество исполнителей больше числа работ. В этом случае матрицу дополняют нулями до квадратной матрицы и решают задачу обычным методом.

2. В некоторых случаях задачу о назначении необходимо решать на максимум. Тогда поступают следующим образом: находят $c_0 = \max_{ij} c_{ij}$; переходят к матрице с элементами $c'_{ij} = c_0 - c_{ij}$; решают для этой матрицы задачу минимизации.

Задача о назначениях на узкие места

Имеется *n* лиц и *n* работ. Заданы эффективности исполнения каждым лицом каждой работы. Каждый исполнитель должен назначаться на выполнение только одной работы и все работы должны быть выполнены. Требуется найти такое назначение исполнителей на работы, при котором наименьшая эффективность выполнения работ максимальна.

Если, например, на конвейере имеется *n* рабочих мест и *n* работников, каждый из которых может выполнять любую работу за некоторое заданное время, то для достижения максимальной скорости движения конвейера надо распределить работников на конвейере так, чтобы минимальное из времён выполнения работ было максимально.

То назначение исполнителя на работу, на котором реализуется минимальная эффективность, называют *узким местом в назначении.*

Нетрудно видеть, что каждое назначение задается взаимно однозначным отображением множества {1,2,…,n} на себя, т.е. образует подстановку

$$\begin{pmatrix} 1 & 2 & \dots & n \\ p(1) & p(2) & \dots & p(n) \end{pmatrix},$$

где $p(i)$ указывает номер назначаемой i-ому исполнителю работы. Количество всевозможных назначений работников на работы равно $n!$

Рассмотрим **алгоритм Гросса** для решения задачи.

Начальный шаг. Фиксируем любое назначение P_0, что соответствует выбору в строках и столбцах матрицы эффективностей $A = \left| a_{ij} \right|_{n \times n}$ ровно по одному элементу. При этом назначении вычисляется значение целевой функции, т.е. вычисляется величина $F(P_0) = \min\{a_{1p_0(1)}, \ldots, a_{np_0(n)}\}$.

Обозначим эту величину через *s*.

Шаг 1. Строим двудольный граф $G=(V_1 \cup V_2, E)$, вершины первой доли которого соответствуют исполнителям, второй – работам. Ребро $(v_i, u_j) \in E$ тогда и только тогда, когда $a_{ij} > s$.

Шаг 2. Ищем максимальное (по числу рёбер) паросочетание в графе $G=(V_1 \cup V_2, E)$.

Если паросочетание имеет ровно *n* рёбер, то по ним строим новое назначение с более высокой минимальной эффективностью (следует из способа построения двудольного графа). Обозначим ее снова через *s* и вернемся к шагу 1.

Если же число ребер в паросочетании окажется меньше *n*, то имеющееся назначение оптимально.

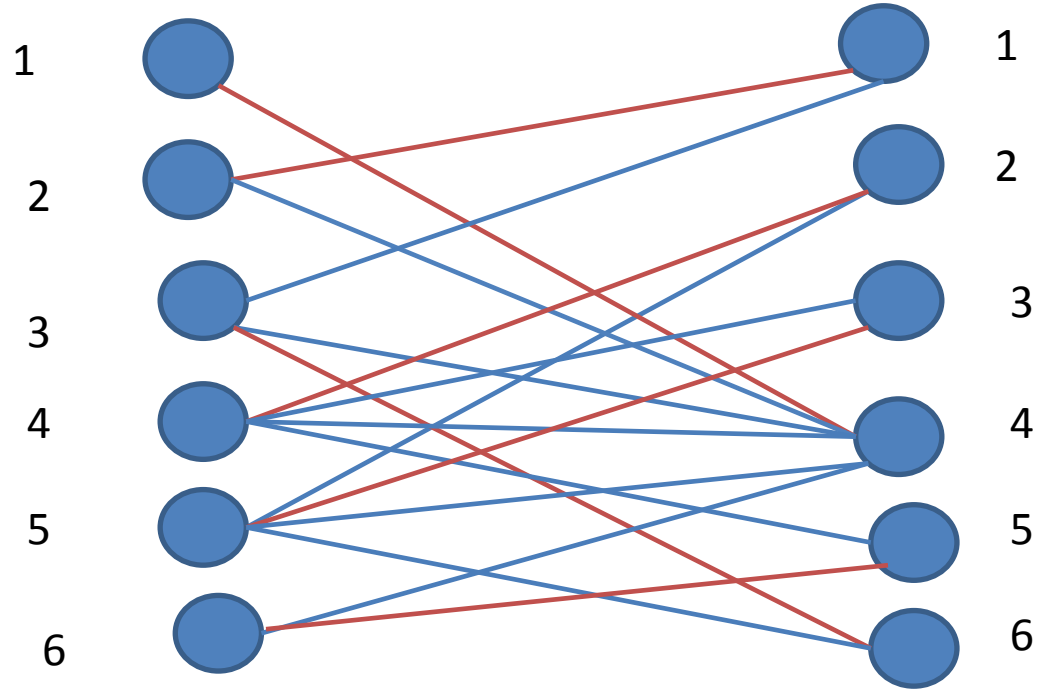
Пример. Пусть задана матрица эффективностей:

$$A = \begin{vmatrix} 1 & 3 & 2 & 6 & 0 & 1 \\ 4 & 2 & 3 & 8 & 3 & 1 \\ 8 & 1 & 1 & 5 & 0 & 9 \\ 3 & 4 & 4 & 8 & 8 & 3 \\ 2 & 9 & 9 & 5 & 2 & 9 \\ 3 & 3 & 3 & 6 & 7 & 1 \end{vmatrix}$$

Возьмём назначение

$$P_0 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 4 & 3 & 6 & 5 \end{pmatrix}$$

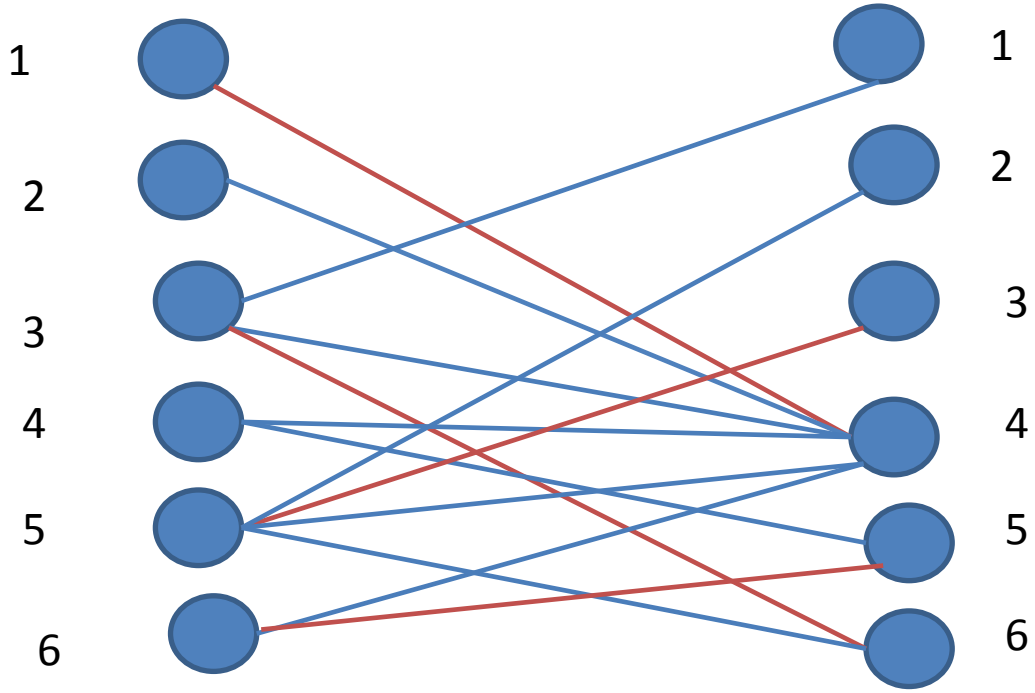
Имеем $F(P_0) = 3$. Строим двудольный граф:



Количество рёбер в максимальном паросочетании равно 6, поэтому находим новую подстановку

$$P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 1 & 6 & 2 & 3 & 5 \end{pmatrix}.$$

Для этой подстановки имеем $F(P_1) = 4$.



Двудольный граф имеет максимальное паросочетание с 4 ребрами. Следовательно, назначение P_1 оптимально.

Для построения максимального паросочетания в двудольном графе можно использовать алгоритм Форда-Фалкерсона для сети, полученной из исходного двудольного графа добавлением источника, связанного с каждой вершиной первой доли, и стока, связанного с вершинами второй доли. Пропускные способности дуг полагаем равными единице. Очевидно, что величина максимального потока в сети будет равна мощности максимального паросочетания в исходном графе, причём насыщенные рёбра от вершин первой доли к вершинам второй, дают рёбра этого паросочетания.

Приведём **алгоритм Кёнига-Эгервари**, построения максимального паросочетания в двудольном графе. Пусть $G=(V_1 \cup V_2, E)$ двудольный граф с $|V_1| = n$, $|V_2| = m$.

Начальный шаг. Строим таблицу размером $n \times m$, строки которой соответствуют вершинам первой, а столбцы – вершинам второй долей. В клетку (i, j) ставим символ * и называем её недопустимой, если в графе нет ребра (v_i, u_j) для вершин $v_i \, V_1$, $u_j \, V_2$. Если $(v_i, u_j) \in E$, то клетку оставляем свободной и называем допустимой. Множество допустимых клеток называем независимым, если среди них нет двух, стоящих в одной строке или одном столбце. Очевидно, что между множествами независимых допустимых клеток построенной таблицы и паросочетаниями исходного двудольного графа существует

Шаг 1. Строим произвольное множество независимых допустимых клеток, помещая в

вошедшие в него клетки символ «1». Например, просмотром в порядке возрастания

номеров строк слева направо и фиксацией «1» в первой по ходу просмотра допустимой

клетке, которая является независимой по отношению к допустимым клеткам, отмеченных

ранее.

Если окажется, что во всей таблице все клетки недопустимы, то это означает, что в графе

нет ребер.

Шаг 2. Находим в таблице строки без символа «1» и помечаем их символом «-» и

переходим к следующему шагу. Если в каждой строке таблицы окажется символ «1», то

рёбра соответствующего паросочетания составляют наибольшее паросочетание, и действия

окончены.

Шаг 3. Просмотрим все получившие пометки строки в порядке возрастании их

номеров. Просмотр очередной строки состоит в следующем: в строке отыскиваются

допустимые клетки, и столбцы, в которых эти клетки расположены, помечаются меткой

(+*p*), где *p* - номером просматриваемой строки. При этом соблюдается принцип: если

пометка уже стоит, то на ее место вторая не ставится.

Если в результате ни один из не имевших метку столбцов не будет помечен, то

Шаг 4. Просмотрим помеченные на шаге 3 столбцы в порядке возрастания их номеров.

Просмотр столбца состоит в следующем: в столбце отыскивается символ «1» и строка, в

которой он расположен, помечается меткой (-*h*), где *h* - номер просматриваемого столбца.

При этом соблюдается прежний принцип: если столбец уже помечен, то метка не

изменяется.

Если возникает ситуация, когда в просматриваемом столбце нет символа «1», то

действия на данном шаге прерываются, и осуществляется переход к следующему шагу 5.

Если же в результате действий шага 4 будут просмотрены все помеченные столбцы и,

возникнет набор новых помеченных строк, то следует вернуться к Шагу 3.

Наконец, если в результате действий шага 4 не возникнет новых помеченных строк, то

это означает, что имеющееся паросочетание является искомым.

Шаг 5. Производим изменение множества независимых допустимых клеток в таблице.

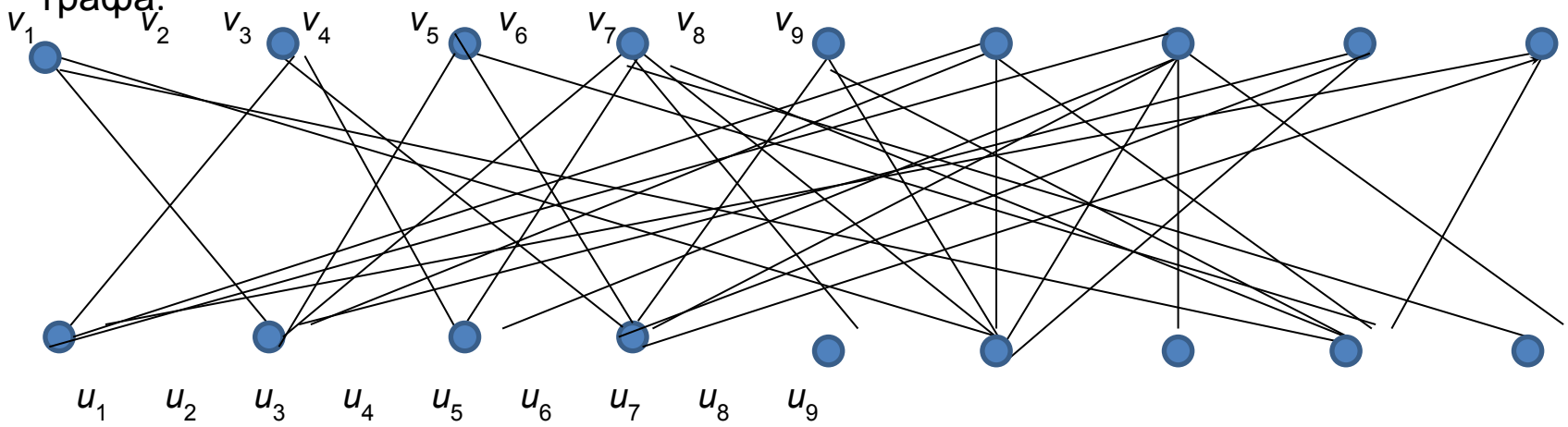
Рассматриваем столбец, имеющий пометку и не содержащий символа «1». В нем ставим

символ «1» в строку, номером которой помечен этот столбец. Затем в этой строке ищем

«старый» символ «1» и перемещаем его по столбцу в строку, номер которой равен пометке

при этом столбце. Далее в строке, куда попал последний символ «1» ищем «старый»

Пример. Найти максимальное паросочетание для следующего двудольного графа:



Начальный и первый шаг. Соответсвующая таблица с первоначальным множеством независимых допустимых клеток имеет вид:

	1	2	3	4	5	6	7	8	9	
1	*	1	*	*	*		*		*	-2
2	1	*			*	*	*	*	*	-1
3	*		*	1	*	*	*		*	-4
4	*		1				*			-3
5	*	*	*		*	1	*		*	-6
6	*		*	*	*		*	1	*	-8
7		*			*		1	*		
8	*		*		*		*	*	*	–
9		*	*		*	*	*		*	–
	+9	+8	+2	+8	+4	+8		+9	+4	

Выполнение шага 5 увеличивает количество независимых допустимых клеток до 8.

	1	2	3	4	5	6	7	8	9	
1	*	1	*	*	*		*		*	-2
2		*	1		*	*	*	*	*	
3	*		*	1	*	*	*		*	-4
4	*				1		*			
5	*	*	*		*	1	*		*	-6
6	*		*	*	*		*	1	*	-8
7		*			*		1	*		
8	*		*		*		*	*	*	–
9	1	*	*		*	*	*		*	
		+8		+8		+8		+1		

Алгоритм завершён.