

# Методы оптимизации

(МО)

Парамонов Александр Иванович

[alex-in-spb@yandex.ru](mailto:alex-in-spb@yandex.ru)

---

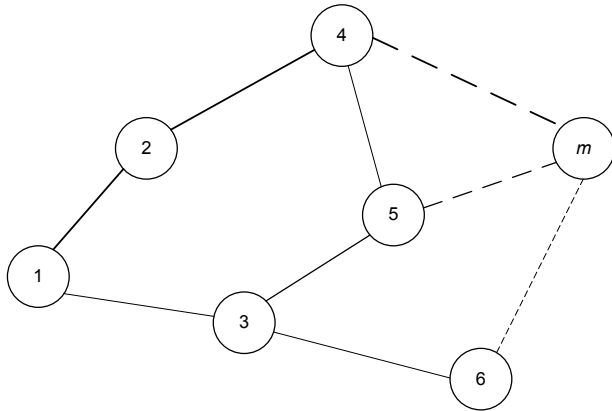
# Содержание

- 4. Примеры постановки задачи оптимизации
  - 4.1 Качество обслуживания в сети с КК
  - 4.2 Надежность сети связи
- 5. Аналитические методы оптимизации
  - 5.1 Экстремумы функции
  - 5.2. Безусловная оптимизация
  - 5.3. Условная оптимизация
    - 5.3.1 Метод множителей Лагранжа
    - 5.3.2 Выпуклые функции
    - 5.3.3 Условия Каруша-Куна-Таккера (ККТ)
- 6. Численные методы оптимизации
  - 6.1 Общий алгоритм численных методов
  - 6.2 Оптимизация функции одной переменной
    - 6.2.1 Прямой поиск
    - 6.2.2 Дихотомия
    - 6.2.3 Метод золотого сечения
    - 6.2.4 Метод чисел Фибоначчи
  - 6.3 Оптимизация функции нескольких переменных
    - 6.3.1 Покоординатный спуск (функция нескольких переменных)
    - 6.3.2 Метод Хука-Дживса
    - 6.3.3 Симплекс метод Нелдера-Мида
    - 6.3.4 Некоторые другие методы оптимизации выпуклых функций
  - 6.4 Стохастические методы
    - 6.4.1 Слепой случайный поиск
    - 6.4.2 Эволюционный метод (генетический алгоритм)
  - 6.5 Динамическое программирование

# 4. Примеры постановки задачи оптимизации

## 4.1 Качество обслуживания в сети с КК

Рассматривается сеть связи с коммутацией каналов (КК). Структура сети задана графом. Сеть состоит из  $m$  узлов связи и  $n$  направлений связи (ребер графа).



$x_j$  Число каналов в  $j$  направлений связи

$y_j$  Интенсивность нагрузки в  $j$  направлений связи (Эрл)

$N$  Общее число каналов в сети = *const*

### 1. Формулировка задачи

- предметная область - качество обслуживания (вероятность потерь вызовов)
- состояние сети задано исходными данными
- требуется найти оптимальное число каналов в каждом из направлений связи для получения минимальных потерь вызовов в сети, при заданном общем числе каналов  $N$ .

### 2. Построение модели системы

- полагаем, что в направлениях сети имеют место простейшие потоки вызовов
- модель направления связи может быть представлена первой формулой Эрланга.

$$p_j = \frac{\frac{y_j^{v_j}}{v_j!}}{\sum_{k=0}^{v_j} \frac{y_j^k}{k!}}; \quad j = 1 \dots n$$

### 3. Выбор параметров управления и показателей состояния

-параметры управления: число каналов в каждом из направлений связи,  $x_j$

-показатели состояния: потери в каждом из направлений связи.  $P_j$

### 4. Построение целевой функции

Выберем в качестве целевого показателя средневзвешенную вероятность потери вызова в сети

$$\bar{p} = \sum_{j=1}^n \frac{y_j}{y_{\Sigma}} \cdot p_j; \quad \text{где} \quad y_{\Sigma} = \sum_{j=1}^n y_j;$$

Целевая функция

$$x_j = \arg \min_{x_j} \{ \bar{p}(x_j) \} \quad \text{при} \quad \sum_{j=1}^n x_j = N$$

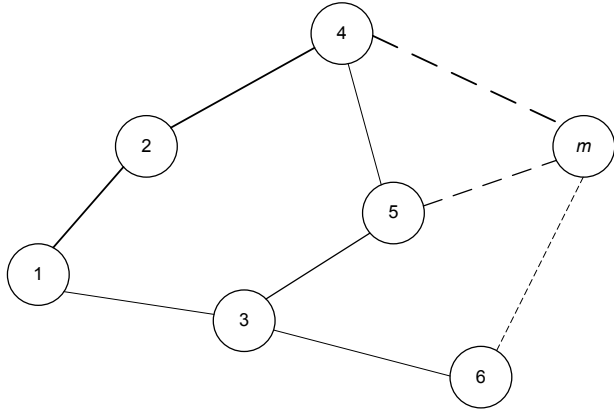
### 5. Выбор метода оптимизации целевой функции

### 6. Решение задачи

Эти этапы будут рассмотрены ниже.

## 4.2 Надежность сети связи

Рассматривается сеть связи. Структура сети задана графом. Сеть состоит из  $m$  узлов связи и  $n$  направлений связи (ребер графа).



$x_j$  Число каналов в  $j$  направлении связи

$w_j$  Надежность канала в  $j$  направлении связи  
(вероятность исправного состояния)

$v_j$  Значимость  $j$  направления связи

$$v_j = 0 \dots 1 \quad \sum_{j=1}^n v_j = 1$$

$N$  Общее число каналов в сети = *const*

### 1. Формулировка задачи

- предметная область - надежность сети связи (вероятность исправного состояния)
- состояние сети задано исходными данными
- требуется найти оптимальное число каналов в каждом из направлений связи для получения максимальной надежности сети, при заданном общем числе каналов  $N$ .

### 2. Построение модели системы

- полагаем, что направление исправно, если исправен хотя бы один канал;
- тогда модель направления связи может быть представлена как вероятность его исправного состояния

$$W_j = 1 - (1 - w_j)^{x_j}; \quad j = 1 \dots n$$

### 3. Выбор параметров управления и показателей состояния

-параметры управления: число каналов в каждом из направлений связи,

 $x_j$ 

-показатели состояния: вероятность исправного состояния каждого из направлений связи.

 $p_j$ 

### 4. Построение целевой функции

Выберем в качестве целевого показателя средневзвешенную вероятность исправного состояния сети

$$\bar{w} = \sum_{j=1}^n v_j \cdot W_j; \quad j = 1 \dots n \quad \text{где} \quad \sum_{j=1}^n v_j = 1$$

Целевая функция

$$x_j = \arg \max_{x_j} \{ \bar{w}(x_j) \} \quad \text{при} \quad \sum_{j=1}^n x_j = N$$

### 5. Выбор метода оптимизации целевой функции

### 6. Решение задачи

Эти этапы будут рассмотрены ниже.

## 5. Аналитические методы оптимизации

### Безусловная оптимизация

-необходимые и достаточные условия существования экстремума функции

### Условная оптимизация

-метод множителей Лагранжа  
-условия Каруша-Куна-Таккера

---

## 5.1 Экстремумы функции

Допустимый вектор  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  является **абсолютным (глобальным)** экстремумом  $f(x)$ , если  $f(x^*) \leq (\geq) f(x)$  для  $\forall x$

Допустимый вектор  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  является **локальным** экстремумом  $f(x)$ , если существует  $\delta > 0$  такое, что при всех  $0 \leq |\Delta x| \leq \delta$  справедливо

$$|f(x^* + \Delta x) - f(x^*)| \geq 0$$



## 5.2 Безусловная оптимизация

Пусть задана функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

### 1. Необходимые условия существования локального экстремума (НУ)

В точке  $x^*$  может иметь место экстремум тогда, когда она дифференцируема в данной точке и все частные производные функции в этой точке равны нулю.

$$\left. \frac{\partial f(x)}{\partial x_1} \right|_{x^*} = \left. \frac{\partial f(x)}{\partial x_1} \right|_{x^*} = \dots = \left. \frac{\partial f(x)}{\partial x_n} \right|_{x^*} = 0$$

Если эти условия выполняются, то точка  $x^*$  называется стационарной точкой.

### 2. Достаточные условия существования локального экстремума (ДУ)

Пусть для функции  $f(x)$  в точке  $x^*$  выполняются ДУ и она имеет вторые смешанные производные в этой точке. Когда второй дифференциал  $f(x)$

$$d^2 f(x) = \sum_{i=1}^n \sum_{k=1}^n \left. \frac{\partial^2 f(x)}{\partial x_i \partial x_k} \right|_{x^*} \Delta x_i \Delta x_k = \sum_{i=1}^n \sum_{k=1}^n a_{ik} \Delta x_i \Delta x_k$$

- есть отрицательно определенная квадратичная форма, то в точке  $x^*$  имеет место максимум,  
- есть положительно определенная квадратичная форма, то в точке  $x^*$  имеет место минимум

Квадратичная форма является положительно определенной, если все собственные значения матрицы  $a_{ik}$  положительны, и отрицательно определенной если все эти собственные значения отрицательны.

Если собственные значения отрицательны матрицы  $a_{ik}$  имеют разные знаки то экстремума в данной точке нет.

$$a_{ik} = \begin{vmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \dots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{vmatrix} \quad \text{Гессиан } f(x)$$

Собственные значения можно найти как корни уравнения

$$\begin{vmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} - \lambda & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} - \lambda & \dots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f(x)}{\partial x_n^2} - \lambda \end{vmatrix} = 0$$

## 5.3 Условная оптимизация

**Математическое программирование** - это область математики, разрабатывающая теорию, численные методы решения многомерных задач с ограничениями. В отличие от классической математики, математическое программирование занимается математическими методами решения задач нахождения наилучших вариантов из всех возможных.

### 5.3.1 Метод множителей Лагранжа

Пусть задана функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$  требуется найти экстремумы  $f(x)$  при заданных ограничениях:

$$\varphi_i(x) = \varphi_i(x_1, x_2, \dots, x_n) = 0; \quad i = 1 \dots m$$

#### 1. Необходимые условия существования локального экстремума (НУ)

**Функция Лагранжа**

$$\Phi(x, \psi) = f(x) + \sum_{i=1}^m \psi_i \varphi_i(x) \quad \psi_i \text{ множители Лагранжа}$$

Если  $x^*$  является локальным экстремумом, то существует вектор  $\psi^* = (\psi_1^*, \psi_2^*, \dots, \psi_m^*)$

$$\text{такой, что } \Phi'_x(x^*, \psi^*) = 0$$

$$\text{где } \Phi'_x(x, \psi) = f'(x) + \sum_{i=1}^m \psi_i \varphi'_i(x)$$

## 1. Достаточные условия существования локального экстремума (ДУ)

Если  $f(x)$  и  $\varphi_i(x)$  дважды дифференцируемы в точке  $x^*$

и в этой точке выполняются ДУ и условия

$$\left(\Phi''_{xx}(x^*, \psi^*)h, h\right) > 0 \quad \text{или} \quad \left(\Phi''_{xx}(x^*, \psi^*)h, h\right) < 0$$

При всех ненулевых  $h$ , удовлетворяющих условиям

$$\left(\varphi'_i(x^*)h, h\right) = 0 \quad i = 1 \dots m$$

То  $x^*$  строгий локальный минимум (максимум)  $f(x)$  при заданных ограничениях.

Где 
$$\Phi''_{xx}(x^*, \psi^*) = f''(x) + \sum_{i=1}^m \psi_i \varphi''_i(x)$$

## Пример

Дано  $f(x_1, x_2) = x_1^2 + x_2^2$ ;  $g_1(x_1, x_2) = x_1 + x_2 - 2 = 0$

Найти  $\text{extr}_{x_1, x_2}(f(x_1, x_2))$ ; при  $g_1(x_1, x_2) = 0$

## Решение

### 1. Необходимые условия

$$(\varphi'_i(x^*)h, h) = 0 \quad i = 1 \dots m$$

То  $x^*$  строгий локальный минимум (максимум)  $f(x)$  при заданных ограничениях.

$\varphi_i(x)$

Где 
$$\Phi''_{xx}(x^*, \psi^*) = f''(x) + \sum_{i=1}^m \lambda_i \varphi''_{xx}(x^*, \psi^*)h, h < 0$$

При всех ненулевых  $h$ , удовлетворяющих условиям  $(\Phi''_{xx}(x^*, \psi^*)h, h) > 0$

и в этой точке выполняются ДУ и условия

## 5.3.2 Выпуклые функции

**Выпуклое программирование** - раздел нелинейного программирования, совокупность методов решения нелинейных экстремальных задач с выпуклыми целевыми функциями и выпуклыми системами ограничений.

Функция выпукла, если для любых двух значений аргумента выполняется неравенство Йенсена

$$f(t \cdot x + (1-t) \cdot y) \leq t \cdot f(x) + (1-t) \cdot f(y)$$
$$t \in 0 \dots 1 \quad x \neq y \quad n$$

### Некоторые свойства выпуклой функции:

- дважды дифференцируемая функция выпукла тогда и только тогда, когда ее график лежит не ниже касательной, проведенной в любой точке интервала выпуклости;
- дважды дифференцируемая функция выпукла, если ее вторая производная не отрицательна;
- если  $f(x)$  и  $g(x)$  выпуклы, то их линейная комбинация тоже выпуклая функция;
- локальный минимум выпуклой функции является глобальным минимумом;
- любая стационарная точка выпуклой функции является глобальным экстремумом;

### 5.3.3 Условия Каруша-Куна-Таккера (ККТ)

**Выпуклое программирование** - раздел нелинейного программирования, совокупность методов решения нелинейных экстремальных задач с выпуклыми целевыми функциями и выпуклыми системами ограничений.

Пусть задана выпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$  требуется найти экстремумы  $f(x)$  при заданных ограничениях:

$$\varphi_j(x) = \varphi_j(x_1, x_2, \dots, x_n) \leq 0; \quad x \geq 0; \quad j = 1 \dots m$$

$\varphi_j(x)$  выпуклые функции

**Необходимые и достаточные условия существования локального экстремума**

**Функция Лагранжа**

$$\Phi(x, \psi) = f(x) + \sum_{j=1}^m \psi_j \varphi_j(x) \quad \psi_i \text{ множители Лагранжа}$$

1) $\frac{\partial \Phi(x, \psi)}{\partial x_i} \geq 0$	4) $\frac{\partial \Phi(x, \psi)}{\partial \psi_j} \geq 0$
2) $\frac{\partial \Phi(x, \psi)}{\partial x_i} x_i = 0$	5) $\frac{\partial \Phi(x, \psi)}{\partial \psi_j} x_i = 0$
3) $x_i \geq 0$ $i = 1 \dots n$	6) $\psi_j \geq 0$ $j = 1 \dots m$

## 6. Численные методы оптимизации

### Безусловная оптимизация

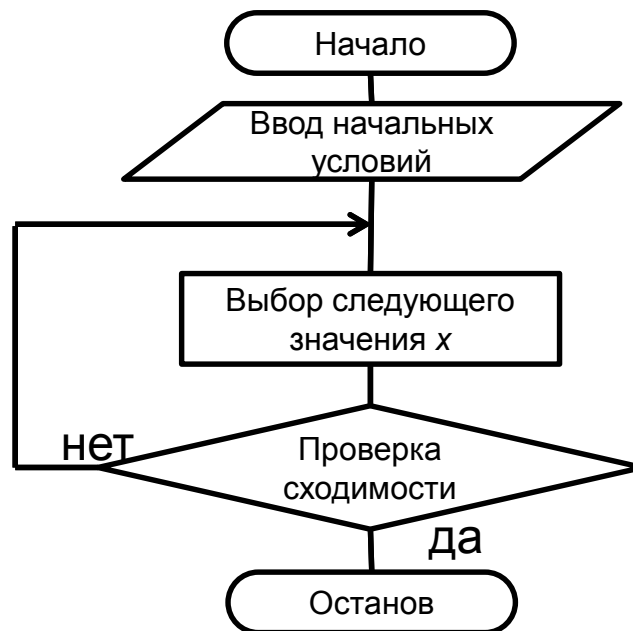
- функции одной переменной
- функции нескольких переменных

### Условная оптимизация

- функции нескольких переменных



## 6.1 Общий алгоритм численных методов



Численный метод представляет собой итерационную циклическую процедуру, на каждом цикле которой производится выбор нового значения переменной (согласно некоторому методу), вычисление значения оптимизируемой функции и проверка критерия сходимости. Циклы повторяются, пока не выполнен критерий сходимости.

## 6.2 Оптимизация функции одной переменной

### 6.2.1 Прямой поиск

Заданы

Функция  $n$  переменной

$$f(x) = x \in [a, b]$$

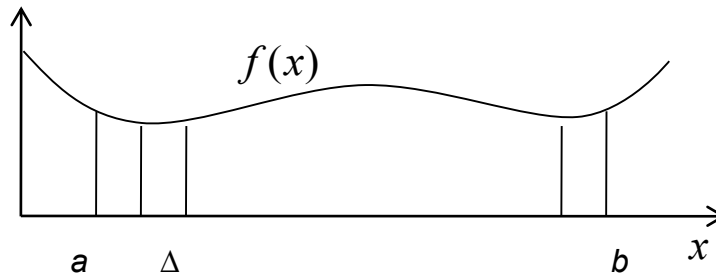
$\frac{\Delta}{2}$  – абсолютная погрешность

2. Поиск экстремума

Требуется найти экстремум

$$x^* = \arg \min_x \{f(x)\} \quad \text{или}$$

$$x^* = \arg \max_x \{f(x)\}$$



$$n = \frac{b-a}{\Delta}$$

$$x^* = \arg \min_{x_i} \{f(x_i)\}, \quad i = 1 \dots n$$

## 6.2.2 Дихотомия

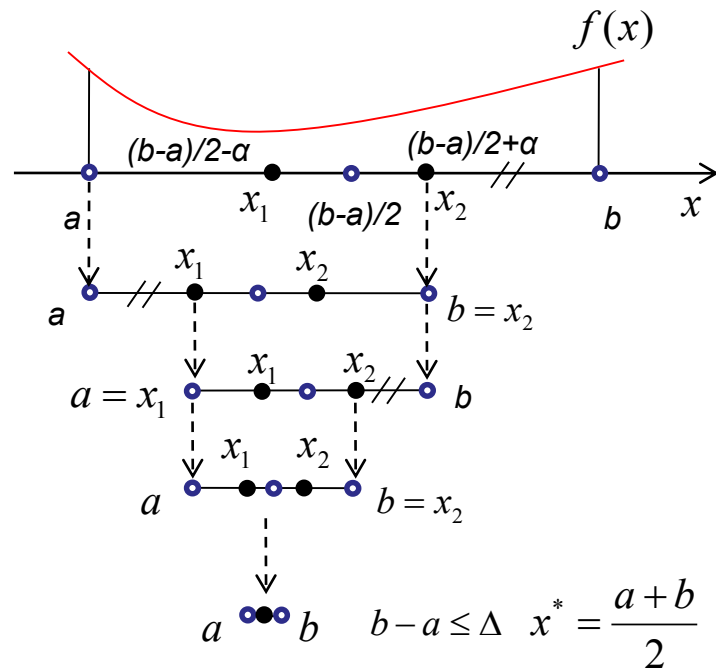
Заданы

Функция 1 переменной

$$f(x) = x \in [a, b]$$

$\frac{\Delta}{2}$  – абсолютная погрешность

2. Поиск экстремума



Требуется найти экстремум

$$x^* = \arg \min_x \{f(x)\} \quad \text{или}$$

$$x^* = \arg \max_x \{f(x)\}$$

- Алгоритм поиска представляет собой много итерационную процедуру
- На каждой итерации интервал поиска экстремума сужается путем исключения правого  $(x_2, b)$  или левого отрезка  $(a, x_1)$
- Правый отрезок исключается если  $f(x_2) > f(x_1)$
- Левый отрезок исключается если  $f(x_1) > f(x_2)$
- Итерации повторяются пока  $|b-a| > \Delta$
- результат  $x^* = (a+b)/2$

## 6.2.3 Метод золотого сечения

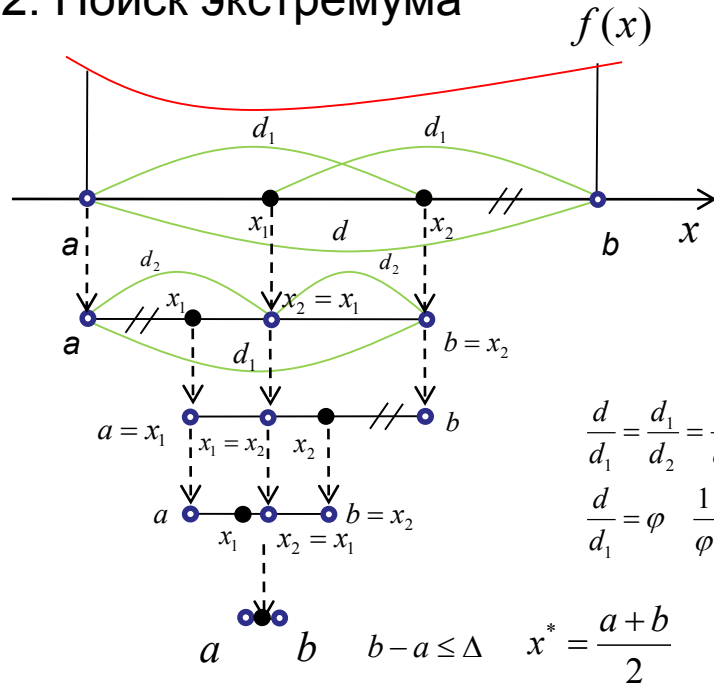
Заданы

Функция 1 переменной

$$f(x) = x \in [a, b]$$

$\frac{\Delta}{2}$  – абсолютная погрешность

2. Поиск экстремума



Требуется найти экстремум

$$x^* = \arg \min_x \{f(x)\} \quad \text{или}$$

$$x^* = \arg \max_x \{f(x)\}$$

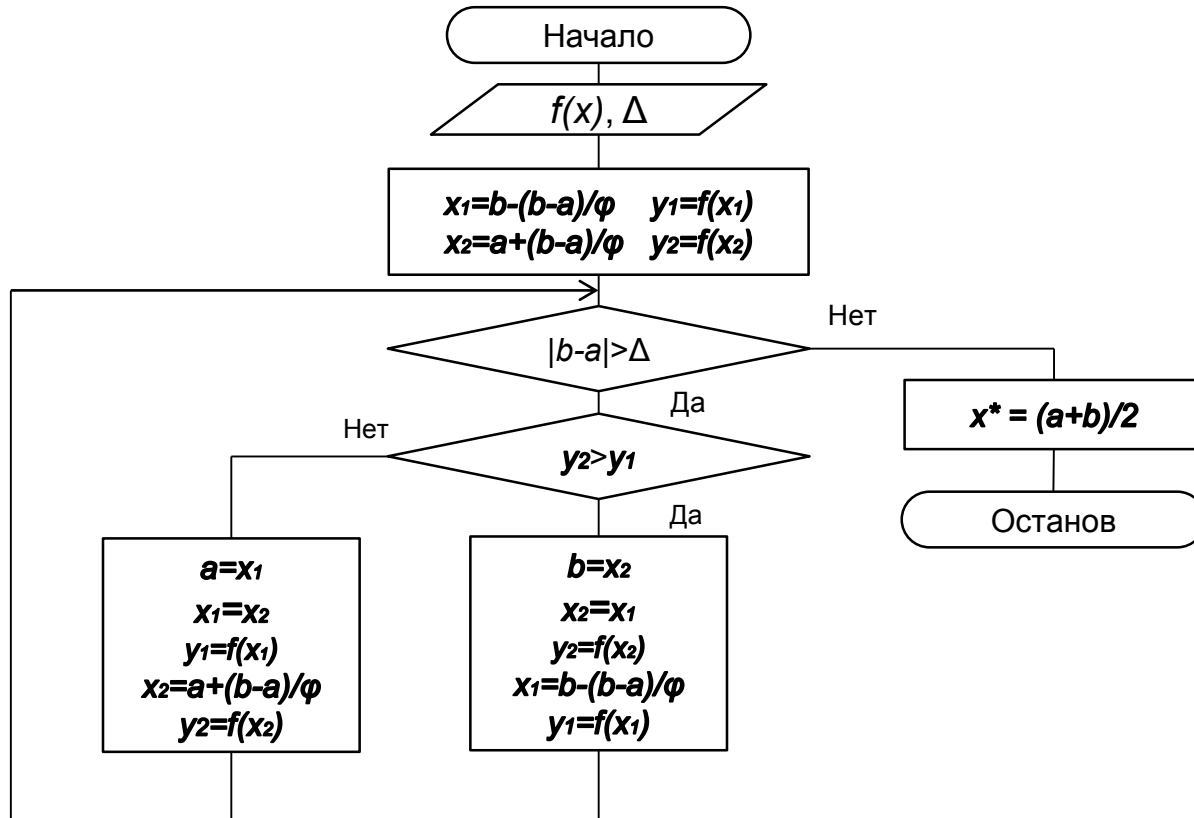
- Алгоритм поиска представляет собой много итерационную процедуру
- На каждой итерации интервал поиска экстремума сужается путем исключения правого ( $x_2, b$ ) или левого отрезка ( $a, x_1$ )
- Правый отрезок исключается если  $f(x_2) > f(x_1)$
- Левый отрезок исключается если  $f(x_1) > f(x_2)$
- Итерации повторяются пока  $|b-a| > \Delta$
- результат  $x^* = (a+b)/2$

$$\frac{d}{d_1} = \frac{d_1}{d_2} = \frac{d_1}{d-d_1}$$

$$\frac{d}{d_1} = \varphi \quad \frac{1}{\varphi} = \varphi - 1 \quad \varphi^2 - \varphi - 1 = 0 \Rightarrow \varphi = \frac{\sqrt{5}+1}{2}$$

$$a \quad b \quad b-a \leq \Delta \quad x^* = \frac{a+b}{2}$$

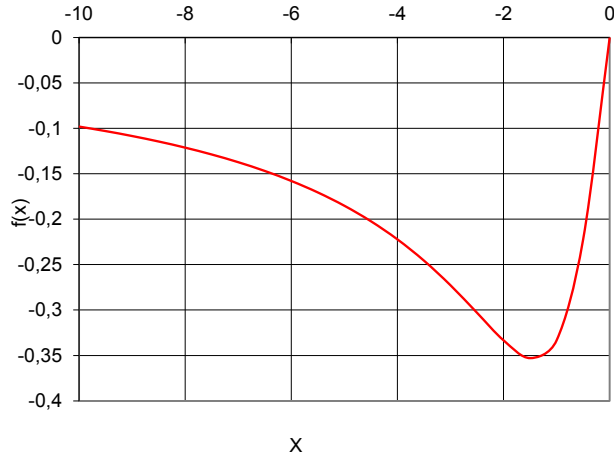
## Метод золотого сечения (алгоритм)



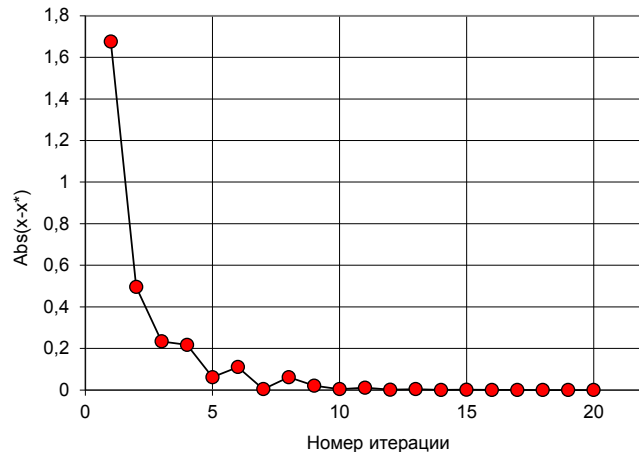
## Метод золотого сечения (пример)

Поиск минимума функции  $f(x) = \frac{x}{x^2 + 2}$

точность 0,001 на интервале [-10;0]



Приближение к экстремуму



Изменение интервала поиска по итерациям

N	Нижняя граница интервала	Верхняя граница интервала	Ширина интервала
0	-10,0000	0,0000	10,0000
1	-6,1803	0,0000	6,1803
2	-3,8197	0,0000	3,8197
3	-2,3607	0,0000	2,3607
4	-2,3607	-0,9017	1,4590
5	-1,8034	-0,9017	0,9017
6	-1,8034	-1,2461	0,5573
7	-1,5905	-1,2461	0,3444
8	-1,4590	-1,2461	0,2129
9	-1,4590	-1,3274	0,1316
10	-1,4590	-1,3777	0,0813
11	-1,4279	-1,3777	0,0502
12	-1,4279	-1,3969	0,0311
13	-1,4279	-1,4087	0,0192
14	-1,4206	-1,4087	0,0119
15	-1,4161	-1,4087	0,0073
16	-1,4161	-1,4115	0,0045
17	-1,4161	-1,4133	0,0028
18	-1,4150	-1,4133	0,0017
19	-1,4150	-1,4139	0,0011
20	-1,4146	-1,4139	0,0007

Результат  $x^* = -1,414$ ; (20 итераций)

## 6.2.4 Метод чисел Фибоначчи

Заданы

Функция 1 переменной

$$f(x) \quad x \in [a, b]$$

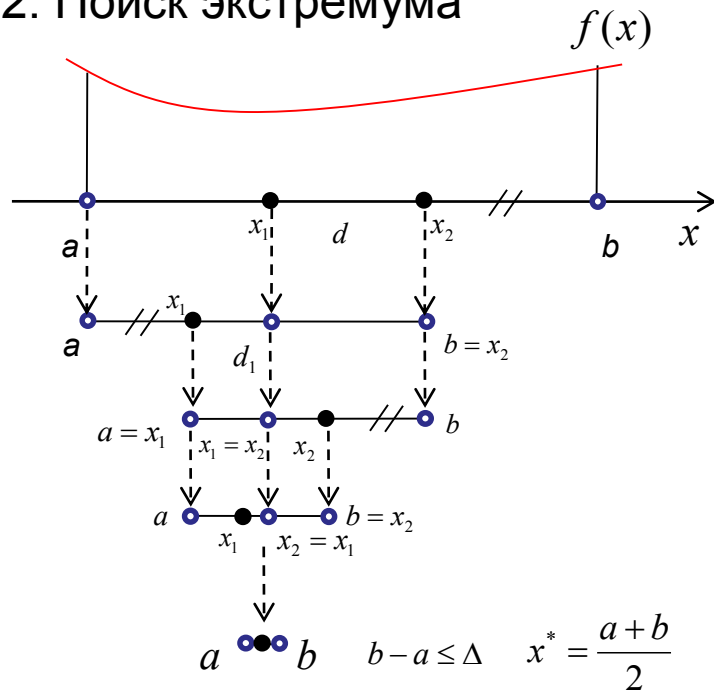
$n$  – число итераций

Требуется найти экстремум

$$x^* = \arg \min_x \{f(x)\} \quad \text{или}$$

$$x^* = \arg \max_x \{f(x)\}$$

### 2. Поиск экстремума

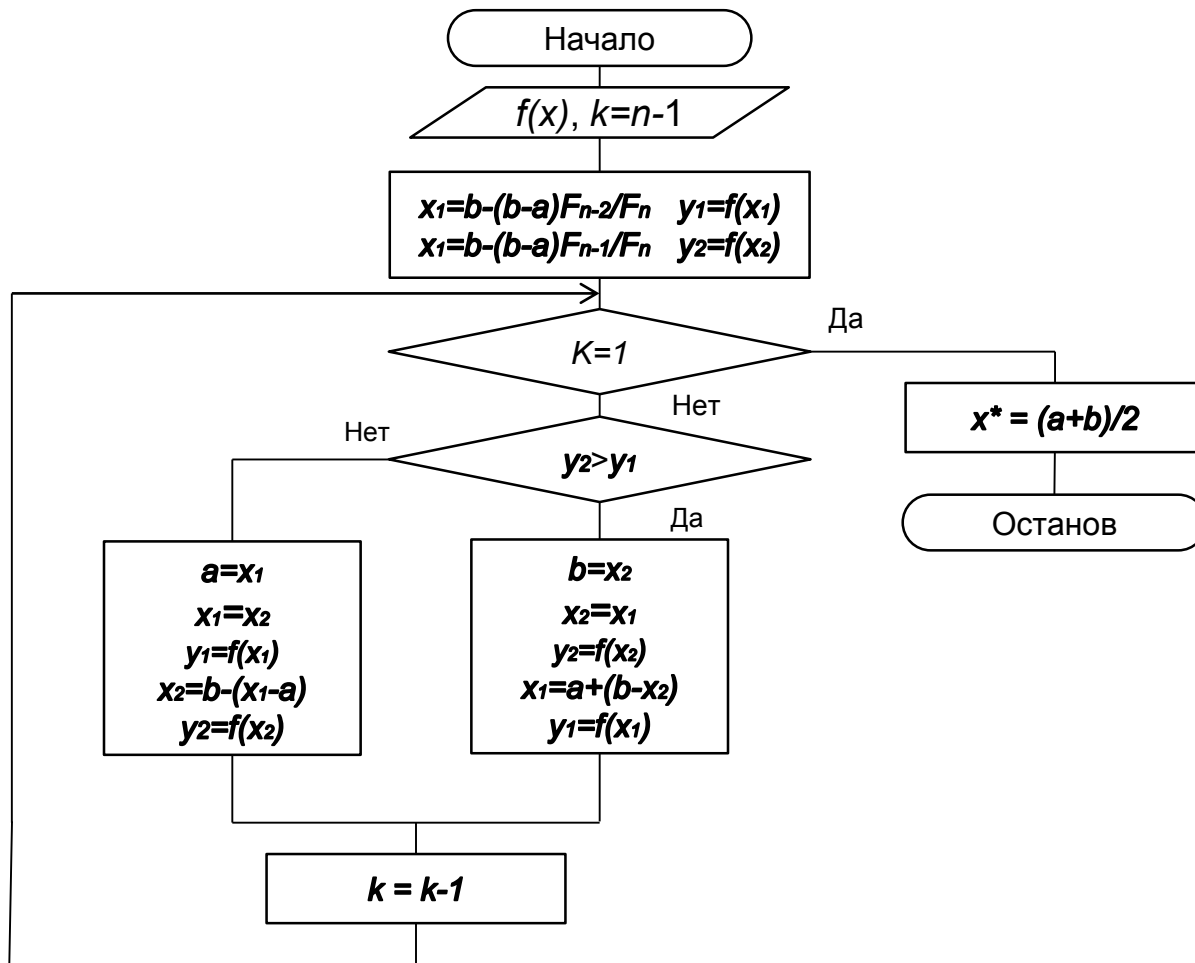


- Алгоритм поиска представляет собой много итерационную процедуру
- На каждой итерации интервал поиска экстремума сужается путем исключения правого ( $x_2, b$ ) или левого отрезка ( $a, x_1$ )
- Правый отрезок исключается если  $f(x_2) > f(x_1)$
- Левый отрезок исключается если  $f(x_1) > f(x_2)$
- Итерации повторяются пока не выполнено заданное число итераций
- результат  $x^* = (a+b)/2$

$$\lim_{n \rightarrow \infty} \frac{F_{n+1}}{F_n} = \varphi$$

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^n}{2\varphi - 1} \quad \text{формула Бине}$$

## Метод чисел Фибоначчи (алгоритм)





## 6.2.5 Метод квадратичной интерполяции

Заданы

Функция 1 переменной

$f(x)$   $h$  – величина шага

-Метод относится к группе методов аппроксимации

-Идея метода состоит в том, чтобы аппроксимировать исследуемую функцию квадратичной функцией вида

$$f(x) \rightarrow \tilde{f}(x) = a \cdot x^2 + b \cdot x + c$$

$$\frac{\partial \tilde{f}(x)}{\partial x} = 0 \Rightarrow \tilde{x}^* = -\frac{b}{2a}$$

Для которой можно легко вычислить точку экстремума (минимума) из условия равенства нулю производной

Коэффициенты  $a$ ,  $b$ ,  $c$  могут быть вычислены, если известны значения функции в 3 точках, например  $\alpha$ ,  $\beta$ ,  $\gamma$ .

$$f(\alpha) = f_\alpha; \quad f(\beta) = f_\beta; \quad f(\gamma) = f_\gamma$$

$$\begin{cases} a \cdot \alpha^2 + b \cdot \alpha + c = f_\alpha \\ a \cdot \beta^2 + b \cdot \beta + c = f_\beta \\ a \cdot \gamma^2 + b \cdot \gamma + c = f_\gamma \end{cases} \Rightarrow \begin{cases} a = [(\gamma - \beta)f_\alpha + (\alpha - \gamma)f_\beta + (\beta - \alpha)f_\gamma] / \Delta \\ b = [(\beta^2 - \gamma^2)f_\alpha + (\gamma^2 - \alpha^2)f_\beta + (\alpha^2 - \beta^2)f_\gamma] / \Delta \\ c = [\beta\gamma(\gamma - \beta)f_\alpha + \gamma\alpha(\alpha - \gamma)f_\beta + \alpha\beta(\beta - \alpha)f_\gamma] / \Delta \end{cases}$$

$$\Delta = (\alpha - \beta)(\beta - \gamma)(\gamma - \alpha)$$

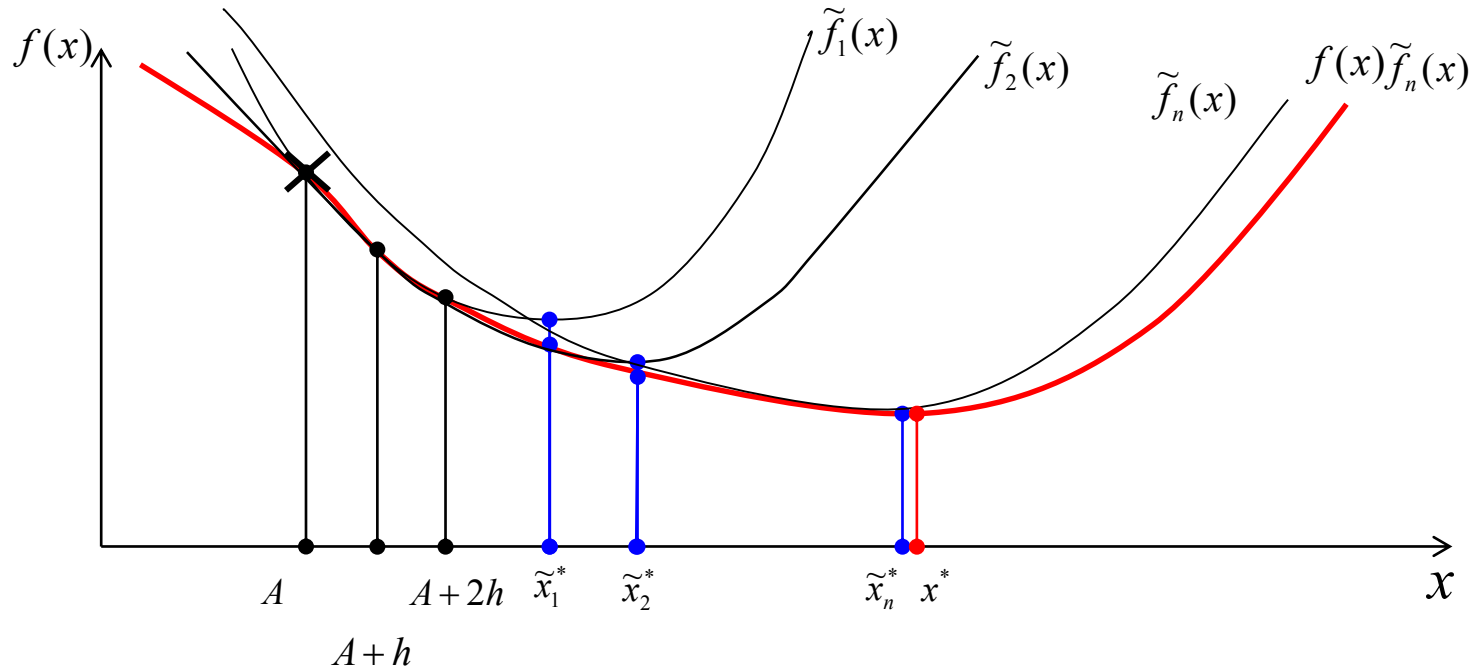
$$\tilde{x}^* = \frac{1}{2} \cdot \frac{(\beta^2 - \gamma^2)f_\alpha + (\gamma^2 - \alpha^2)f_\beta + (\alpha^2 - \beta^2)f_\gamma}{(\beta - \gamma)f_\alpha + (\gamma - \alpha)f_\beta + (\alpha - \beta)f_\gamma} \quad (*)$$

0) Перед первой итерацией выбирается стартовая точка  $x=A$ , в ней вычисляется значение функции  $f(A)$ .

Аргументу дается приращение  $h$  и вычисляется точка  $f(A+h)$ , если  $f(A+h) < f(A)$ , то вычисляется третья точка  $f(A+2h)$ , если  $f(A+h) > f(A)$ , то третья точка вычисляется, как  $f(A-h)$ .

# Метод квадратичной интерполяции

- 1) Для полученных 3 точек, согласно (\*) вычисляется значение экстремума аппроксимирующей функции.
- 2) Проверяется условие сходимости, например, по изменению значения функции, по отношению к ее значению на предыдущей итерации.
- 3) Если условие сходимости выполнено, то процесс останавливается и за результат  $x^*$  принимается,  $\tilde{x}^*$  полученное на последней итерации.
- 4) Если условие сходимости не выполнено, то из 3 точек отбрасывается «наихудшая», а вместо принимается значение  $\tilde{x}^*$  и производится переход к п.1.



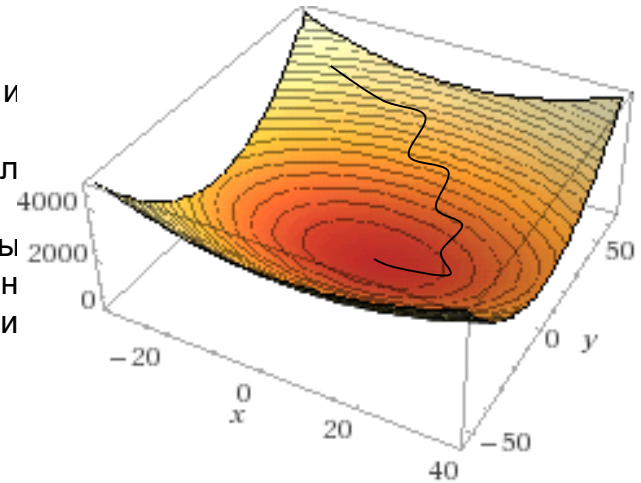
## 6.3 Оптимизация функция нескольких переменных

### 6.3.1 Покоординатный спуск

Пусть задана выпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

**Данный метод предполагает следующие шаги:**

1. Фиксируются все значения переменных кроме одной
2. Производится поиск экстремума по одной переменной одним и известными методов
3. Найденное значение присваивается данной переменной и дл поиска выбирается следующая переменная
4. После нахождения экстремумов по каждой из переменны проверяется критерий сходимости. Если условие сходимости н выполняется, то к 1, если выполняется, то полученные значени переменных и принимаются как искомый экстремум.



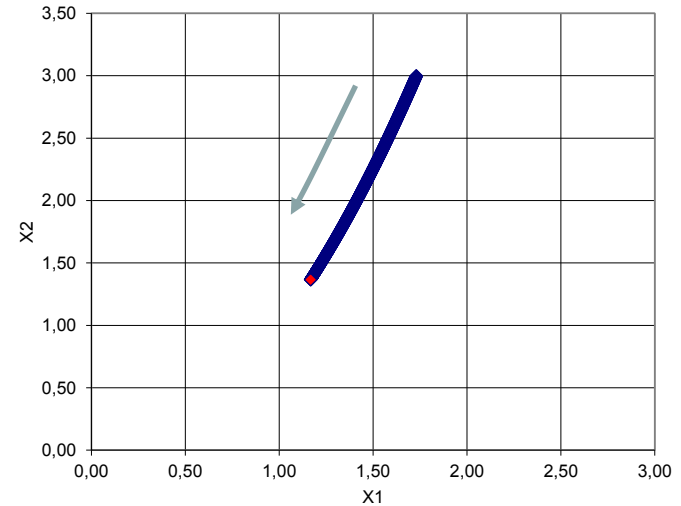
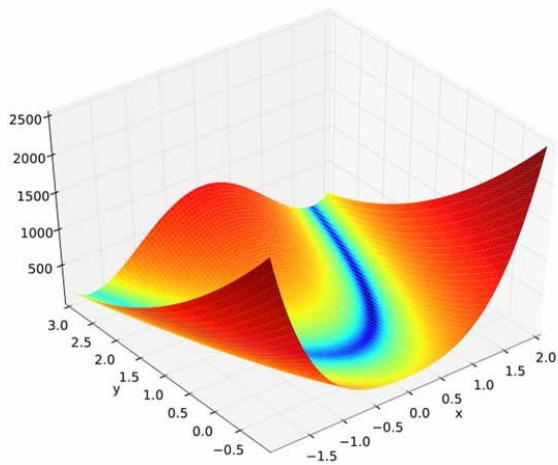
Данный метод может быть эффективен только при оптимизации относительно «простых» функций

## Покоординатный спуск (пример)

Задана выпуклая функция 2 переменных

$$f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

Шаги поиска экстремума  
(на каждом из шагов используется  
метод золотого сечения)



Результат  $x^*=(1,16; 1,36)$ ,  
функции

потребовалось около 1138х20 вычислений

## 6.3.2 Метод Хука-Дживса (поиск по образцу)

Пусть задана выпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

Данный метод включает в себя две процедуры:

-исследовательский поиск;

-поиск по образцу.

При инициализации Выбирается начальная точку  $B$  и величина шага  $h$

**Исследовательский поиск:**

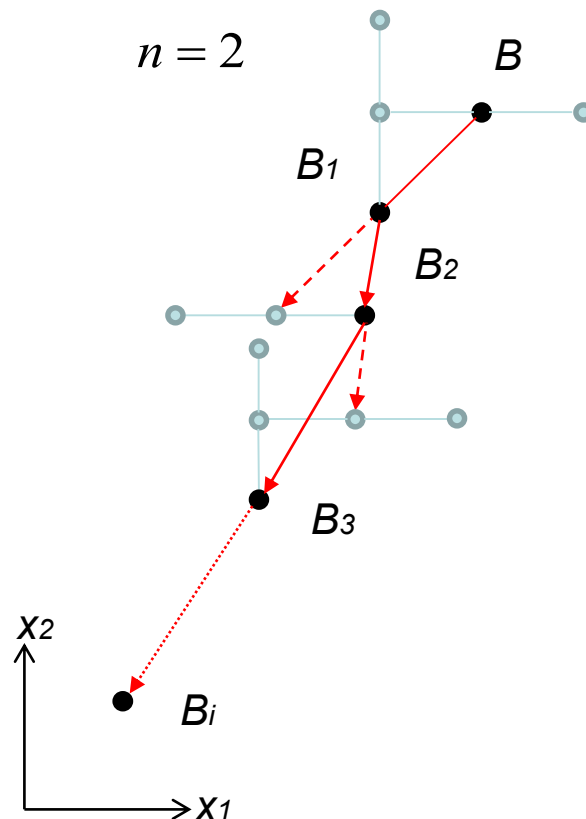
По числу переменных, поочередно, производятся пробные шаги из  $(.)$  на величину  $h$ , если шаг не приводит к уменьшению  $f(x)$ , то производится шаг  $(-h)$ , и т.д. по всем переменным.

В результате получаем базисную точку  $B_i$ .

**Поиск по образцу:**

После нахождения базисной точки  $B_i$  производится поиск по образцу, который заключается в том, что в направлении от  $B_{i-1}$  к  $B_i$  делается шаг из точки  $B_i$  величиной, например, равной расстоянию между  $B_{i-1}$  и  $B_i$ .

**Сходимость.** Если исследовательский поиск не приводит к получению новой базисной точки, то уменьшается величина шага  $h$  и процедура исследовательского поиска повторяется снова. Если величина шага становится меньше заданной величины  $h_0$ , то алгоритм завершает работу и возвращает значение последней найденной базисной точки.



## 6.3.2 Метод Хука-Дживса (поиск по образцу)

Пусть задана выпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

Данный метод включает в себя две процедуры:

-исследовательский поиск;

-поиск по образцу.

При инициализации Выбирается начальная точку  $B$  и величина шага  $h$

**Исследовательский поиск:**

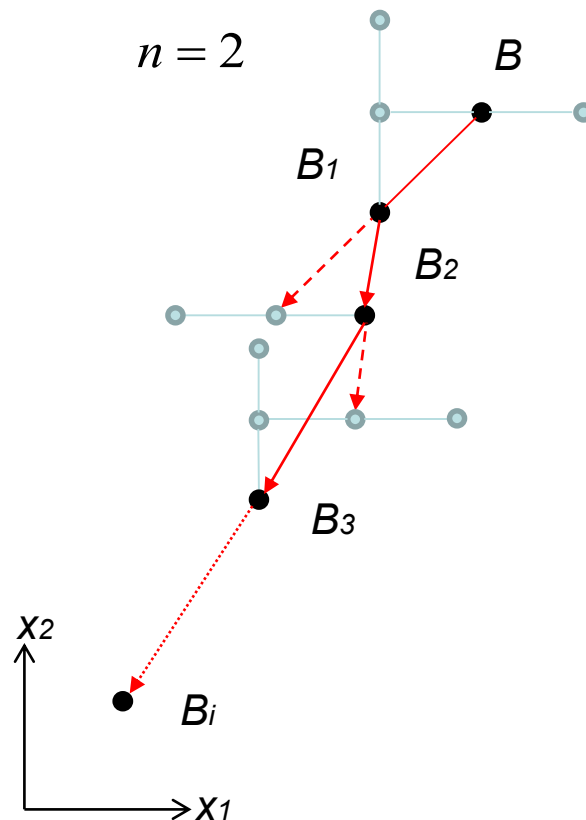
По числу переменных, поочередно, производятся пробные шаги из  $(.)$  на величину  $h$ , если шаг не приводит к уменьшению  $f(x)$ , то производится шаг  $(-h)$ , и т.д. по всем переменным.

В результате получаем базисную точку  $B_i$ .

**Поиск по образцу:**

После нахождения базисной точки  $B_i$  производится поиск по образцу, который заключается в том, что в направлении от  $B_{i-1}$  к  $B_i$  делается шаг из точки  $B_i$  величиной, например, равной расстоянию между  $B_{i-1}$  и  $B_i$ .

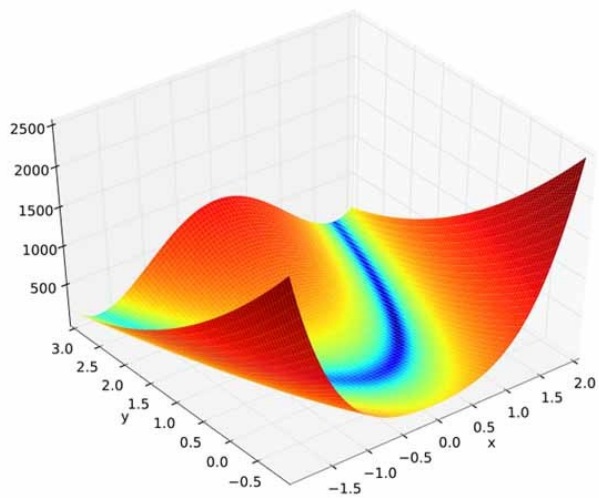
**Сходимость.** Если исследовательский поиск не приводит к получению новой базисной точки, то уменьшается величина шага  $h$  и процедура исследовательского поиска повторяется снова. Если величина шага становится меньше заданной величины  $h_0$ , то алгоритм завершает работу и возвращает значение последней найденной базисной точки.



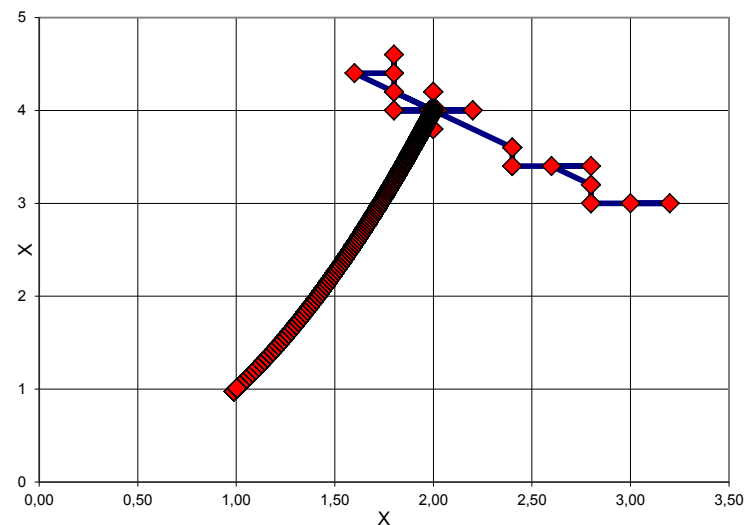
## Метод Хука-Дживса (пример)

Пример поиска минимума функции Розенброка

$$f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$



Шаги поиска экстремума из точки (3; 3)



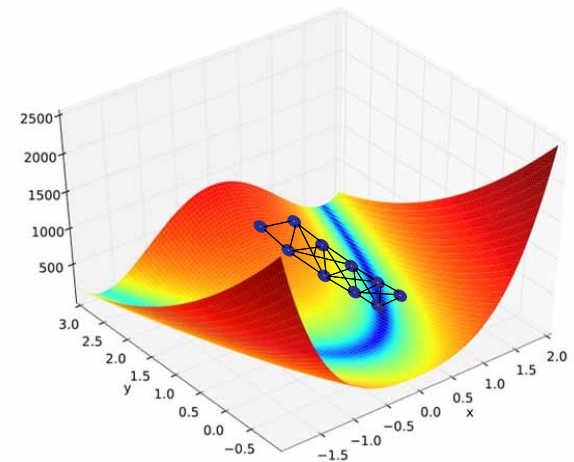
Результат (1,004; 1,008); 1261 вычисления функции

### 6.3.3 Симплекс метод Нелдера-Мида (поиск по деформируемому многограннику)

Пусть задана выпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

#### Этапы метод предполагает следующие шаги:

1. Подготовка. Задается исходный многогранник, содержащий  $n+1$  вершину (т.е.  $n+1$  точка - симплекс)
2. Оценка. Вычисляются значения функции в вершинах многогранника. Находят «худшую»  $H$ , «лучшую»  $L$ , и предшествующую «худшей» (по величине) вершину  $K$ .
3. Отражение. Относительно «наихудшей» точки определяется центр тяжести противоположной грани. Выполняется попытка отражения наихудшей точки через найденный центр тяжести. В результате чего получают точку  $R$ .
4. Растяжение. Если отраженная точка «лучше» «лучшей», то делается попытка растянуть многогранник в данном направлении. Если растяжение успешное, то полученная точка ( $E$ ) включается в список вершин многогранника, в противном случае в список вершин включается отраженная точка ( $R$ ).
5. Сокращение. Сокращение производится, если отраженная точка ( $R$ ) хуже точки  $K$ . Если  $R$  лучше  $H$ , то выполняется внешнее сокращение, в результате которого получают точку  $C1$ . Если  $R$  хуже  $H$  выполняется внутреннее сокращение, в результате которого получают точку  $C2$ . Если  $f(C1)$  или  $f(C2) < f(R)$ , то соответствующая точка включается в список вершин.
6. Перемещение. Включенная в многогранник точка замещает наихудшую точку ( $H$ ). Точка  $H$  исключается, этим достигается перемещение многогранника.
7. Сжатие. Сжатие многогранника производится во всех направлениях. При этом лучшая вершина ( $L$ ) остается на месте, а остальные пересчитываются.
8. Проверка сходимости. Вычисляется среднеквадратическое отклонение значений функции в вершинах многогранника. Если вычисленное значение меньше заданной величины, то сходимость достигнута.

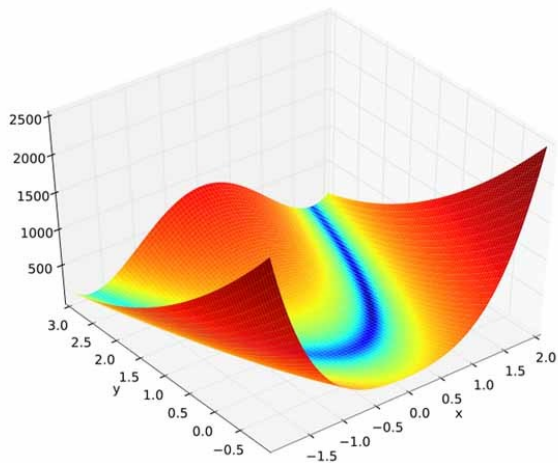




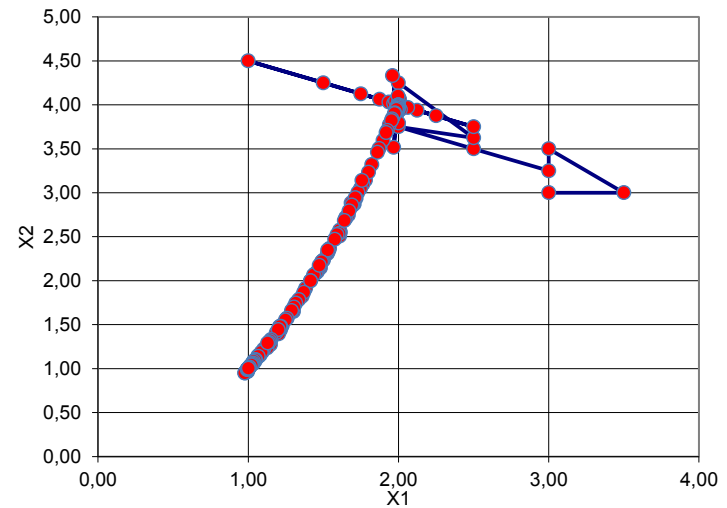
## Симплекс метод Нелдера-Мида (пример)

Пример поиска минимума функции Розенброка

$$f(x) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$



Шаги поиска экстремума из точки (3; 3)



Результат (1,000; 1,002); 243 вычисления функции

## 6.3.4 Комплексный метод Бокса (Условная оптимизация)

Пусть задана выпуклая функция нескольких переменных  $f(x) = f(x_1, x_2, \dots, x_n)$  и ограничения в виде явных и неявных ограничений  $n$

$$l_i \leq x_i \leq u_i \quad i = 1 \dots n \quad g_j(x) \leq b_j \quad j = 1 \dots m$$

(Данный метод представляет собой модификацию симплекс метода Нелдера-Мида, в отличие от последнего, поиск ведется с помощью многогранника, содержащего  $2n$  вершин (комплекса))

### Метод предполагает следующие шаги:

1. **Подготовка.** Задается стартовая точка  $x_0$ , удовлетворяющая всем условиям ограничений. Остальные  $2n-1$  точек могут быть получены, например «случайным» выбором, с учетом явных ограничений  $x_i = l_i + rnd * (u_i - l_i)$ , где  $rnd$  – случайное число от 0 до 1. После этого полученные точки проверяются на их соответствие неявным ограничениям, если точка  $x_i$  не удовлетворяет неявным ограничениям, то она «подтягивается» к центру масс точек, удовлетворяющих всем ограничениям, т.е. пересчитывается, например как  $x_i = (x_i + x_c) / 2$ , где  $x_c$  – центр масс.

$$x_c = \frac{1}{i-1} \sum_{r=1}^i x_r$$

2. **Оценка.** Вычисляются значения функции в вершинах многогранника. Находят «худшую» вершину  $x_n$ .
3. **Отражение.** Относительно «наихудшей» точки определяется центр тяжести противоположной грани многогранника

$$x_0 = \frac{1}{2n-1} \sum_{i=1}^{2n-1} x_i$$

4. Выполняется попытка отражения наихудшей точки через найденный центр тяжести. В результате чего получают точку  $x_R$ .
5. **Проверка на допустимость точки  $x_R$ .** Если отраженная точка удовлетворяет явным и неявным ограничениям, то она включается в комплекс, а «наихудшая» точка  $x_n$  исключается. Если  $x_R$  не удовлетворяет явным ограничениям, то  $x_R$  присваивается ближайшее допустимое значение. После этого производится проверка на соответствие неявным ограничениям. Если условия неявных ограничений не выполняются, то точка  $x_R$  «подтягивается» к центру масс  $x_R = (x_i + x_0) / 2$ , и вновь выполняется проверка. При каждом «подтягивании» точки к центру масс проверяется критерий сходимости, в качестве которого обычно принимается величина среднеквадратического отклонения значения функции в точках комплекса и максимальное расстояние между точками комплекса  $dm$ .

$$\sigma = \sqrt{\frac{\sum_{i=1}^{2n-1} (f(x_i) - \bar{f})^2}{2n}}; \quad \bar{f} = \frac{1}{2n} \sum_{i=1}^{2n} f(x_i)$$

6. Условие сходимости достигается, если  $\sigma$  и  $dm$  меньше некоторых заданных величин.

## 6.3.5 Метод штрафных функций (Условная оптимизация)

Пусть задана выпуклая функция  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$  и ограничения в виде  $C_j(x) > 0 \quad j = 1 \dots m$

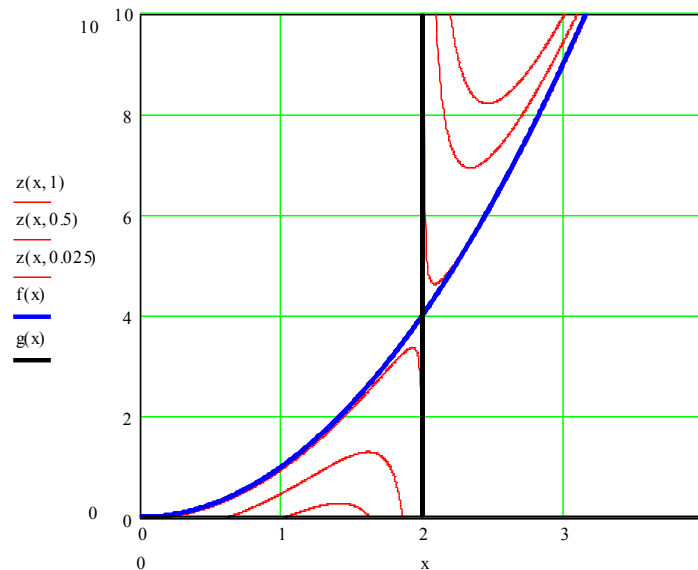
Метод заключается в замене исследуемой функции некоторой модифицированной функцией, которая в области допустимых значений близка к исходной функции, а вблизи области ограничений ее значение резко увеличивается.

$z(x) = f(x) + P(x); \quad P(x) - \text{штрафная функция}$

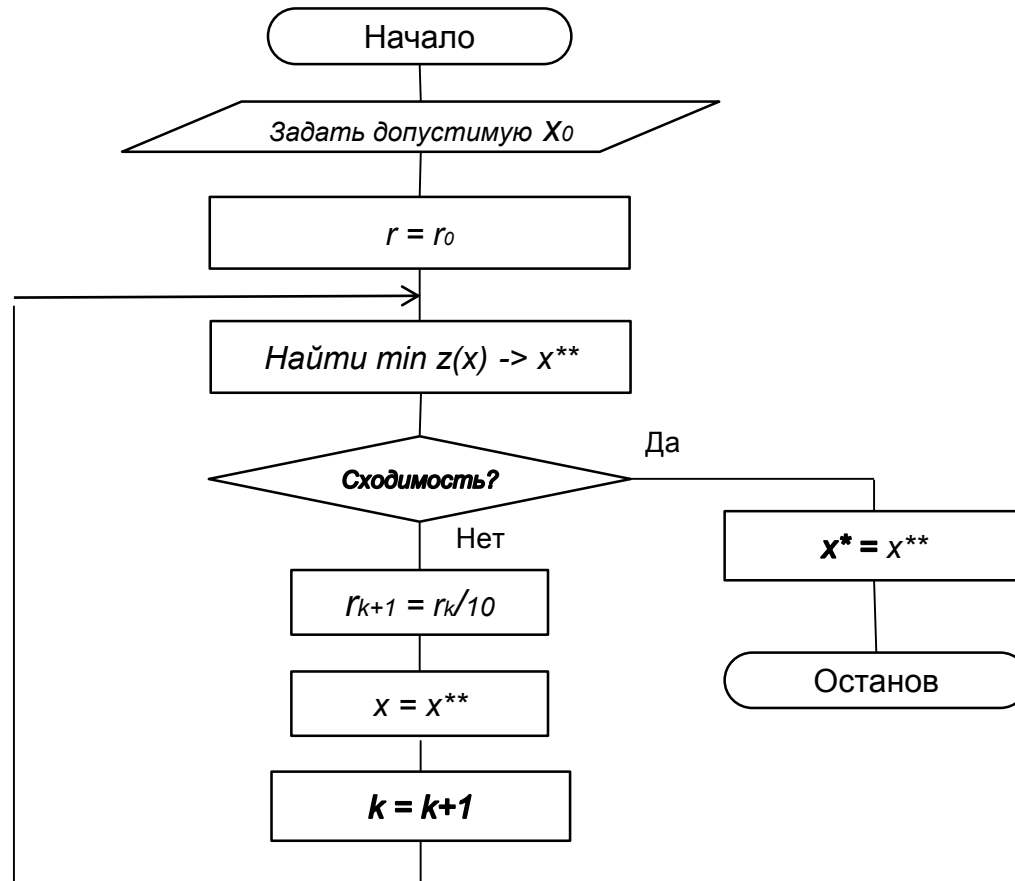
Например,  $P(x) = r \sum_{j=1}^m \frac{1}{C_j(x)}$ ;  $r > 0 \Rightarrow z(x, r) = f(x) + r \sum_{j=1}^m \frac{1}{C_j(x)}$ ;

Пример.

$$f(x) = x^2; \quad x > 2 \quad z(x, r) = x^2 + r \frac{1}{x-2};$$



## Метод штрафных функций (алгоритм)



---

## 6.3.4 Некоторые другие методы оптимизации выпуклых функций

Существует множество методов оптимизации функций их различных модификаций, в качестве примера можно привести следующие методы.

### **Безусловная оптимизация (Ф1П):**

-метод касательных, метод Ньютона, метод квадратичной, кубической интерполяции, (аппроксимационные методы)

### **Безусловная оптимизация (ФНП):**

-метод Хука-Дживса (поиск по образцу),

-градиентные методы (метод наискорейшего спуска, сопряженных градиентов ...)

### **Условная оптимизация (ФНП):**

-комплексный метод Бокса;

-метод штрафных функций

....

**Многие методы реализованы в математических и прикладных пакетах программ, таких как: Mathcad, Matlab, MS Excel и др.**

## 6.4 Стохастические методы

Целью разработки и использования данных методов является оптимизация невыпуклых функций. Общего метода численной оптимизации невыпуклых функций не существует. Существуют лишь подходы, позволяющие на основании имеющихся сведений об оптимизируемой функции использовать те или иные методы, позволяющие с достаточной уверенностью находить оптимальные (или приемлемые) решения.

Пусть задана невыпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

Оптимизацию невыпуклых функций называют многоэкстремальной оптимизацией.

### Примеры методов оптимизации:

-слепой случайный поиск

(поиск экстремума в заданной области путем проб в случайных точках области);

-локальный случайный поиск

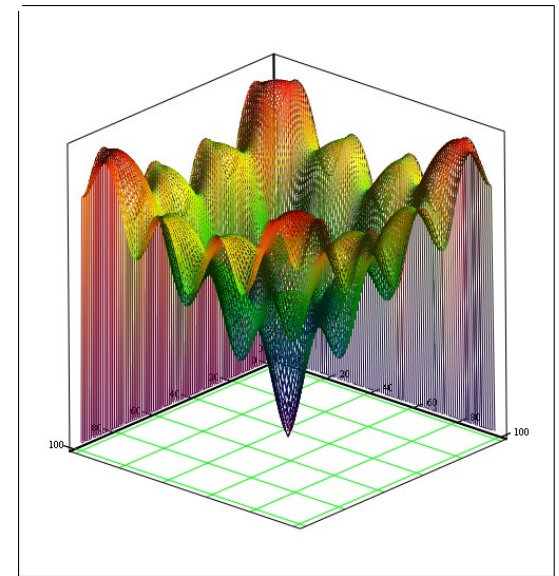
(поиск экстремума в заданной области путем проб в случайных точках, сосредоточенных вокруг некоторой базисной точки с последующим переходом к другой базисной точке);

-мультистартовый метод

(многократный запуск метода поиска экстремума выпуклой функции из различных стартовых точек);

-эволюционный метод (генетический алгоритм)

(метод имитирующий процесс эволюции, происходящий в живой природе).



F

Пример: функция Эккли

## 6.4.1 Слепой случайный поиск

Пусть задана невыпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

### слепой случайный поиск

В заданной области поиска экстремума  $[a, b]$  по каждой из переменных генерируется множество случайных чисел, равномерно распределенных на интервале  $[a, b]$ . В каждой из полученных точек вычисляется значение функции и выбирается наименьшее (при минимизации функции).

Если  $\Delta$  - заданное значение абсолютной погрешности

Относительная погрешность  $\varepsilon = \frac{\Delta}{b-a}$

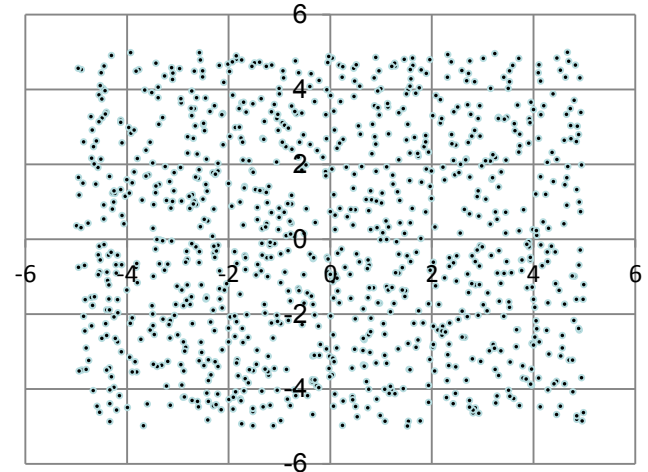
Вероятность того, что в серии из  $N$  испытаний (точек), хотя бы одно попадет в область минимума равна

$$p = 1 - (1 - \varepsilon^n)^N$$

Тогда необходимое число испытаний  $N$  равно

$$N = \frac{\ln(1-p)}{\ln(1-\varepsilon^n)}$$

При  $p$  близкой к единице данный метод требует большого числа испытаний, что приводит к его редкому применению в «чистом» виде.



Пример: Слепой случайный поиск,  $n=2$

## 6.4.2 Эволюционный метод (генетический алгоритм)

Пусть задана невыпуклая функция нескольких  $n$  переменных  $f(x) = f(x_1, x_2, \dots, x_n)$

### Генетический алгоритм

Предполагает имитацию процесса эволюции (различные реализации могут использовать различные приемы имитирующие этапы эволюционного процесса)

Например:

Допущения и ассоциации,

-значения переменных (точка) – особь

-значение функции в точке характеризует приспособленность особи (выживаемость)

Шаг 1. Создание начальной популяции – множества особей, численность  $p$  ;

Шаг 2. Выбор «родителей». (полагаем, что для появления новой особи требуются наследственные признаки двух других особей). Выбор может производиться различными способами, например, случайно (панмиксия) (Напрмер особи  $k$  и  $m$ ).

Шаг 3. **Рекомбинация**. Передача наследственных признаков потомку. Существуют различные способы реализации данного шага. Например, вектор переменных можно рассматривать как набор генов, при этом одноименные гены родителей объединяются со случайными коэффициентами:

$$x_i = \xi \cdot x_i^{(k)} + (1 - \xi) \cdot x_i^{(m)} \quad \text{где} \quad \xi = 0 \dots 1 \quad \text{случайное число}$$

Шаг 4. **Мутация**. Случайное изменение наследственных признаков. Например:  $x_i = \zeta \cdot x_i$

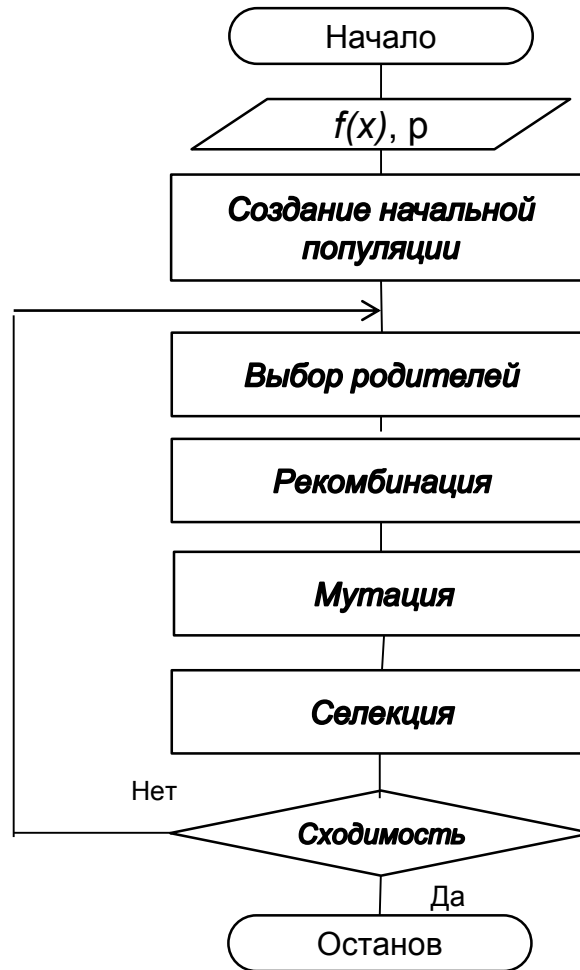
где  $\zeta$  случайный коэффициент

Шаг 5. **Селекция**. Новая особь помещается в популяцию, на место наименее приспособленной особи, которая освобождая позицию («погибает»).

**Сходимость**. Процесс смены поколений повторяется пока на протяжении некоторого (заданного) числа поколений не будет происходить уменьшение (увеличение) значения целевой функции.



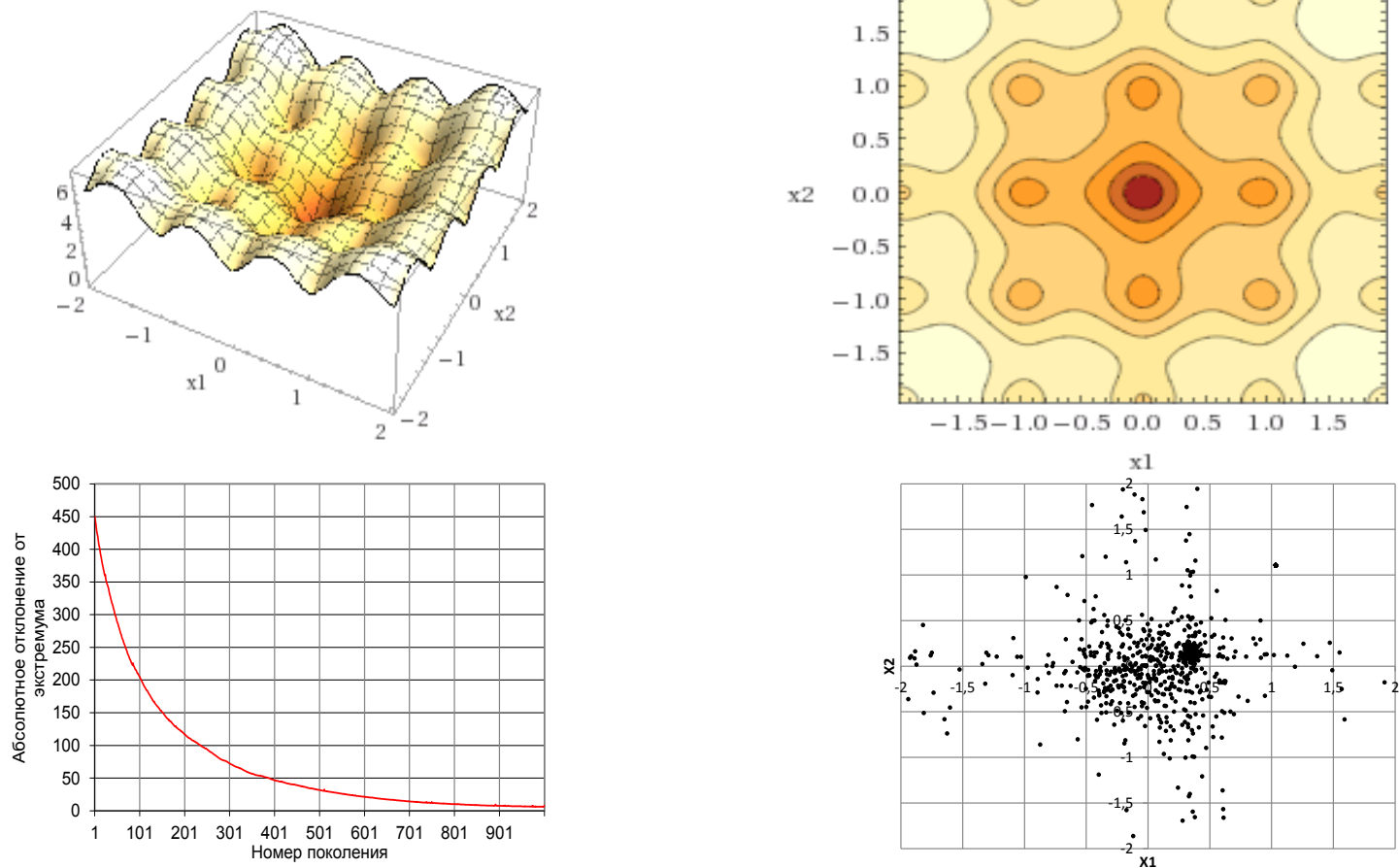
## Генетический алгоритм



**Сходимость.** Процесс смены поколений повторяется пока на протяжении некоторого (заданного) числа поколений не будет происходить уменьшение (увеличение) значения целевой функции.

## Генетический алгоритм (пример)

Пример. Поиск минимума функции Экли (для двух переменных).



Найденное значение:  $x^* = (-0,002; -0,014)$ ,  $f(x^*) = 0,040$  (размер популяции 50, число поколений 1000)

---

## 6.5 Динамическое программирование

Динамическое программирование - процесс нахождения решения задачи определенного типа, когда ответ на одну задачу может быть получен после решения предшествующей задачи.

**Основные принципы динамического программирования:**

**-многошаговость (разбиение задачи на подзадачи);**

**-оптимальность (оптимальное поведение по отношению к предшествующим решениям);**

Оптимальное поведение обладает тем свойством, что каковы бы ни были первоначальное состояние и решение в начальный момент, последующие решения должны составлять оптимальное поведение относительно состояния, получившегося в результате первого решения.

Беллман

Основное функциональное уравнение динамического программирования. (уравнение Беллмана)

$$f(x) = \max_q \{H(p, q, f(T(p, q)))\}$$

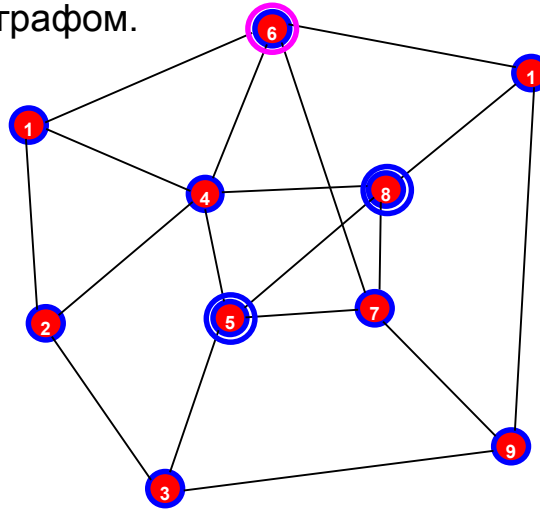
$p$  – вектор состояния

$q$  – вектор переменных

Классический пример задачи динамического программирования – поиск кратчайших путей в графе.

## Динамическое программирование (пример) размещение узла в сети связи

Пусть имеется некоторая структура, заданная, например, расположением населенных пунктов и связывающих их дорог, или зданий и сооружений кабельной канализации между ними, модель которой может быть описана графом.



**Задача 1** состоит в том, что требуется найти такую вершину графа, от которой расстояния до других вершин минимальны, например, когда требуется обеспечить минимальную длину абонентской линии (например, при выборе места установки концентратора ADSL).

**Задача 2** состоит в том, что требуется найти такую вершину графа, от которой сумма расстояний до других вершин минимальна, например, когда требуется обеспечить минимальный расход кабеля абонентской сети.

Задача 1 – (минимаксная задача) может быть решена путем поиска ЦЕНТРОВ графа.

Задача 2 – (минисуммная задача) может быть решена путем поиска МЕДИАН графа.

## 14.5 Структура с наименьшей протяженностью линий (задача поиска кратчайшего остова (SST) графа)

Рассмотрим взвешенный связный неориентированный граф  $G = (X, A)$ ; вес ребра  $(x_i, x_j)$  обозначим  $c_{ij}$ .

Из большого числа остовов графа нужно найти один, у которого сумма весов ребер наименьшая.

### Алгоритм Краскала (Крускала Kruskal's algorithm)

Шаг 1. Начать с вполне несвязного графа  $T$ , содержащего  $n$  вершин.

Шаг 2. Упорядочить ребра графа  $G$  в порядке неубывания их весов.

Шаг 3. Начав с первого ребра в этом списке, добавлять ребра в графе  $T$ , соблюдая условие: такое добавление не должно приводить к появлению цикла в  $T$ .

Шаг 4. Повторять шаг 3 до тех пор, пока число ребер в  $T$  не станет равным  $n - 1$ .

Получившееся дерево является SST графа  $G$ .

### Алгоритм Прима (Prim's Algorithm)

Этот алгоритм порождает SST посредством разрастания только одного поддерева, например  $T_s$ , содержащего больше одной вершины.

«Одиночные» вершины рассматриваются как отдельные поддеревья.

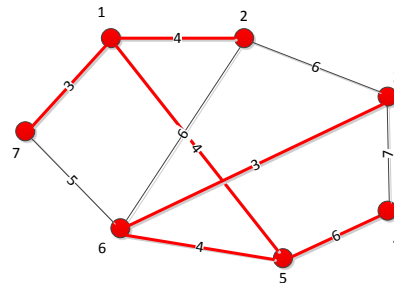
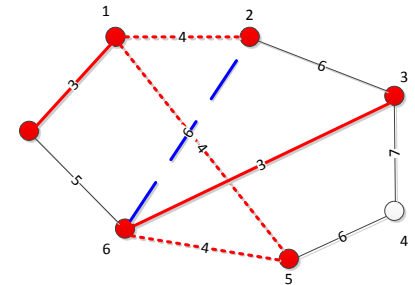
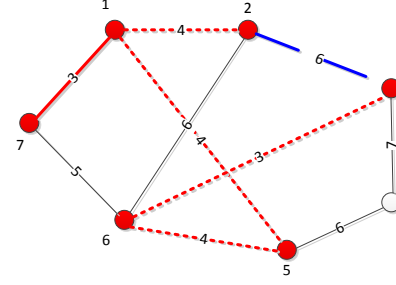
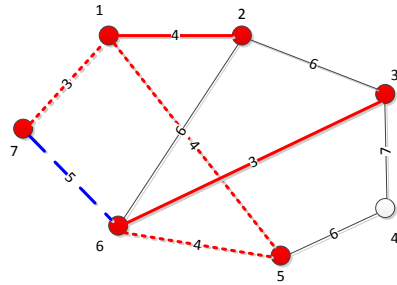
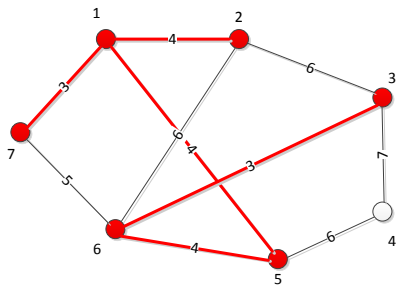
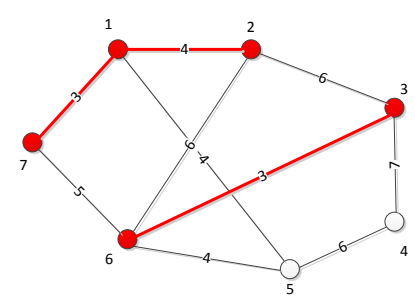
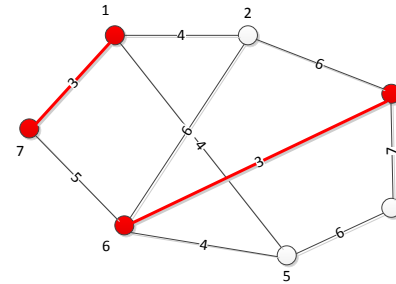
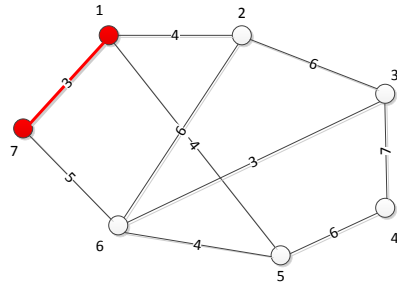
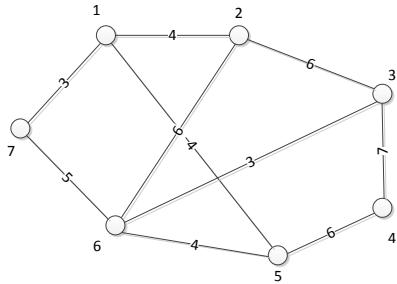
Поддерево  $T_s$  постепенно разрастается за счет присоединения ребер  $(x_i, x_j)$ ,

где  $x_i$  принадлежит  $T_s$  и  $x_j$  не принадлежит  $T_s$ ; причем добавляемое ребро должно иметь наименьший вес  $c_{ij}$ .

Процесс продолжается до тех пор, пока число ребер в  $T_s$  не станет равным  $n - 1$ . Тогда поддерево  $T_s$  будет требуемым SST.

## 14.5 Пример алгоритма Краскала

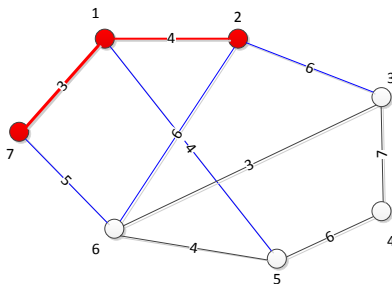
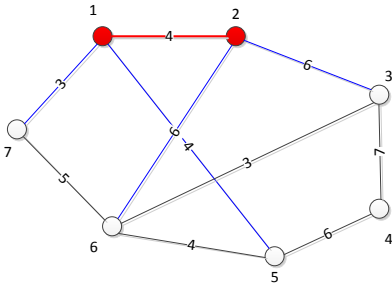
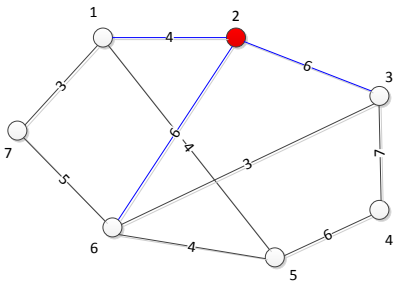
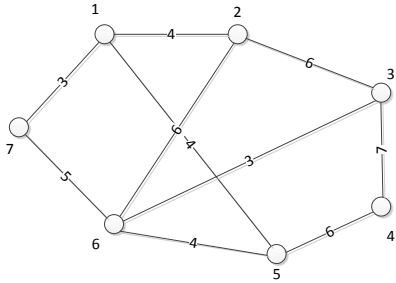
$(1,7)=3$ ;  $(3,6)=3$ ;  $(1,2)=4$ ;  $(1,5)=4$ ;  $(5,6)=4$ ;  $(6;7)=5$ ;  $(2,3)=6$ ;  $(2,6)=6$ ;  $(4,5)=6$ ;  $(3,4)=7$



### 14.5 Пример алгоритма Прима

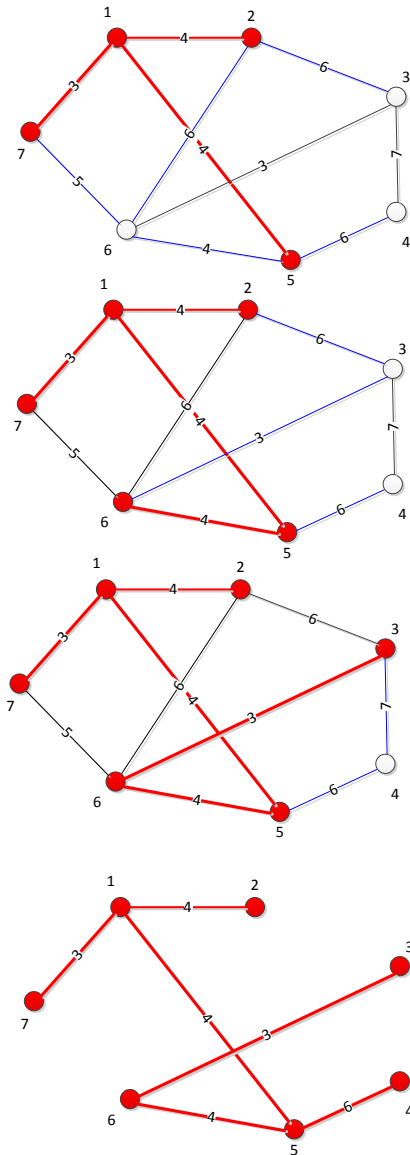
$$T_s = \{2\}$$

$$A_s = \{\emptyset\}$$



$T_s$ вершины	$T_s$ ребра	$A_s$	Пометки вершин
2	(1,2)=4 (6,2)=6 (3,2)=6	(1,2)	1 [2, 4] 3 [2, 6] 4 [0, ∞] 5 [0, ∞] 6 [2, 6] 7 [0, ∞]
1,2	(7,1)=3 (5,1)=4 (6,2)=6 (3,2)=6	(1,2) (7,1)	3 [2, 6] 4 [0, ∞] 5 [1, 4] 6 [2, 6] 7 [1, 3]
1,2,7	(6,7)=5 (5,1)=4 (6,2)=6 (3,2)=6	(1,2) (7,1) (5,1)	3 [2, 6] 4 [0, ∞] 5 [1, 4] 6 [7, 5]

### 14.5 Пример алгоритма Прима (продолжение)



$T_s$	Ребра	$A_s$	Пометки вершин
1,2,5,7	(6,7)=5 (6,2)=6 <b>(6,5)=4</b> (4,5)=6 (3,2)=6	(1,2) (7,1) (5,1) <b>(6,5)</b>	3 [2, 6] 4 [0, $\infty$ ] 6 [5, 4]
1,2,5,6,7	<b>(3,6)=3</b> (3,2)=6 (4,5)=6	(1,2) (7,1) (5,1) (6,5) <b>(3,6)</b>	3 [6, 3] 4 [5, 6]
1,2,3,5,6,7	(4,3)=7 <b>(4,5)=6</b>	(1,2) (7,1) (5,1) (6,5) (3,6) <b>(4,5)</b>	4 [4, 5]
1,2,3,4,5,6,7		(1,2) (7,1) (5,1) (6,5) (3,6) (4,5)	



---

## Алгоритм Дейкстры (Dijkstra's algorithm) (поиск кратчайших путей от заданной вершины)

Метод основан на приписывании вершинам временных пометок, причем пометка вершины дает верхнюю границу длины пути от  $s$  к этой вершине. Эти пометки (их величины) постепенно уменьшаются с помощью итерационной процедуры, и на каждом шаге точно одна из временных пометок становится постоянной. Последнее указывает на то, что пометка уже не является верхней границей, а дает точную длину кратчайшего пути от  $s$  к рассматриваемой вершине.

Пусть  $L(x_i)$  — пометка вершины  $x_i$ .

*Присвоение начальных значений*

Шаг 1. Положить  $L(s) = 0$  и считать эту пометку постоянной.

Положить  $L(x_i) = \infty$  для всех  $x_i \neq s$  и считать эти пометки временными.

Положить  $p = s$ .

*Обновление пометок*

Шаг 2. Для всех  $x_i \in \Gamma(p)$ , пометки которых временные, изменить пометки в соответствии со следующим выражением:

$$L(x_i) = \min [L(x_i), L(p) + c(p, x_i)]$$

*Превращение пометки в постоянную*

Шаг 3. Среди всех вершин с временными пометками найти такую, для которой

$$L(x_i^*) = \min [L(x_i)].$$

Шаг 4. Считать пометку вершины  $x_i^*$  постоянной и положить  $p = x_i^*$ .

Шаг 5.

(а) (Если надо найти лишь путь от  $s$  к  $t$ .) Если  $p = t$ , то  $L(p)$  является длиной кратчайшего пути.

*Останов.*

Если  $p \neq t$ , перейти к шагу 2.

(б) (Если требуется найти пути от  $s$  ко всем остальным вершинам.)

Если все вершины отмечены как постоянные, то эти пометки дают длины кратчайших путей.

*Останов.*

Если некоторые пометки являются временными, перейти к шагу 2.

## Алгоритм Дейкстры (описание 2)

Алгоритм использует три массива из  $N$  (= числу вершин сети) чисел каждый.

- Первый массив  $S$  содержит метки с двумя значениями: 0 (вершина еще не рассмотрена) и 1 (вершина уже рассмотрена);
- второй массив  $B$  содержит расстояния - текущие кратчайшие расстояния от/до соответствующей вершины;
- третий массив  $C$  содержит номера вершин -  $k$ -й элемент  $C[k]$  есть номер предпоследней вершины на текущем кратчайшем пути из  $V_i$  в  $V_k$ .

Матрица расстояний  $A[i,k]$  задает длины дуге  $A[i,k]$ ; если такой дуги нет, то  $A[i,k]$  присваивается большое число  $B$ , равное "бесконечности".

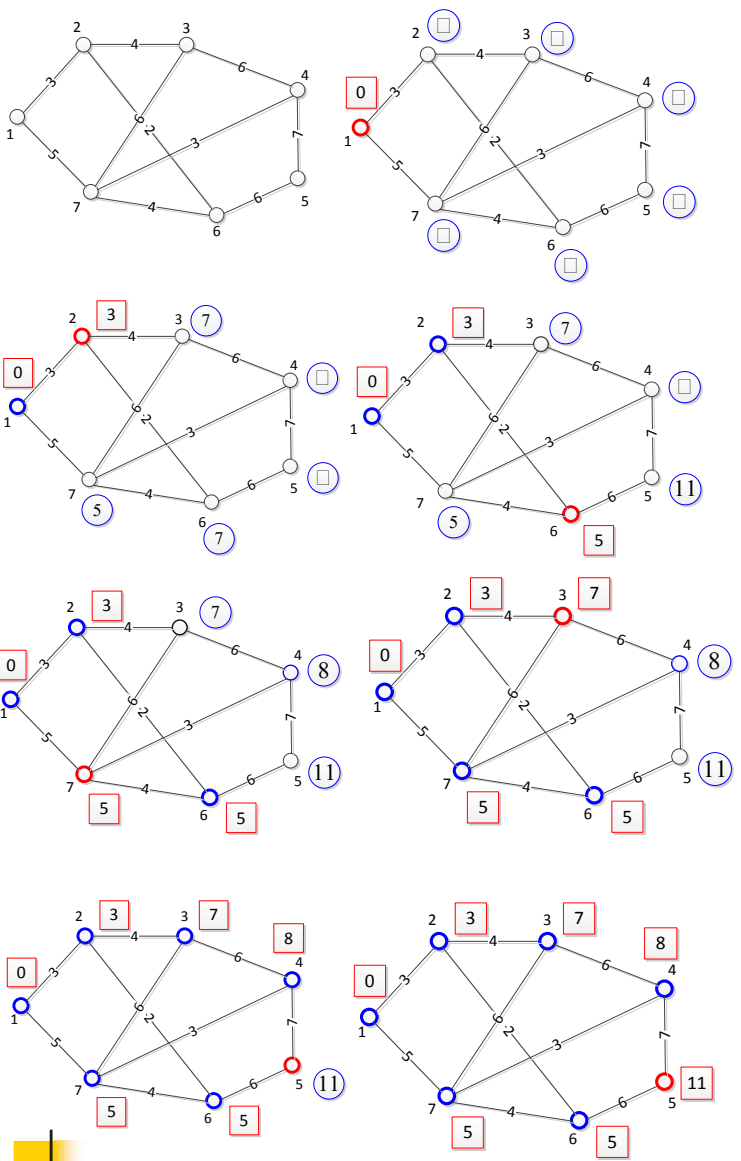
1 (инициализация). В цикле от 1 до  $N$  заполнить нулями массив  $S$ ; заполнить числом  $i$  массив  $C$ ; перенести  $i$ -ю строку матрицы  $A$  в массив  $B$ ,  
 $S[i]:=1$ ;  $C[i]:=0$  ( $i$  - номер стартовой вершины)

2 (общий шаг). Найти минимум среди неотмеченных (т. е. тех  $k$ , для которых  $S[k]=0$ ); пусть минимум достигается на индексе  $j$ , т. е.  $B[j] \leq B[k]$

Затем выполняются следующие операции:  $S[j]:=1$ ; если  $B[k] > B[j]+A[j,k]$ , то ( $B[k]:=B[j]+A[j,k]$ ;  $C[k]:=j$ ) (Условие означает, что путь  $V_i \dots V_k$  длиннее, чем путь  $V_i \dots V_j V_k$ ). (Если все  $S[k]$  отмечены, то длина пути от  $V_i$  до  $V_k$  равна  $B[k]$ . Теперь надо) перечислить вершины, входящие в кратчайший путь).

3 (выдача ответа). (Путь от  $V_i$  до  $V_k$  выдается в обратном порядке следующей процедурой:) 3.1.  $z:=C[k]$ ; 3.2. Выдать  $z$ ; 3.3.  $z:=C[z]$ . Если  $z = 0$ , то конец, иначе перейти к 3.2.

## Алгоритм Дейкстры (пример)



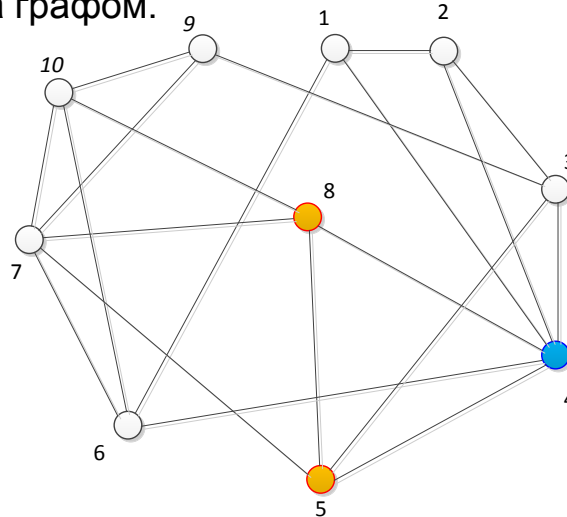
1	2	3	4	5	6	7
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$0+3=3$	$\infty$	$\infty$	$\infty$	$\infty$	5
		$3+4=7$	$\infty$	$\infty$	$3+2=5$	5
		<del><math>\infty</math></del> 7	$\infty$	$5+6=11$		<del><math>5+4=9</math></del> 5
		<del><math>5+6=11</math></del> 7	$5+3=8$	<del><math>\infty</math></del> 11		
			<del><math>7+6=13</math></del> 8	<del><math>\infty</math></del> 11		
				<del><math>8+7=15</math></del> 11		

## Алгоритм Дейкстры (поиск кратчайшего пути – пример реализации)

```
#include <iostream>
using namespace std;
int w[500][500]; // массив весов
bool used[500]; // массив использованных вершин
int d[500]; // массив длин пути
int inf=1000000000; // условная бесконечность
int main()
{
    int n,m,v1,v2,x=0,y=0,z=0;
    cin>>n>>m>>v1>>v2;
    v1--;v2--;
    memset(w,0,sizeof(w));
    memset(d,1000000000,sizeof(d));
    memset(used,false,sizeof(used));
    for (int i=0;i<n;i++) d[i]=inf;
    for (int i=0;i<m;i++)
    {cin>>x>>y>>z;w[x-1][y-1]=z;w[y-1][x-1]=z;}
    d[v1]=0;
    while (true){
        int from,zfrom=inf;
        for (int i=0;i<n;i++)
            if ((zfrom>d[i]) && !(used[i])) {from=i;zfrom=d[i];}
        if (zfrom>=inf) break;
        used[from]=true;
        for(int to=0;to<n;to++)
            if (w[from][to]!=0)
                if ((!used[to]) && (d[to]>d[from]+w[from][to])) d[to]=d[from]+w[from][to];
    }
    if (d[v2]<inf) cout<<d[v2];else cout<<"-1";
    return 0;
}
```

## Размещение узла в сети связи

Пусть имеется некоторая структура, заданная, например, расположением населенных пунктов и связывающих их дорог, или зданий и сооружений кабельной канализации между ними, модель которой может быть описана графом.



**Задача 1** состоит в том, что требуется найти такую вершину графа, от которой расстояния до других вершин минимальны, например, когда требуется обеспечить минимальную длину абонентской линии (например, при выборе места установки концентратора ADSL).

**Задача 2** состоит в том, что требуется найти такую вершину графа, от которой сумма расстояний до других вершин минимальна, например, когда требуется обеспечить минимальный расход кабеля абонентской сети.

Задача 1 – (минимаксная задача) может быть решена путем **поиска ЦЕНТРОВ** графа.

Задача 2 – (минисуммная задача) может быть решена путем **поиска МЕДИАН** графа.

## размещение центров графа

Для любой вершины  $x_i$  графа  $G = (X, \Gamma)$  пусть  $R_{\lambda 0}(x_i)$  есть множество тех вершин  $x_j$  графа  $G$ , которые достижимы из вершины  $x_i$  с помощью путей со взвешенными длинами  $v_j d(x_i, x_j)$ , не превосходящими величины  $\lambda$ .

Через  $R_{\lambda t}(x_i)$  будет обозначаться множество тех вершин  $x_j$  графа  $G$ , из которых вершина  $x_i$  может быть достигнута с использованием путей, имеющих взвешенные длины  $v_j d(x_j, x_i) \leq \lambda$ .

Таким образом,

$$R_{\lambda 0}(x_i) = \{x_j \mid v_j d(x_i, x_j) \leq \lambda, x_j \in X\}$$

$$R_{\lambda t}(x_i) = \{x_j \mid v_j d(x_j, x_i) \leq \lambda, x_j \in X\}$$

Для каждой вершины  $x_i$  определим следующие два числа:

$$S_0(x_i) = \max\{v_j d(x_i, x_j) \mid x_j \in X\} \quad \text{Число внешнего разделения}$$

$$S_t(x_i) = \max\{v_j d(x_j, x_i) \mid x_j \in X\} \quad \text{Число внутреннего разделения}$$

Если матрица  $D(G)$  - матрица расстояний (длин путей) между всеми вершинами графа, то  $S_0(x_i)$  – наибольшее число в строке, а  $S_t(x_i)$  – наибольшее число в столбце.

$$S_0(x_i^*) = \min\{S_0(x_i) \mid x_i \in X\} \quad \text{- внешний центр графа}$$

$$S_t(x_t^*) = \min\{S_t(x_i) \mid x_i \in X\} \quad \text{- внутренний центр графа}$$

## размещение центров графа

Матрица длин кратчайших путей

$D(G)=$

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	1	2	1	2	2	3	2
2	1	0	1	1	2	2	3	2	2	3
3	2	1	0	2	1	3	2	2	1	2
4	1	1	2	0	1	1	2	1	3	2
5	2	2	1	1	0	2	1	1	2	2
6	1	2	3	1	2	0	1	2	2	1
7	2	3	2	2	1	1	0	1	1	2
8	2	2	2	1	1	2	1	0	2	1
9	3	2	1	3	2	2	1	2	0	1
10	2	3	2	2	2	1	2	1	1	0

Числа  
внутреннего  
разделения

3	3	3	3	3	2	3	3	2	3	3
---	---	---	---	---	---	---	---	---	---	---

Числа  
внешнего  
разделения

Решение задачи 1  
Минимум максимумов  
по строкам

3
3
3
3
2
3
3
2
3
3

**Решение**

Вершины 5 и 8 – внешние центры графа (граф не ориентированный, следовательно, они же и внутренние центры графа)

---

## размещение медиан графа

Пусть дан граф  $G = (X, \Gamma)$ . Для каждой вершины  $x_i \in X$  определим два числа, которые назовем передаточными числами:

$$\sigma_o(x_i) = \sum_{x_j \in X} v_j d(x_i, x_j)$$

$$\sigma_t(x_i) = \sum_{x_j \in X} v_j d(x_j, x_i)$$

(внешнее и внутреннее передаточные числа)

Вершина  $x_0$  для которой  $\sigma_o(x_0) = \min_{x_i \in X} \{\sigma_o(x_i)\}$  - внешняя медиана графа

Вершина  $x_0$  для которой  $\sigma_t(x_0) = \min_{x_i \in X} \{\sigma_t(x_i)\}$  - внутренняя медиана графа



## размещение медиан графа

Матрица длин кратчайших путей

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	1	2	1	2	2	3	2
2	1	0	1	1	2	2	3	2	2	3
3	2	1	0	2	1	3	2	2	1	2
4	1	1	2	0	1	1	2	1	3	2
5	2	2	1	1	0	2	1	1	2	2
6	1	2	3	1	2	0	1	2	2	1
7	2	3	2	2	1	1	0	1	1	2
8	2	2	2	1	1	2	1	0	2	1
9	3	2	1	3	2	2	1	2	0	1
10	2	3	2	2	2	1	2	1	1	0

Решение задачи 2  
Минимум сумм по  
строкам

16
17
16
14
14
15
15
14
37
25

Внутренние  
передаточные  
числа

16	17	16	14	14	15	15	14	37	25
----	----	----	----	----	----	----	----	----	----

Внешние  
передаточны  
е числа

**Решение**

Вершина 4,5,7 – внешние медианы графа (граф не ориентированный, следовательно, они же и внутренние медианы графа)

# размещение узла в сети связи – поиск центра и медианы графа

## 2. Матрица длин кратчайших путей

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	1	2	1	2	2	3	2
2	1	0	1	1	2	2	3	2	2	3
3	2	1	0	2	1	3	2	2	1	2
4	1	1	2	0	1	1	2	1	3	2
5	2	2	1	1	0	2	1	1	2	2
6	1	2	3	1	2	0	1	2	2	1
7	2	3	2	2	1	1	0	1	1	2
8	2	2	2	1	1	2	1	0	2	1
9	3	2	1	3	2	2	1	2	0	1
10	2	3	2	2	2	1	2	1	1	0

Решение задачи 1  
Минимум максимумов  
по строкам

Решение задачи 2  
Минимум сумм по  
строкам

3	16
3	17
3	16
3	14
2	14
3	15
3	15
2	14
3	37
3	25

Числа  
внутреннего  
разделения

Внутренние  
передаточные  
числа

3	3	3	3	2	3	3	2	3	3
16	17	16	14	14	15	15	14	37	25

Числа  
внешнего  
разделения

Внешние  
передаточны  
е числа

### Решение

Вершины 5 и 8 – внешние центры графа (граф не ориентированный, следовательно, они же и внутренние центры графа)

Вершина 4 – внешняя медиана графа (граф не ориентированный, следовательно, она же и внутренняя медиана графа)

(радиус графа = 2)

## Вычисление длин кратчайших путей между вершинами

Граф задан матрицей длин ребер между вершинами.

C=

	1	2	3	4	5	6	7	8	9	10
1	0	1	$\infty$	1	$\infty$	1	$\infty$	$\infty$	$\infty$	$\infty$
2	1	0	1	1	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
3	$\infty$	1	0	$\infty$	1	$\infty$	$\infty$	$\infty$	1	$\infty$
4	1	1	$\infty$	0	1	1	$\infty$	1	$\infty$	$\infty$
5	$\infty$	$\infty$	1	1	0	$\infty$	1	1	$\infty$	$\infty$
6	1	$\infty$	$\infty$	1	$\infty$	0	1	$\infty$	$\infty$	1
7	$\infty$	$\infty$	$\infty$	$\infty$	1	1	0	1	1	$\infty$
8	$\infty$	$\infty$	$\infty$	1	1	$\infty$	1	0	$\infty$	1
9	$\infty$	$\infty$	1	$\infty$	$\infty$	$\infty$	1	$\infty$	0	1
10	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	1	$\infty$	1	1	0

Для решения задачи необходимо найти кратчайшие пути между всеми вершинами графа.

Для этой цели может быть использован алгоритм Флойда-Уоршалла

D=

	1	2	3	4	5	6	7	8	9	10
1	1	2	2	4	4	6	6	4	2	6
2	1	2	3	4	3	1	3	4	3	1
3	2	2	3	2	5	2	5	5	9	9
4	1	2	2	4	5	6	5	8	2	6
5	4	3	3	4	5	4	7	8	3	8
6	1	1	1	4	4	6	7	4	7	10
7	6	5	5	5	5	6	7	8	9	6
8	4	4	5	4	5	4	7	8	7	10
9	3	3	3	3	3	7	7	7	9	10
10	6	6	9	6	8	6	6	8	9	10

1. Результат – матрица кратчайших путей.

## Алгоритм Флойда-Уоршелла

Задана матрица весов  $c=[c_{ij}]$   $i, j = 1 \dots n$

Пусть вершины графа пронумерованы от 1 до  $n$  и введено обозначение  $C_{kij}$  для длины кратчайшего пути от  $i$  до  $j$ , который кроме самих вершин  $i$  и  $j$  проходит через вершины  $1 \dots k$ . Очевидно, что  $C_{0ij}$  — длина (вес) ребра  $(i, j)$ , если таковое существует (в противном случае его длина может быть обозначена как  $\infty$ ).

Существует два варианта значения  $C_{kij}, k \in (1, \dots, n)$ :

Кратчайший путь между  $i, j$  не проходит через вершину  $k$ , тогда  $C_{kij} = C_{k-1 ij}$

Существует более короткий путь между  $i, j$ , проходящий через  $k$ , тогда он сначала идёт от  $i$  до  $k$ , а потом от  $k$  до  $j$ . В этом случае, очевидно,  $C_{kij} = C_{ik} + C_{kj}$

Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений.

Тогда формула для  $C_{kij}$  имеет вид:

$C_{0ij}$  — длина ребра  $(i, j)$ ;

$C_{kij} = \min(C_{ij}, C_{ik} + C_{kj})$ .

Алгоритм Флойда-Уоршелла последовательно вычисляет все значения  $C_{kij}, \forall i, j$  для  $k = 1 \dots n$ . Полученные значения  $C_{nij}$  являются длинами кратчайших путей между вершинами  $i, j$ .

Предположим, что в начальной матрице весов  $c_{ij} = 0$  для всех  $i = 1, 2, \dots, n$  и  $c_{ij} = \infty$ , если в графе отсутствует дуга  $(x_i, x_j)$ .

*Присвоение начальных значений*

Шаг 1. Положить  $k = 0$ .

*Итерация*

Шаг 2.  $k = k + 1$ .

Шаг 3. Для всех  $i \neq k$ , таких, что  $c_{ik} \neq \infty$ , и для всех  $j \neq k$ , таких, что  $c_{kj} \neq \infty$ , введем операцию

$$c_{ij} = \min\{c_{ij}, (c_{ik} + c_{kj})\}$$

*Проверка на окончание*

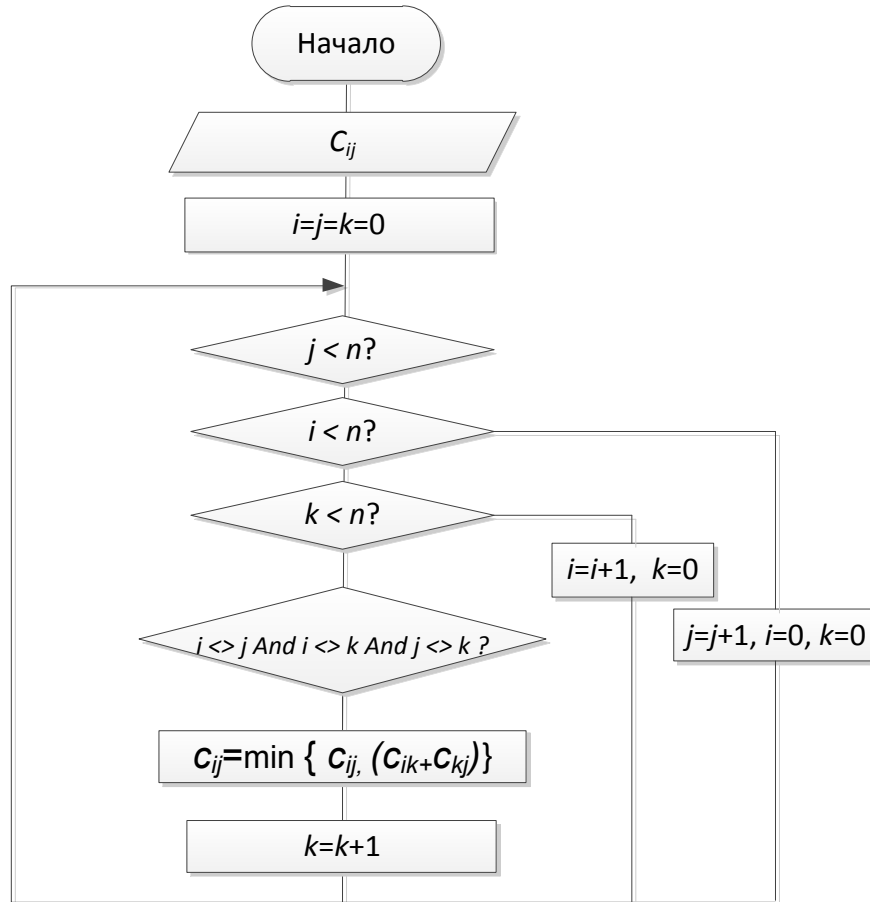
Шаг 4.

(а) Если  $c_{ij} < 0$ , то в графе  $G$  существует цикл отрицательного веса, содержащий вершину  $x_i$  и решения нет. Останов.

(б) Если все  $c_{ij} \geq 0$  и  $k = n$ , то получено решение. Матрица  $[c_{ij}]$  дает длины всех кратчайших путей. Останов.

(в) Если все  $c_{ij} \geq 0$  и  $k < n$ , то вернуться к шагу 2.

# Алгоритм Флойда-Уоршалла (нахождение всех кратчайших путей в графе)



# Пример реализации алгоритма Флойда-Уоршелла на VB

```
Public Function Floid()  
Dim l, j, k As Integer  
Dim x, y As Double  
xBase = 5  
yBase = 8  
  
n = Worksheets("CM").Cells(6, 5)  
ReDim d(n, n) As Long           'Массив длин дуг – исходный длин путей - результат  
ReDim Node(n, n) As Integer     'Массив путей  
For j = 1 To n                  'Загрузить матрицу длин дуг  
    d(i, j) = Worksheets("CM").Cells(yBase + i, xBase + j)  
    If d(i, j) = 0 Then  
        d(i, j) = 9999999  
    End If  
    If i = j Then  
        d(i, j) = 0  
    End If  
Node(i, j) = j  
Next i  
Next j  
  
'Triple operation to update the D Matr (Собственно реализация алгоритма Флойда-Уоршалла)  
For j = 1 To n  
    For i = 1 To n  
        For k = 1 To n  
            If i <> j And i <> k And j <> k Then  
                x = d(i, j) + d(j, k)  
                y = d(i, k)  
                If x < y Then  
                    d(i, k) = x  
                    Node(i, k) = Node(i, j)  
                End If  
            End If  
        Next k  
    Next i  
Next j  
For i = 1 To n  
    For j = 1 To n  
        Worksheets("CM").Cells(22 + j, 5 + i) = Node(j, i)  
        Worksheets("CM").Cells(34 + j, 5 + i) = d(j, i)  
    Next j  
Next i  
End Function
```

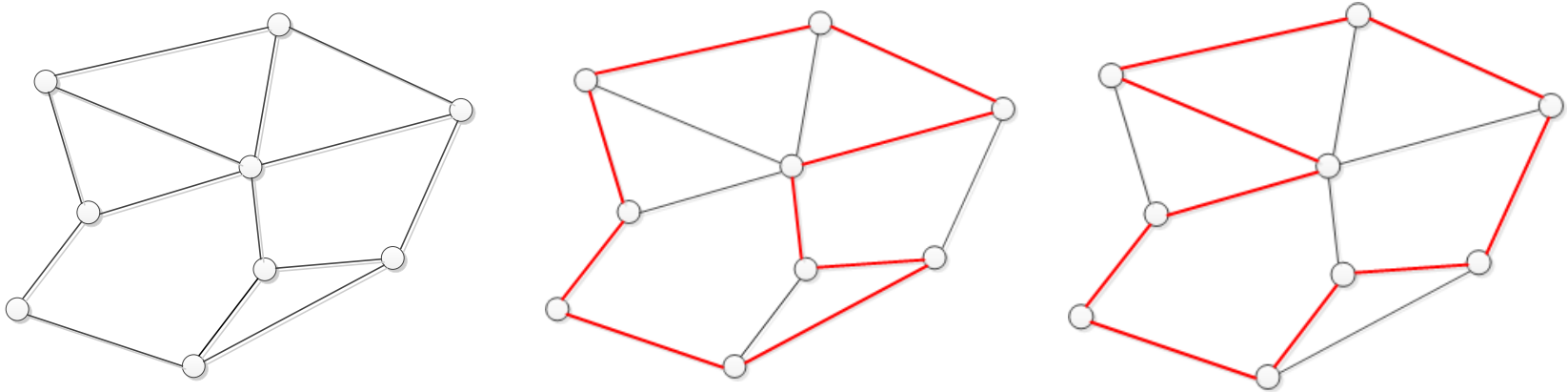
# Задача коммивояжера

(TSP - Travelling salesman problem)

Гамильтонов цикл

Гамильтонова цепь (путь)

**Условие Оре:** если  $n$  количество вершин в графе и  $n > 2$  и если для любой пары несмежных вершин  $i$  и  $j$  выполняется неравенство  $d_i + d_j \geq n - 1$ , то граф является гамильтоновым графом (сумма степеней любых двух несмежных вершин не меньше числа вершин в графе).

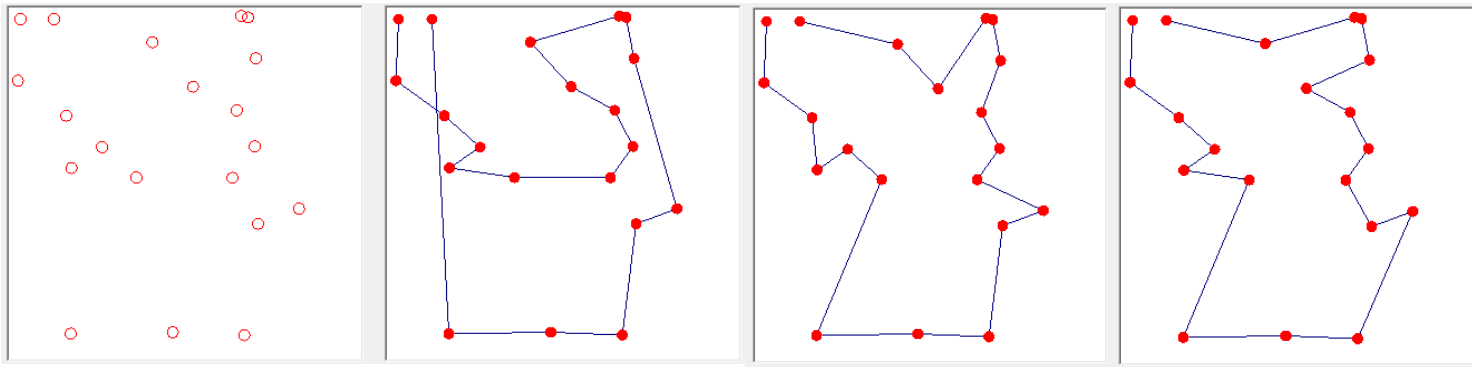


**Задача коммивояжера:** нахождение Гамильтонова цикла (цепи) наименьшей длины.  
NP – полная задача

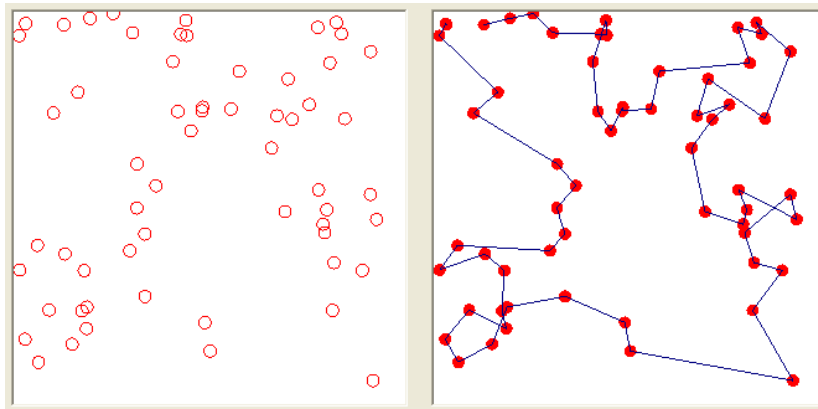
# Приближенные решения

- Метод ближайшего соседа
- Метод штрафования вершин
- Метод ветвей и границ

.....



**Метод ближайшего соседа, штрафования вершин, случайное штрафование вершин**





# Муравьиный алгоритм

## (Ant Colony Optimization ACO)

Имитация поведения муравьиной колонии.

В одной или нескольких вершинах графа находятся муравьи, которые могут двигаться по ребрам графа между вершинами. При движении они оставляют феромоновый след, который привлекает других муравьев. Вероятность выбора того или иного ребра зависит от расстояния между вершинами и количества оставленного на нем феромона, предшественниками. Феромон может выветриваться со временем.

При выборе пути учитываются два фактора:

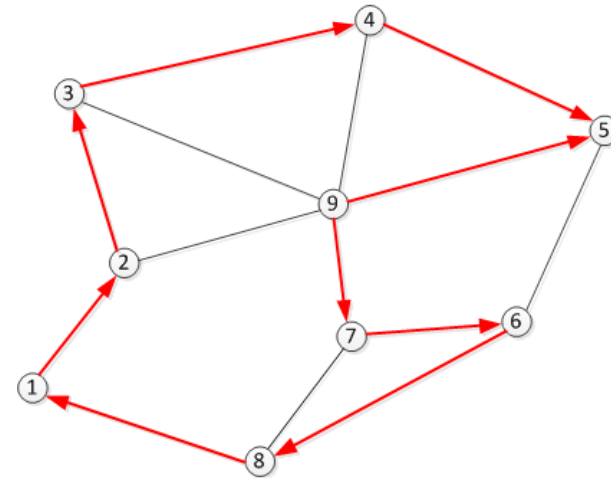
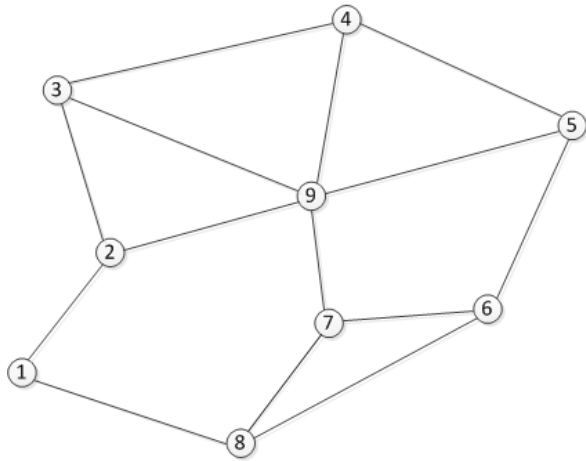
- расстояние между узлами;
- количество феромона.

$$p_{ij} = \frac{\eta_{ij}^{\beta} \tau_{ij}^{\alpha}}{\sum_{k \in D} (\eta_{ik}^{\beta} \tau_{ik}^{\alpha})} \quad D - \text{множество номеров достижимых вершин (из вершины } i).$$

$$\eta_{ij}^{\beta} = \frac{A}{L_{ij}} \quad - \text{характеризует удаленность вершины. } L_{ij} - \text{расстояние между вершинами}$$

$$\tau_{ij}^{\alpha} \quad - \text{характеризует привлекательность ребра между вершинами.}$$

## муравьиный алгоритм



n	Edge	L	tau	eta_k	tau_k		sum		random
1	1-2	41,3	5,6	0,024	5,6	0,135	0,204	0,66	0,397
2	1-8	39,6	2,7	0,025	2,7	0,068	0,204	0,34	
3	2-3	14,5	2,9	0,069	2,9	0,203	0,642	0,32	0,205
4	2-9	22,0	9,7	0,045	9,7	0,439	0,642	0,68	
5	3-4	25,1	7,1	0,040	7,1	0,282	0,489	0,58	0,038
6	3-9	44,4	9,2	0,023	9,2	0,207	0,489	0,42	
7	4-9	12,6	9,3	0,079	9,3	0,733	0,990	0,74	0,987
8	4-5	31,5	8,1	0,032	8,1	0,257	0,990	0,26	
9	5-9	41,6	9,4	0,024	9,4	0,227	0,931	0,24	0,218
10	5-6	6,1	4,3	0,164	4,3	0,704	0,931	0,76	
11	6-7	16,1	5,4	0,062	5,4	0,333	0,406	0,82	0,218
12	6-8	49,0	3,6	0,020	3,6	0,073	0,406	0,18	
13	7-9	13,8	8,0	0,073	8,0	0,580	0,917	0,63	0,261
14	7-8	5,0	1,7	0,198	1,7	0,337	0,917	0,37	
15	7-6	6,1	5,4	0,164	5,4	0,877	1,214	0,72	0,919
16	7-9	13,8	1,7	0,073	1,7	0,123	1,000	0,12	

---

## муравьиный алгоритм

$$L_0^{(r)} = L_{12} + L_{23} + L_{34} + L_{45} + L_{59} + L_{97} + L_{76} + L_{68} + L_{81}$$

$$\Delta \tau^{(r)} = \frac{Q}{L_0^{(r)}}$$

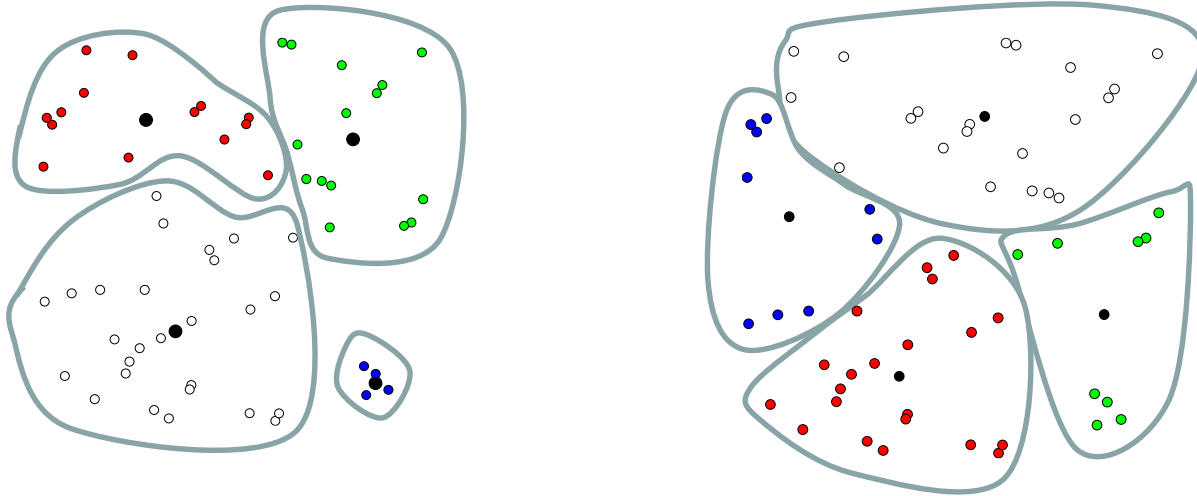
$$\tau_{kj}^{(*)} = (1 - p)\tau_{kj} + \Delta \tau^{(r)}$$

# Кластерный анализ (кластеризация)

Задано множество объектов  $A = \{a_1, a_2, \dots, a_n\}$

Объекты имеют некоторые характеристики (например, координаты). Задача кластеризации состоит в выделении подмножеств объектов - кластеров, таким образом, чтобы в рамках кластера свойства объектов были близки, а между объектами разных кластеров они максимально отличались.

Примером может служить разбиение множества точек на плоскости на подмножества, по признаку близости их координат.

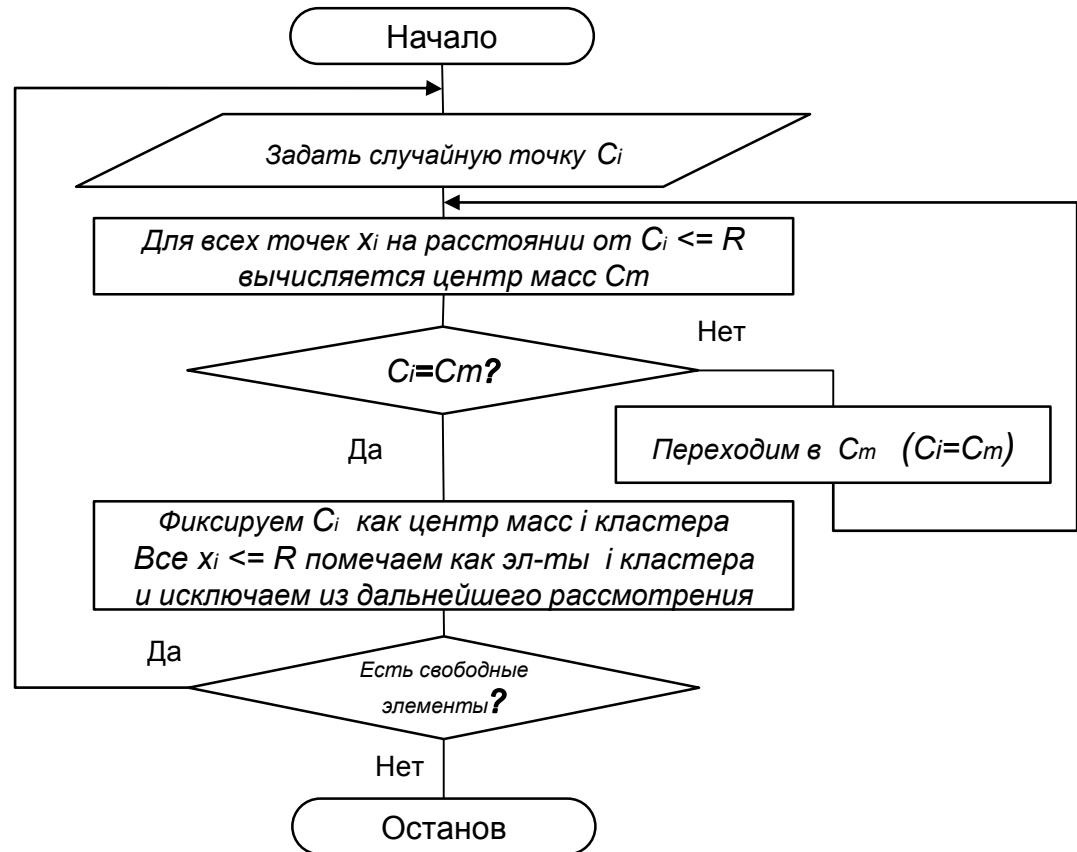
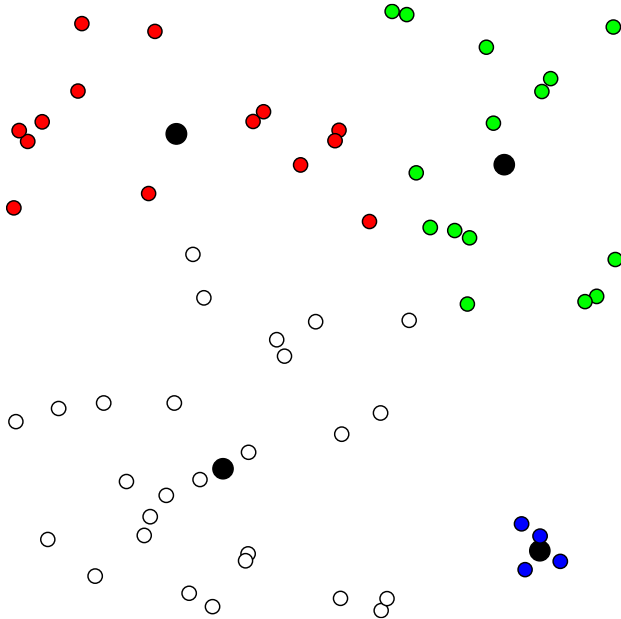


Решение задачи заключается в минимизации суммарного отклонения расстояний (метрик) объектов от центров кластеров (центров масс)

# Алгоритм кластеризации FOREL (произвольный элемент)

Задано множество объектов  $A = \{a_1, a_2, \dots, a_n\}$

Объекты имеют некоторые характеристики (например, координаты на плоскости  $x$  и  $y$ ).  
Задан размер (радиус) кластера  $R$ .

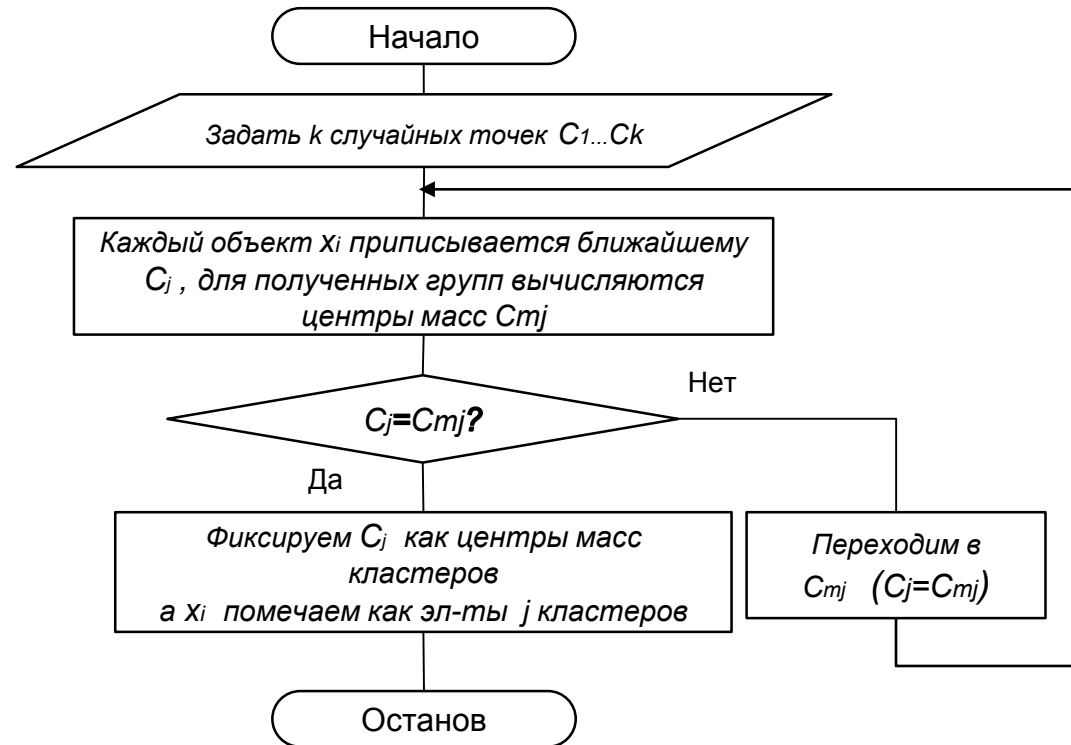
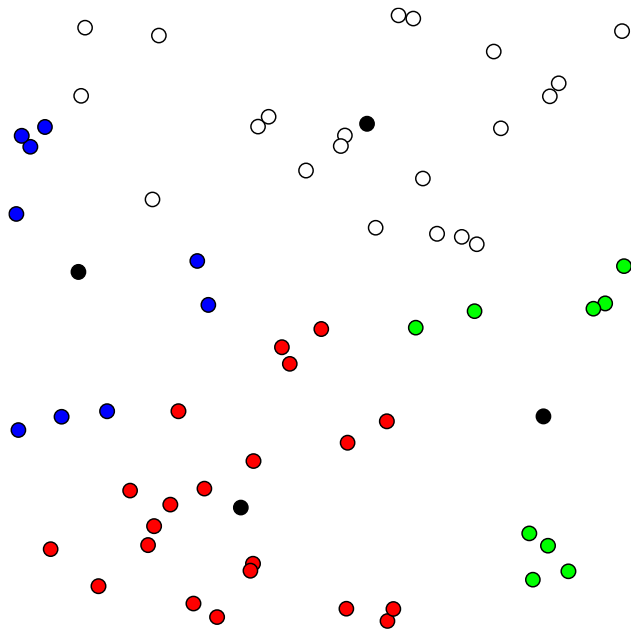


В результате решения получается некоторое число кластеров, средний размер которых близок к  $R$

# Алгоритм кластеризации $k$ -средних ( $k$ -means)

Задано множество объектов  $A = \{a_1, a_2, \dots, a_n\}$

Объекты имеют некоторые характеристики (например, координаты на плоскости  $x$  и  $y$ ).  
Задано число кластеров  $k$ .



В результате решения получает  $k$  кластеров

# Алгоритм нечеткой кластеризации *c-means*

Задано множество объектов  $X = \{x_1, x_2, \dots, x_n\}$

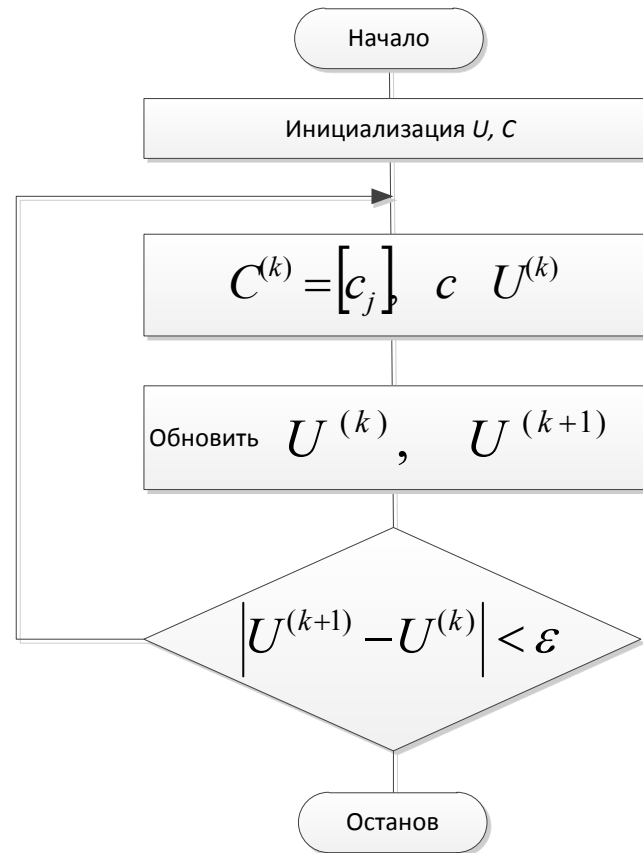
Объекты имеют некоторые характеристики (например, координаты на плоскости  $x$  и  $y$ ).  
Задано число кластеров  $k$ .

$$J_m = \sum_{i=1}^N \sum_{j=1}^C u_{ij}^m \|x_i - C_j\|^2, \quad m > 1$$

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left( \frac{\|x_i - C_j\|}{\|x_i - C_k\|} \right)^{\frac{2}{m-1}}}$$

$$C_j = \frac{1}{\sum_{i=1}^N u_{ij}^m} \sum_{i=1}^N u_{ij}^m \cdot x_j$$

$$\max_{ij} \left\{ u_{ij}^{(k+1)} - u_{ij}^{(k)} \right\} < \varepsilon$$



$$C_j = \frac{1}{\sum_{i=1}^N u_{ij}^m} \sum_{i=1}^N u_{ij}^m \cdot x_j$$

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left( \frac{\|x_i - C_j\|}{\|x_i - C_k\|} \right)^{\frac{2}{m-1}}}$$

# Применение кластерного анализа для выбора структуры сети

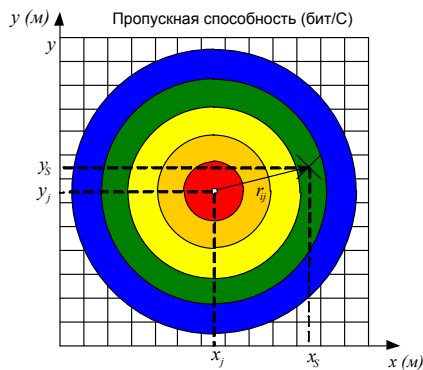


# Сеть беспроводной связи

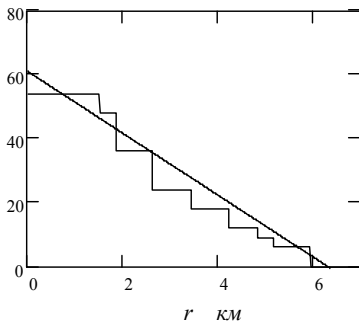
- зависимость качества обслуживания от SNR
- схема модуляции
  - вид модуляции
  - метод кодирования
- зависимость SNR от уровня сигнала
- зависимость уровня от затухания
- зависимость затухания от расстояния
- зависимость QoS от расстояния
- модель зависимости
- выбор координат БС

# Зависимость пропускной способности (интенсивности обслуженного трафика) от распределения трафика по территории

Пропускная способность БС

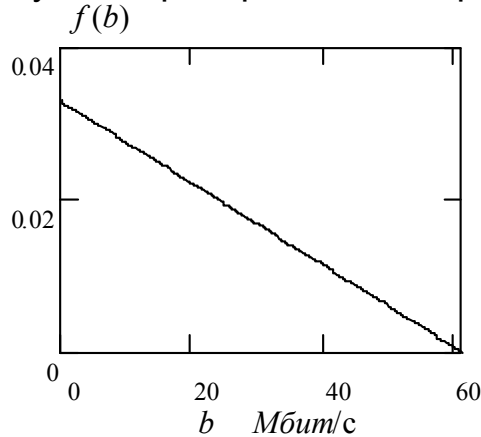


$b(r)$  Мбум/с

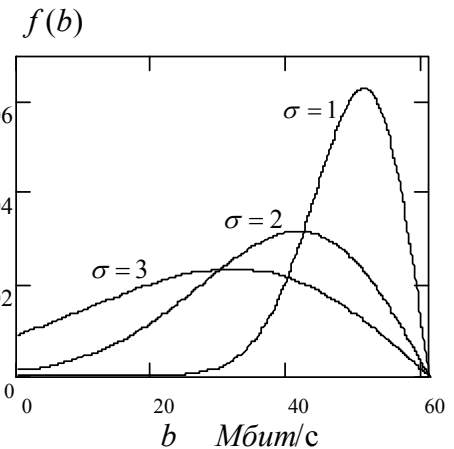


$$b(r) = b_0 \left( 1 - \frac{r}{R} \right)$$

Функция распределения пропускной способности БС



Равномерное  
распределение  
трафика по  
территории



Нормальное  
распределение  
трафика по  
территории

$$\arg \max_{x,y} \{b(x,y)\}, \quad a \leq x \leq b, \quad c \leq y \leq d$$

## Выбор координат базовой станции при произвольном законе распределения трафика по территории

1. При некоторых допущениях координаты размещения базовой станции будут совпадать с центром «масс» абонентского трафика.
2. В качестве целевой функции используется максимум пропускной способности (интенсивности обслуженного трафика), от координат размещения базовых станций.

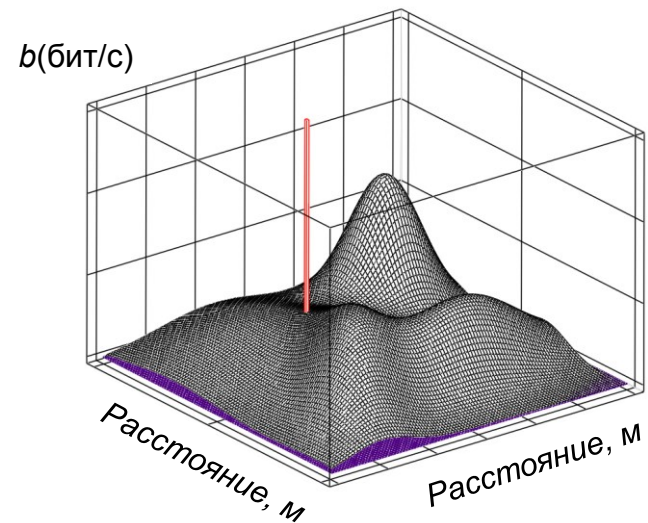
$$x_0 = \frac{1}{A} \iint_S y \cdot f_a(x, y) dx dy \quad (\text{м})$$

$$y_0 = \frac{1}{A} \iint_S x \cdot f_a(x, y) dx dy \quad (\text{м})$$

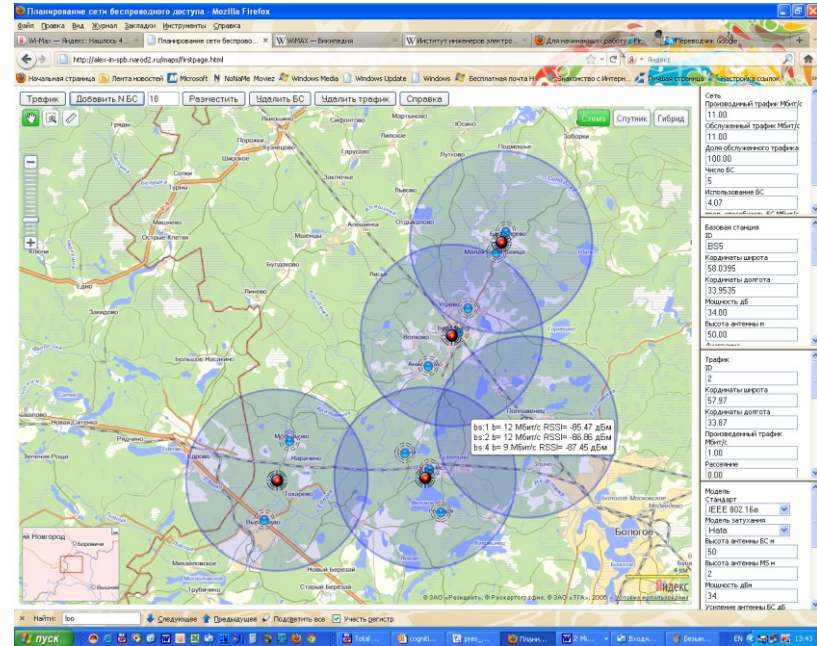
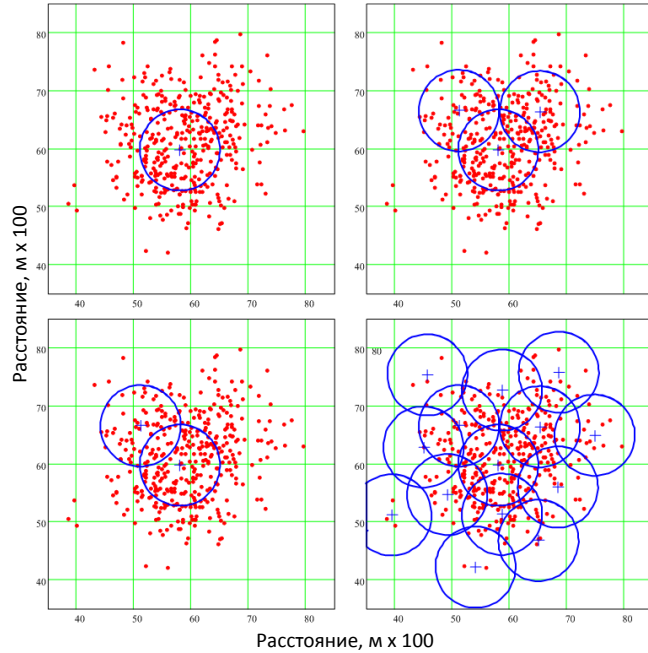
$$A = \iint_S f_a(x, y) dx dy \quad (\text{бит/с})$$

$$x_0 = \frac{1}{A} \sum_{i=1}^N a_i \cdot x_i \quad (\text{м}) \quad y_0 = \frac{1}{A} \sum_{i=1}^N a_i \cdot y_i \quad (\text{м})$$

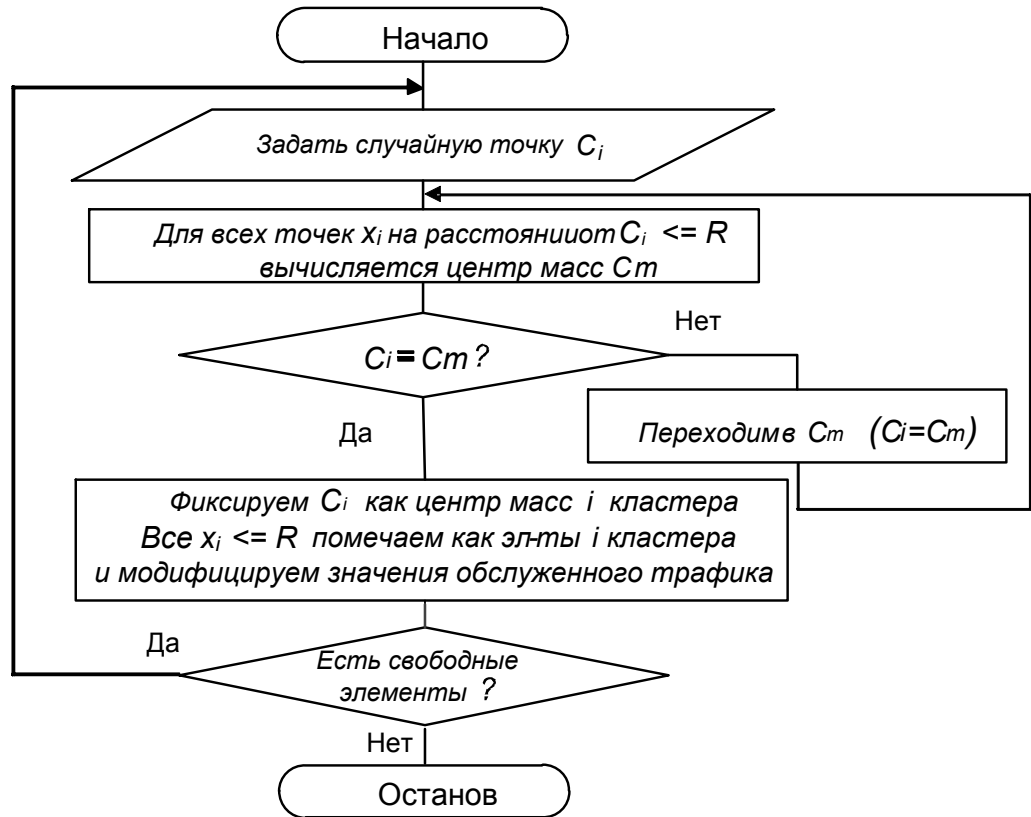
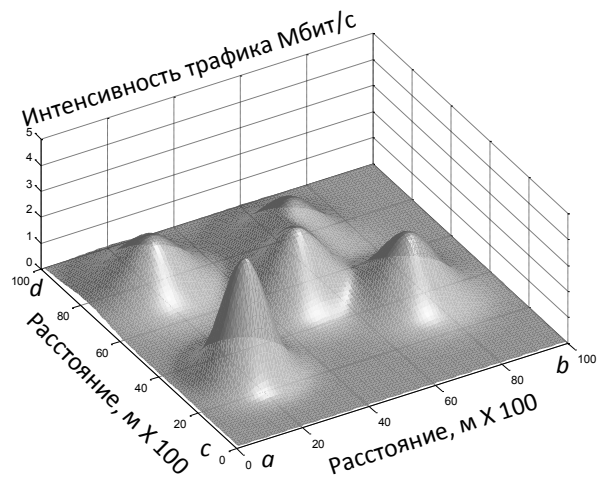
$$A = \sum_{i=1}^N a_i \quad (\text{бит/с})$$



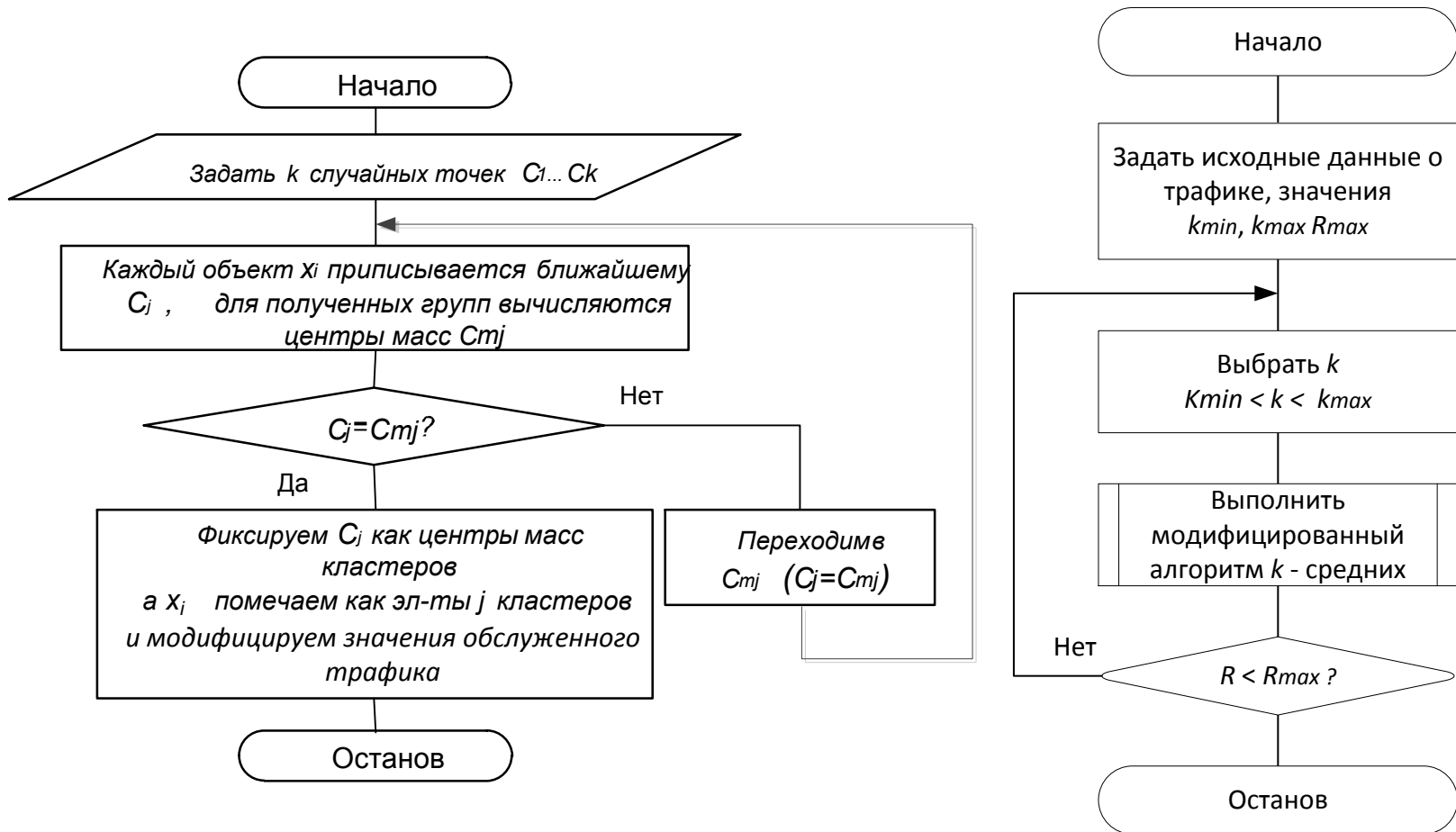
## 4.4 Моделирование и реализация, публикации



# Модифицированный метод FOREL



## Модифицированный метод k-средних



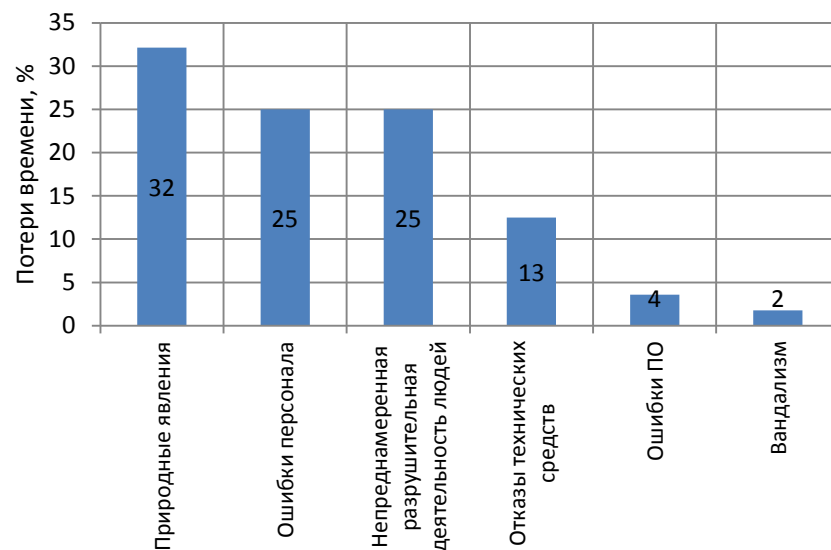
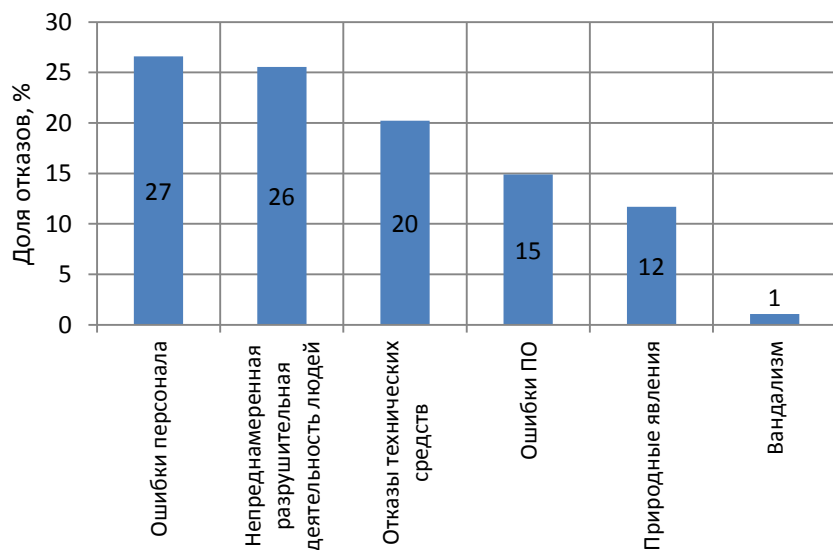
# Задача о надежности сети

(Применение динамического программирования )

# Иллюстрации (надежность сетей связи)

Alcatel-Lucent Bell Labs

Уязвимости системы телекоммуникаций

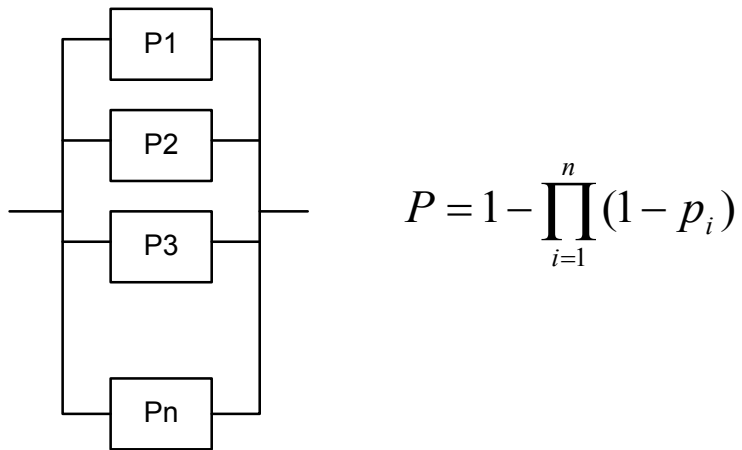
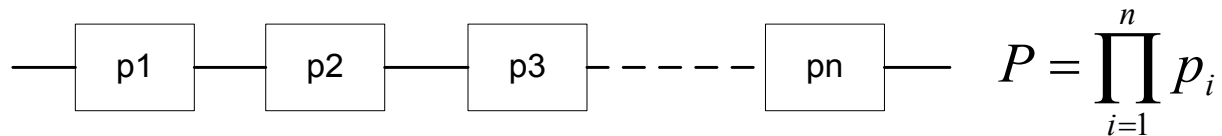


США (FCC)

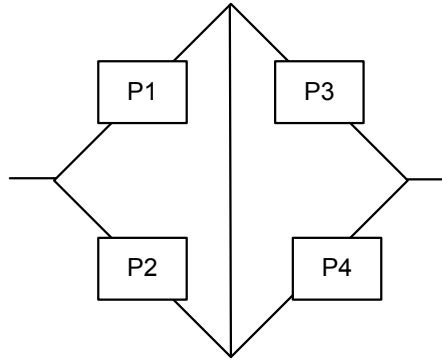
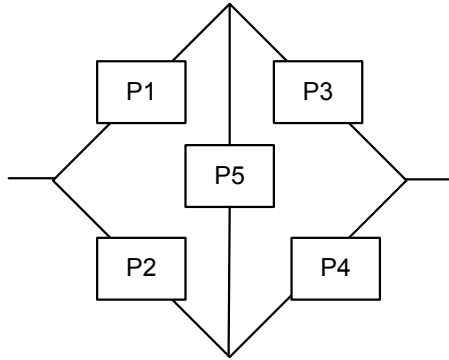
N	Группа факторов	Доля от общего количества отказов, %	Потери пользовательского времени, %
1	Отказы технических средств	19	7
2	Перегрузки сети	6	44
3	Ошибки ПО	14	2
4	Ошибки персонала	25	14
5	Вандализм	1	1
6	Непреднамеренная разрушительная деятельность людей	24	14
7	Природные явления	11	18



## Надежность простейших структур (последовательное и параллельное включение элементов)

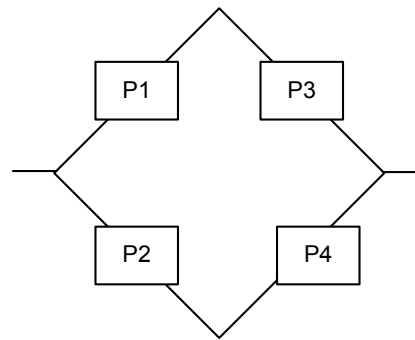


## Мостовая схема включения



$$P(W | e_5) = (1 - q_1 q_2)(1 - q_3 q_4)$$

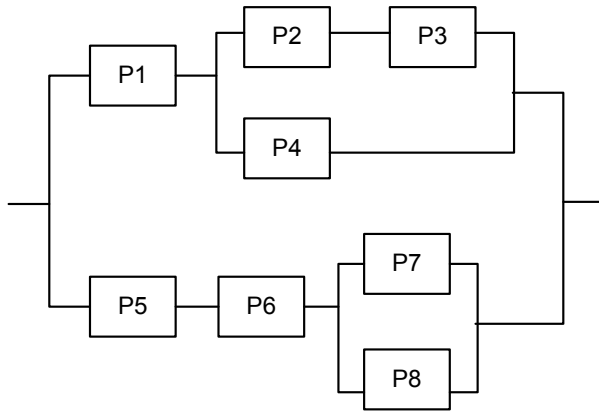
$$P = p_5 P(W | e_5) + q_5 P(W | \bar{e}_5)$$



$$P(W | \bar{e}_5) = 1 - (1 - p_1 p_3)(1 - p_2 p_4)$$

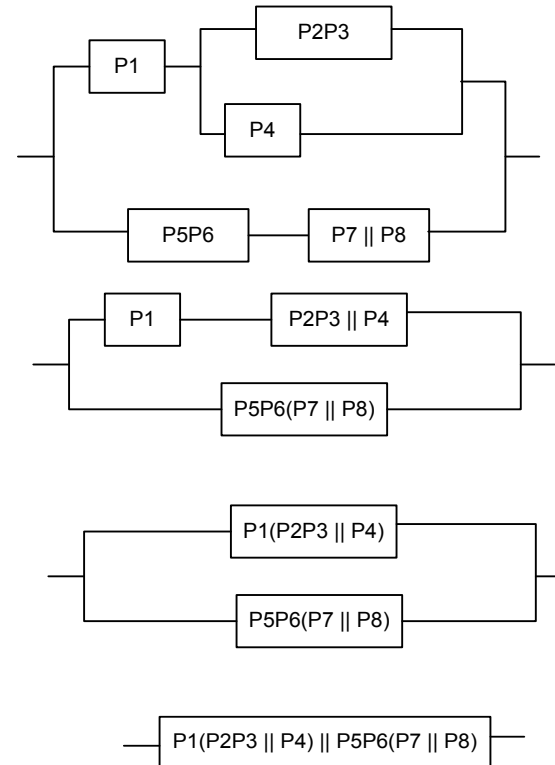
$$P = p_5 (1 - q_1 q_2)(1 - q_3 q_4) + q_5 (1 - (1 - p_1 p_3)(1 - p_2 p_4))$$

## Метод декомпозиции



$$p_2 p_3 \parallel p_4 = 1 - (1 - p_2 p_3)(1 - p_4)$$

$$p_7 \parallel p_8 = 1 - (1 - p_7)(1 - p_8)$$



$$P = p_1(1 - (1 - p_2 p_3)(1 - p_4)) \parallel p_5 p_6(1 - (1 - p_7)(1 - p_8)) =$$

$$= 1 - (1 - p_1(1 - (1 - p_2 p_3)(1 - p_4)))(1 - p_5 p_6(1 - (1 - p_7)(1 - p_8)))$$

## Метод включения-исключения (IE – Inclusion – Exclusion)

Пусть граф сети имеет  $l$  путей между заданными двумя узлами

$E_j$  Событие, которое заключается в том что все элементы пути  $T_j$  исправны

$$P_r(E_j) = \prod_{i \in T_j} p_i$$

Система из  $l$  работает, когда работает хотя бы один путь

### 3.5 Метод добавления-удаления (IE – inclusion-exclusion)

Неравенства Бонферрони

$$P = P_r\left(\bigcup_{j=1}^l E_j\right)$$

$$P \leq S_1$$

$$P = \sum_{k=1}^l (-1)^{k-1} S_k$$

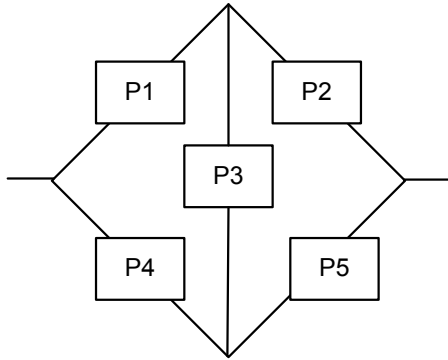
$$P \geq S_1 - S_2$$

$$P \leq S_1 - S_2 + S_3$$

$$S_k = \sum_{1 \leq i_1 < i_2 < \dots < i_k \leq l} P_r(E_{i_1} \cap E_{i_2} \cap \dots \cap E_{i_k})$$

$$P \geq S_1 - S_2 + S_3 - S_4$$

## Метод включения-исключения (пример)



$$P_r(MP_1) = p_1 p_2$$

$$P_r(MP_2) = p_4 p_5$$

$$P_r(MP_3) = p_1 p_3 p_5$$

$$P_r(MP_4) = p_2 p_3 p_4$$

$$S_1 = P_r(MP_1) + P_r(MP_2) + P_r(MP_3) + P_r(MP_4) = p_1 p_2 + p_4 p_5 + p_1 p_3 p_5 + p_2 p_3 p_4$$

$$\begin{aligned} S_2 &= P_r(MP_1 MP_2) + P_r(MP_1 MP_3) + P_r(MP_1 MP_4) + P_r(MP_2 MP_3) + P_r(MP_2 MP_4) + P_r(MP_3 MP_4) = \\ &= p_1 p_2 p_4 p_5 + p_1 p_2 p_3 p_5 + p_1 p_2 p_3 p_4 + p_1 p_3 p_4 p_5 + p_2 p_3 p_4 p_5 + p_1 p_2 p_3 p_4 p_5 \end{aligned}$$

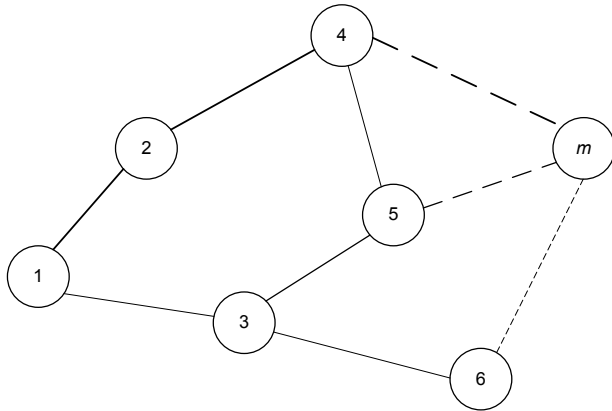
$$\begin{aligned} S_3 &= P_r(MP_1 MP_2 MP_3) + P_r(MP_1 MP_2 MP_4) + P_r(MP_1 MP_3 MP_4) + P_r(MP_2 MP_3 MP_4) = \\ &= 4 p_1 p_2 p_3 p_4 p_5 \end{aligned}$$

$$S_4 = P_r(MP_1 MP_2 MP_3 MP_4) = p_1 p_2 p_3 p_4 p_5$$

$$\begin{aligned} P &= S_1 - S_2 + S_3 - S_4 = p_1 p_2 + p_4 p_5 + p_1 p_3 p_5 + p_2 p_3 p_4 - (p_1 p_2 p_5 + p_1 p_2 p_3 p_5 + p_1 p_2 p_3 p_4 + \\ &+ p_2 p_3 p_4 p_5 + p_1 p_2 p_3 p_4 p_5) + 2 p_1 p_2 p_3 p_4 p_5 \end{aligned}$$

## Оптимизация надежности сети связи (метод динамического программирования)

Рассматривается сеть связи. Структура сети задана графом. Сеть состоит из  $m$  узлов связи и  $n$  направлений связи (ребер графа).



$x_j$  Число каналов в  $j$  направлении связи

$w_j$  Надежность канала в  $j$  направлении связи  
(вероятность исправного состояния)

$v_j$  Значимость  $j$  направления связи

$$v_j = 0 \dots 1 \quad \sum_{j=1}^n v_j = 1$$

$N$  Общее число каналов в сети = *const*

### 1. Формулировка задачи

- предметная область - надежность сети связи (вероятность исправного состояния)
- состояние сети задано исходными данными
- требуется найти оптимальное число каналов в каждом из направлений связи для получения максимальной надежности сети, при заданном общем числе каналов  $N$ .

### 2. Построение модели системы

- полагаем, что направление исправно, если исправен хотя бы один канал;
- тогда модель направления связи может быть представлена как вероятность его исправного состояния

$$W_j = 1 - (1 - w_j)^{x_j}; \quad j = 1 \dots n$$

### 3. Выбор параметров управления и показателей состояния

-параметры управления: число каналов в каждом из направлений связи,

 $x_j$ 

-показатели состояния: вероятность исправного состояния каждого из направлений связи.

 $p_j$ 

### 4. Построение целевой функции

Выберем в качестве целевого показателя средневзвешенную вероятность исправного состояния сети

$$\bar{w} = \sum_{j=1}^n v_j \cdot W_j; \quad j = 1 \dots n \quad \text{где} \quad \sum_{j=1}^n v_j = 1$$

Целевая функция

$$x_j = \arg \max_{x_j} \{ \bar{w}(x_j) \} \quad \text{при} \quad \sum_{j=1}^n x_j = N$$

### 5. Выбор метода оптимизации целевой функции

### 6. Решение задачи

Эти этапы будут рассмотрены ниже.

# Примеры решения задач оптимизации



## 1. Вычисление 1 формулы Эрланга

## 2. Оптимизация надежности

## 2. Оптимизация качества обслуживания в сети с КК

## 2. Оптимизация качества обслуживания в сети с КП

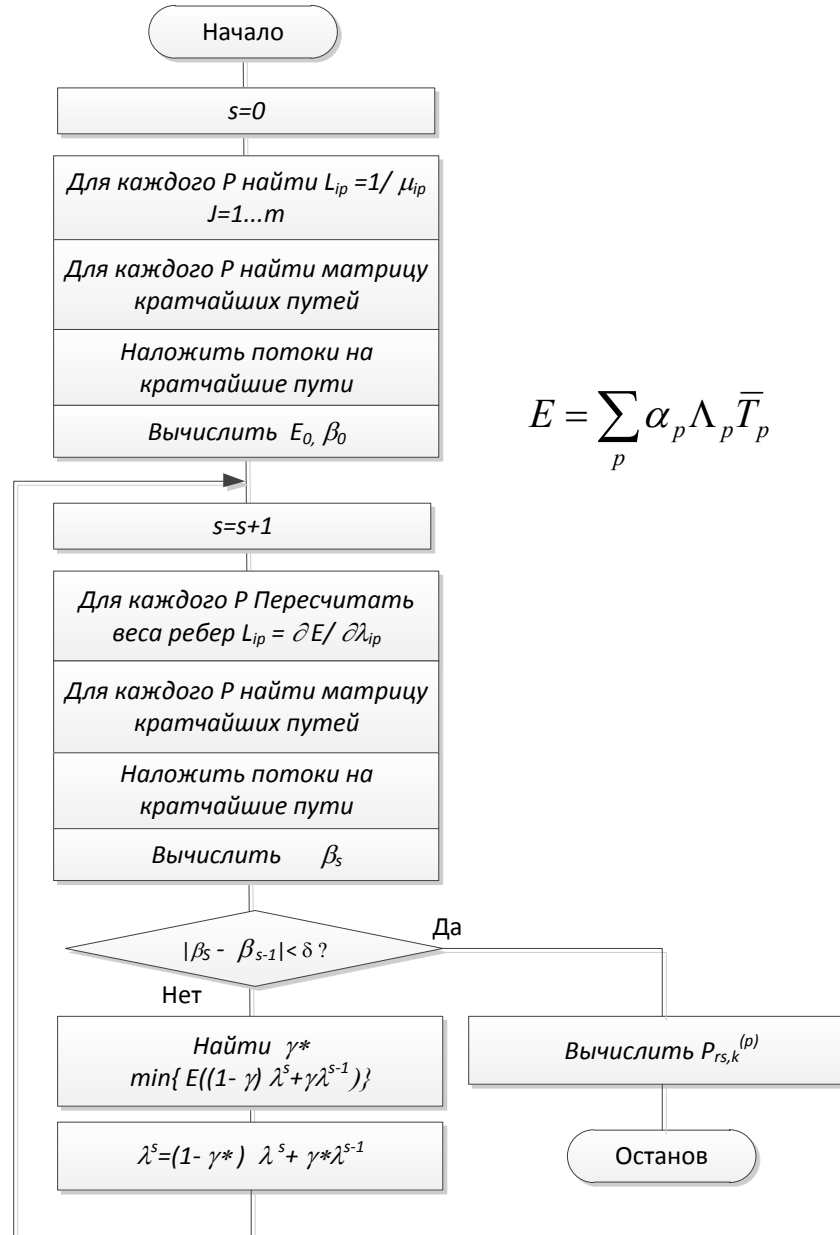
# Задача распределения трафика и Оптимизация структуры сети

# Приоритетное обслуживание

$$W_p = \frac{V}{2(1 - \rho'_{p-1})(1 - \rho'_p)} \quad \rho'_p = \sum_{i=1}^P \rho_i \quad \rho'_i = \lambda_i \cdot \bar{t}_i$$

$$V = \sum_{i=1}^P \lambda_i \sigma_i^2$$

# Алгоритм распределения трафика



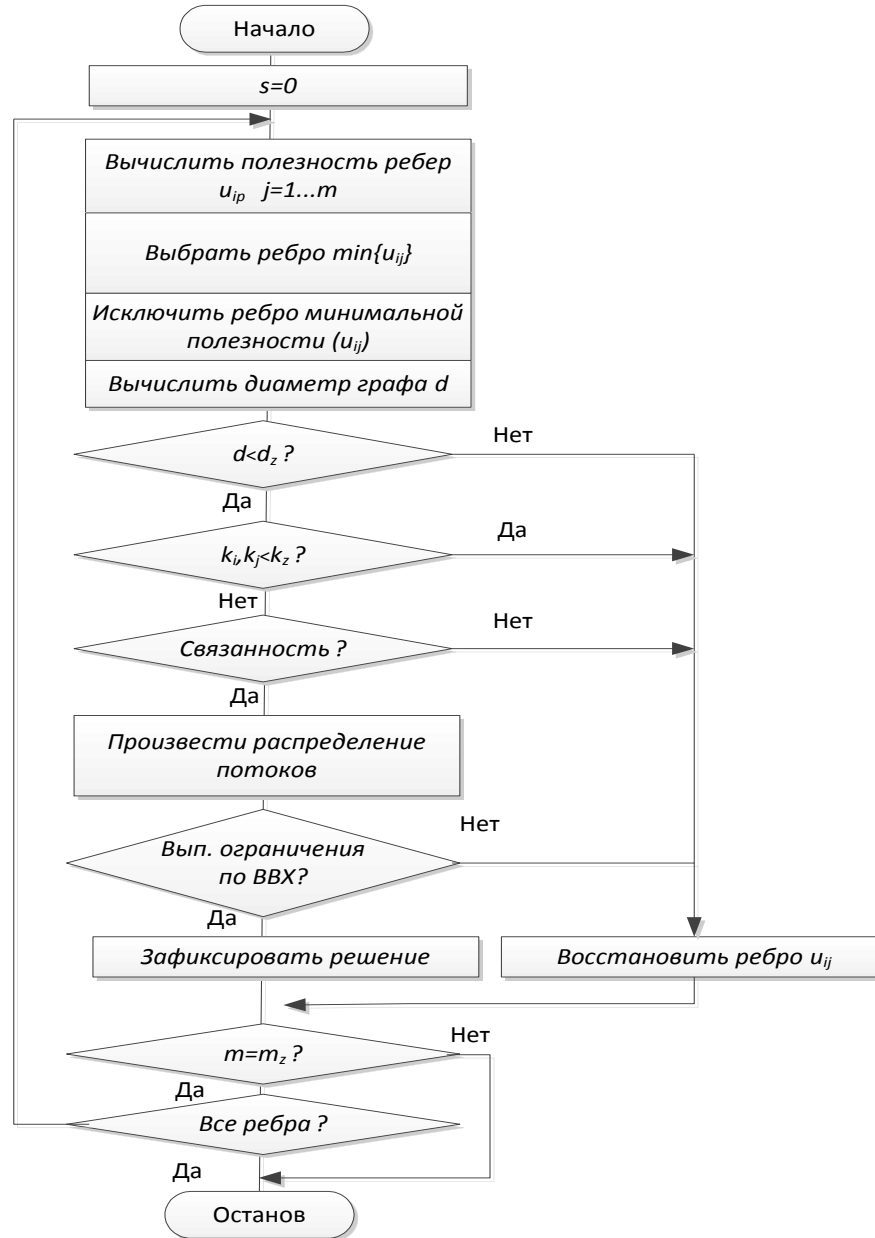
$$E = \sum_p \alpha_p \Lambda_p \bar{T}_p$$

$$\beta_s = \sum_{p=1}^P \sum_{j=1}^m L_{jp} \lambda_{jp}^{(s)}$$

$$P_{rs,k}^{(p)} = \frac{\lambda_{rs,k}^{(p)}}{\sum_{i=1}^{Ar} \lambda_{ri,k}^{(p)}}$$

# Оптимизация структуры сети

$$u_{ij} = \sum_{k=1}^K \alpha_{ij} y_{ij}^{(k)}$$





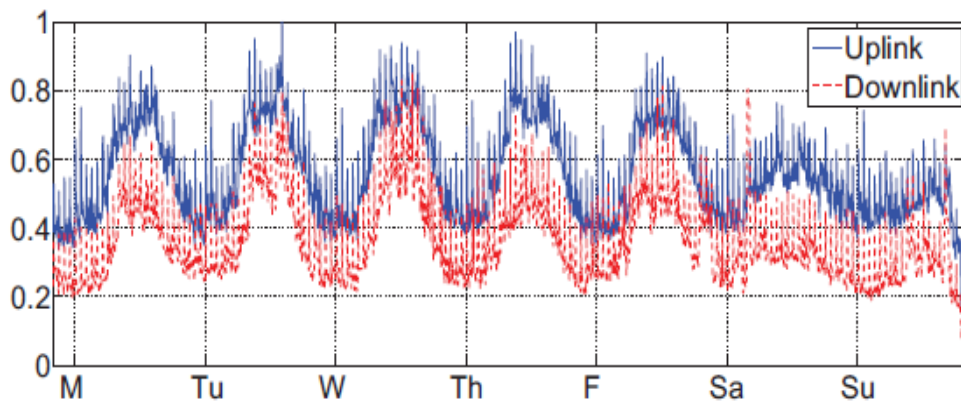
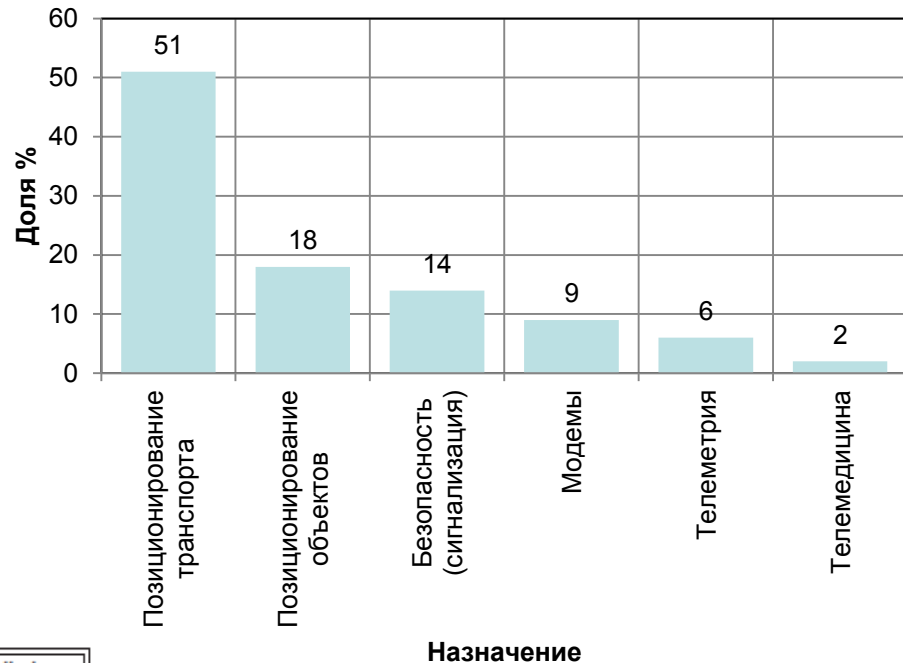
# Расписание управления трафиком

- Трафик M2M
- Пример
- Качество обслуживания
- Постановка задачи
- Модель расписания управления трафиком
- Задача оптимизации расписания
- Модель реакции трафика на расписание управления
- Пример оптимизации расписания управления
- Выводы

# Трафик M2M

## Источники трафика:

- системы мониторинга ТП
- системы позиционирования
- системы сигнализации
- телемедицина
- КП мобильных терминалов
- др.



## A First Look at Cellular Machine-to-Machine Traffic –Large Scale Measurement and Characterization

M. Zubair Shafiq† Lusheng Ji Alex X. Liu† Jeffrey Pang Jia Wang

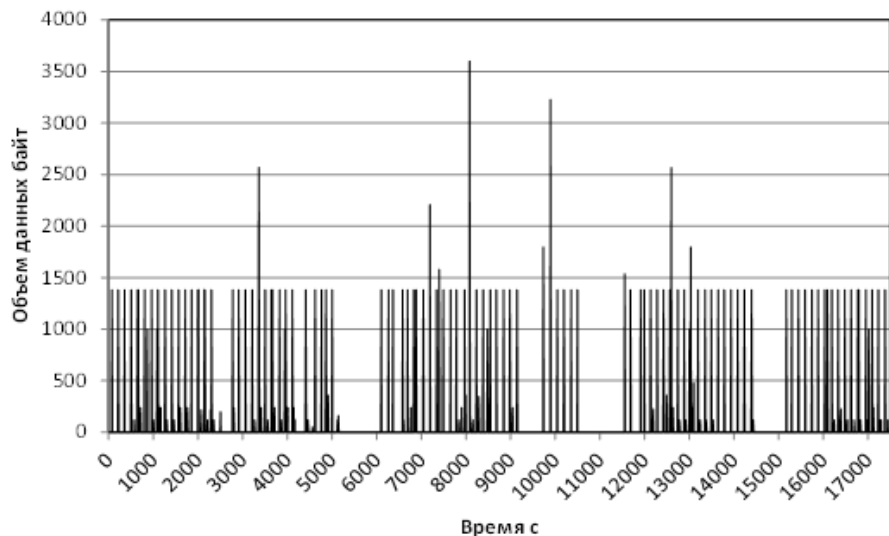
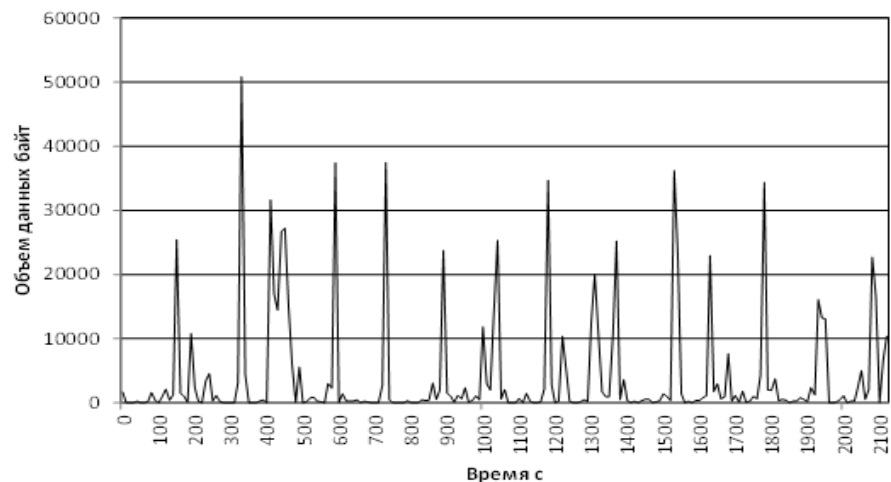
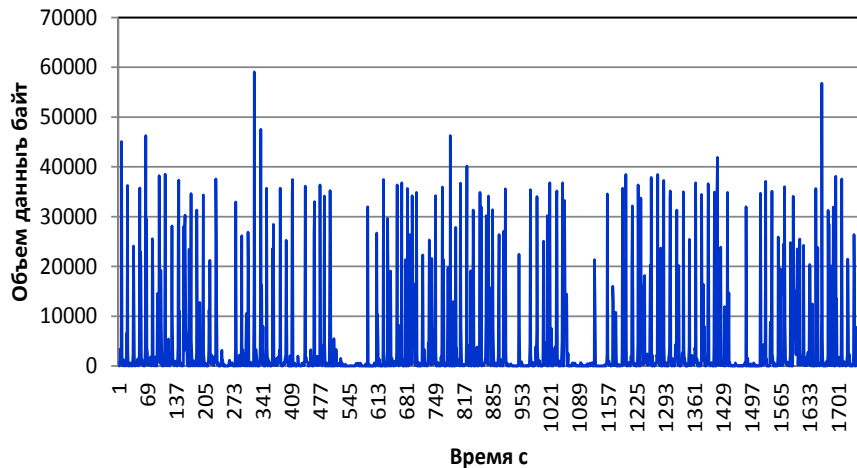
Department of Computer Science and Engineering,  
Michigan State University, East Lansing, MI, USA

AT&T Labs – Research, Florham Park, NJ, USA

{shafiqmu,alexliu}@cse.msu.edu,

{lji,jeffpang,jiawang}@research.att.com

# Пример M2M трафика мобильного терминала в состоянии покоя



Интенсивность 292,5 байт/с  
(1,7 пакетов/с)

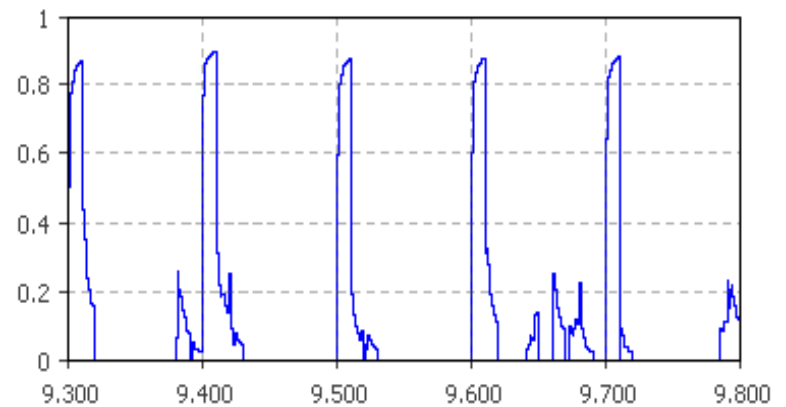
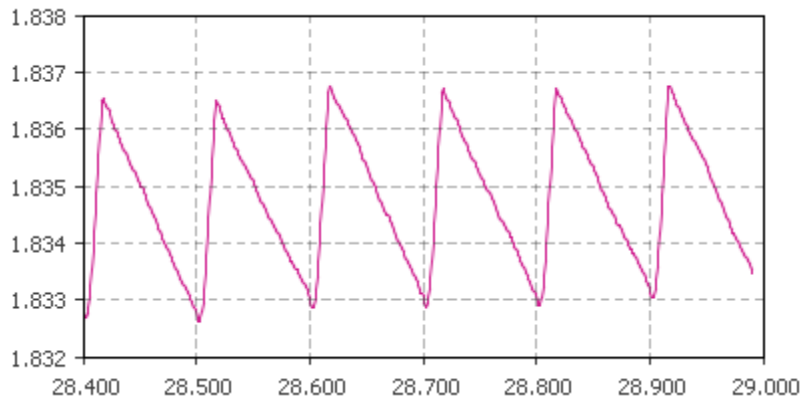
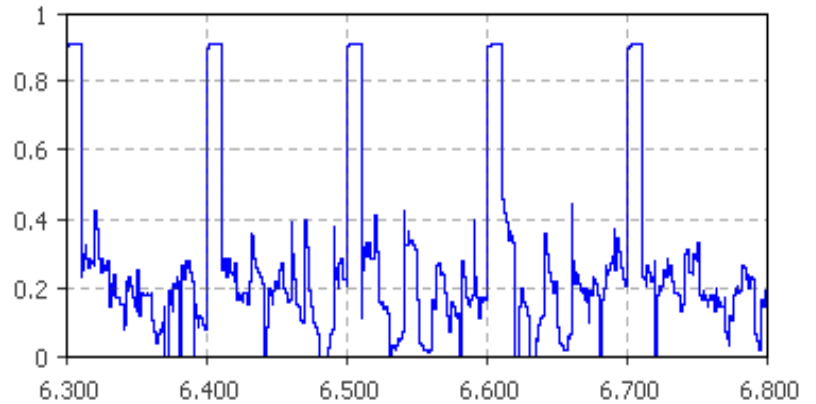
Средняя длина пакета 292,5 байт)

Коэффициент Херста 0,66.

(Продолжительность измерений  
около 5 часов)

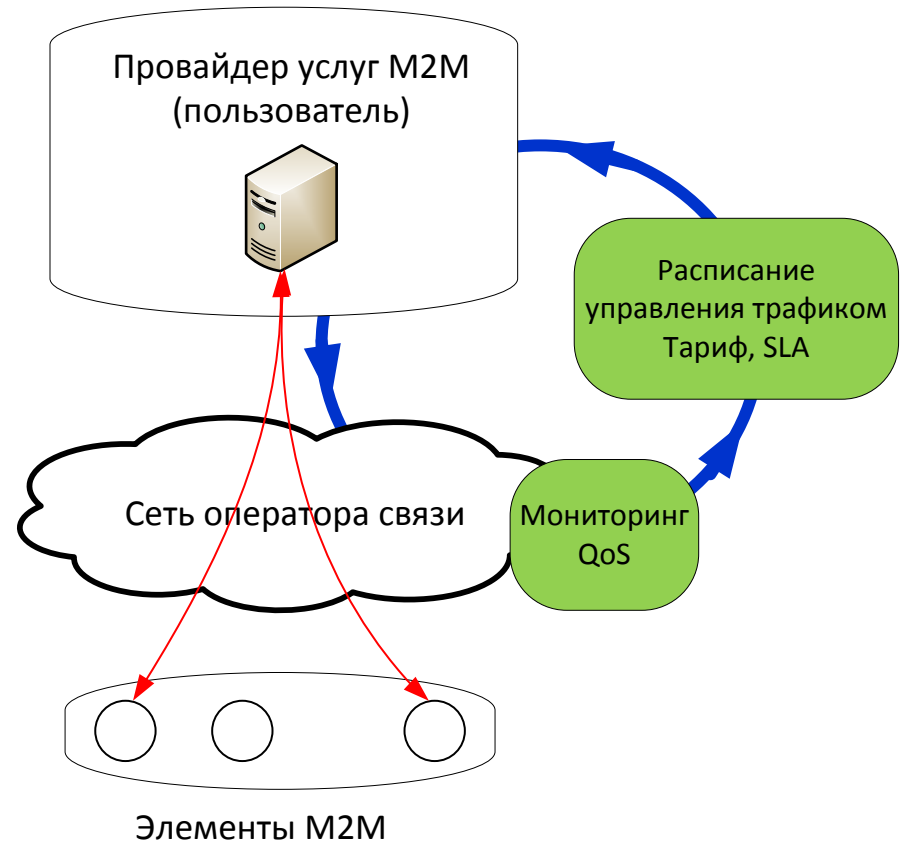
# Качество обслуживания

Пример моделирования влияния «периодического» трафика на вероятность потерь и задержку доставки пакетов.



# Постановка задачи

1. Выбрать метод построения расписания управления (тарифного расписания)
2. Построить модель реакции трафика на расписание управления.
3. Решить задачу оптимизации расписания с учетом реакции трафика.



# Модель расписания управления трафиком

$$T(t), \quad 0 \leq t \leq H \quad \bar{T} = \frac{1}{H} \int_0^H T(t) dt = const$$

Предполагаем, что величина тарифа (управляющего воздействия) влияет на поведение провайдера. Это приводит к тому, что M2M трафик, при введении тарифного расписания трансформируется.

$$a(t) \quad \rightarrow \quad a(t, T(\tau), 0 \leq \tau \leq H)$$

$a(t; T(\tau), 0 \leq \tau \leq H)$  - фактическая интенсивность нагрузки в момент  $t$ , учитывающая реакцию абонентов на тарифное расписание  $T(\tau)$  на протяжении суток.

Для характеристики влияния трафика на качество обслуживания введем гипотетическое обслуживающее устройство M/M/1/K.

Вероятности потерь определяется как

$$p = \frac{(1 - \rho(t)) \cdot \rho(t)^K}{1 - \rho(t)^{K+1}}$$

# Задача оптимизации расписания управления

Целевая функция 
$$M = \min \left\{ \frac{1}{\bar{a} \cdot H} \int_0^H a(t; T(\tau), 0 \leq \tau \leq H) \cdot p(t) dt \right\}$$

При ограничениях  $\bar{T} = const, 0 \leq T(t) \leq 1, \bar{a} = const, 0 \leq a(t) \leq a_{\max}$

Требуется найти неизвестную функцию  $T(t)$ . Это означает, что задача относится к не классическому вариационному исчислению или, в иной терминологии, к задачам оптимального управления, для решения которых, как правило, используется принцип максимума Л.С. Понтрягина.

Предлагается рассмотреть аппроксимацию исходной задачи, позволяющую свести ее к задаче математического программирования, где ищется не функция, а значения параметров управления.

Для этого вводится допущение:

-что ось времени дискретна (разделена на равные интервалы - кванты),

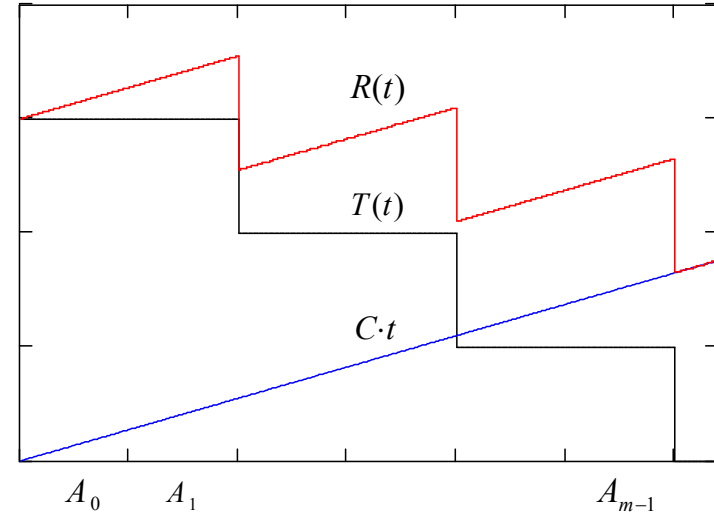
-искомая функция (расписание управления трафиком) является ступенчатой функцией времени, с конечным числом участков;

$$S = \min \left\{ \sum_{i=1}^m a_i(T_j, 1 \leq j \leq N) \cdot p_i \right\}$$

# Модель реакции трафика

Полагаем, что провайдер услуг может произвести сеанс обмена немедленно, или перенести сеанс на другое время. Тогда расходы провайдера будут определяться расходами на ожидание, которые линейно зависят от времени, и расходами на передачу данных, которые зависят от значения величины управления и объема передаваемых данных.

$$R(t) = T(t) \cdot \mathcal{G} + C \cdot t$$



Сеанс обмена может быть перенесен на  $N-1$  интервал управления, при этом расходы от переноса вызова на  $i$ -й интервал будут равны

$$R_i = C(A_i - A_0) + T_i \cdot \mathcal{G}; \quad 1 \leq i \leq N-1$$



## Условия переноса трафика

Необходимое условие для переноса  
сеанса

$$C(A_i - A_0) + T_i \cdot \mathcal{G} < T_0 \mathcal{G}; \quad 1 \leq i \leq N - 1$$

Достаточное условие для переноса  
сеанса

$$C(A_i - A_0) + T_i \cdot \mathcal{G} < C(A_j - A_0) + T_j \cdot \mathcal{G}; \quad 1 \leq i \leq N - 1$$

Необходимое и достаточное условия для переноса  
сеанса

$$\begin{cases} C(A_i - A_0) + T_i \mathcal{G} < T_0 \mathcal{G}, & 1 \leq i \leq N - 1 \\ C(A_i - A_0) + T_i \mathcal{G} < C(A_j - A_0) + T_j \mathcal{G}, & 1 \leq j \leq N - 1 \end{cases}$$

$$\max \left( \frac{T_i - T_{i+1}}{A_{i+1} - A_i}, \dots, \frac{T_i - T_m}{A_m - A_i} \right) \leq C < \min \left( \frac{T_0 - T_i}{A_i - A_0}, \dots, \frac{T_{i-1} - T_i}{A_i - A_{i-1}} \right)$$

$i \neq 0, i \neq m$

Если  $C \geq \max \left( \frac{T_0 - T_1}{A_1 - A_0}, \dots, \frac{T_0 - T_m}{A_m - A_0} \right)$ , то сеанс не переносится

## Описание стоимости времени

Для множества услуг стоимость единицы времени - это случайная величина, имеющая некоторое распределение вероятности  $S(x)$ .

$$P_{0i} = \begin{cases} 1 - S(\mu_0), & i = 0 \\ S(\nu_i) - S(\mu_i), & 0 < i < m \\ S(\nu_m), & i = m \end{cases}$$

$$\mu_i = \max\left(\frac{T_i - T_{i+1}}{A_{i+1} - A_i}, \dots, \frac{T_i - T_m}{A_m - A_i}\right), \quad 0 \leq i \leq m-1 \quad \nu_i = \min\left(\frac{T_0 - T_i}{A_i - A_0}, \dots, \frac{T_{i-1} - T_i}{A_i - A_{i-1}}\right), \quad 0 \leq i \leq m$$

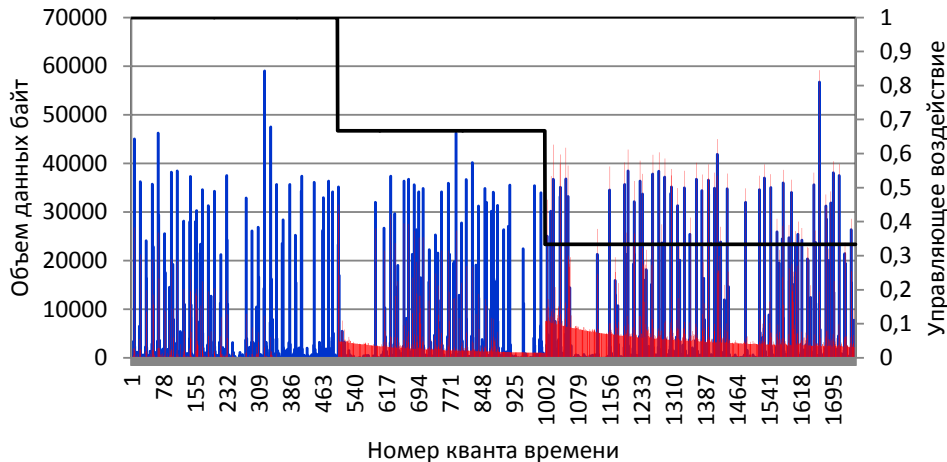
Тогда объем данных, переносимых с кванта  $k$  на начало  $i$ -го участка управления, можно определить следующим образом:

$$\mathcal{G}_{ki} = \mathcal{G}_k P_{ki}, \quad 0 \leq k \leq m-1, \quad 1 \leq i \leq N$$

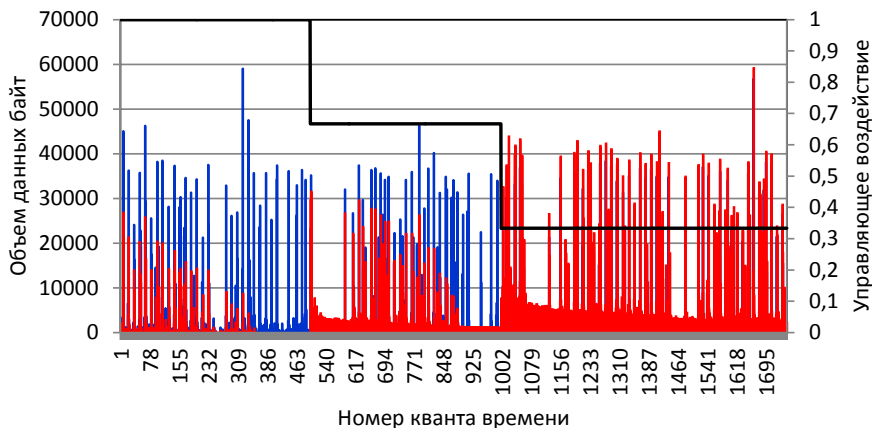
# Пример оптимизации расписания управления

Для решения задачи оптимизации необходимо выбрать положение и число участков управления и определить соответствующие им значения величины управления, которые приводили бы к минимизации суммы целевой функции.

Для сокращения объема вычислений можно использовать методы спуска (подъема) с вычислением координат градиента, позволяющие учесть все ограничения.

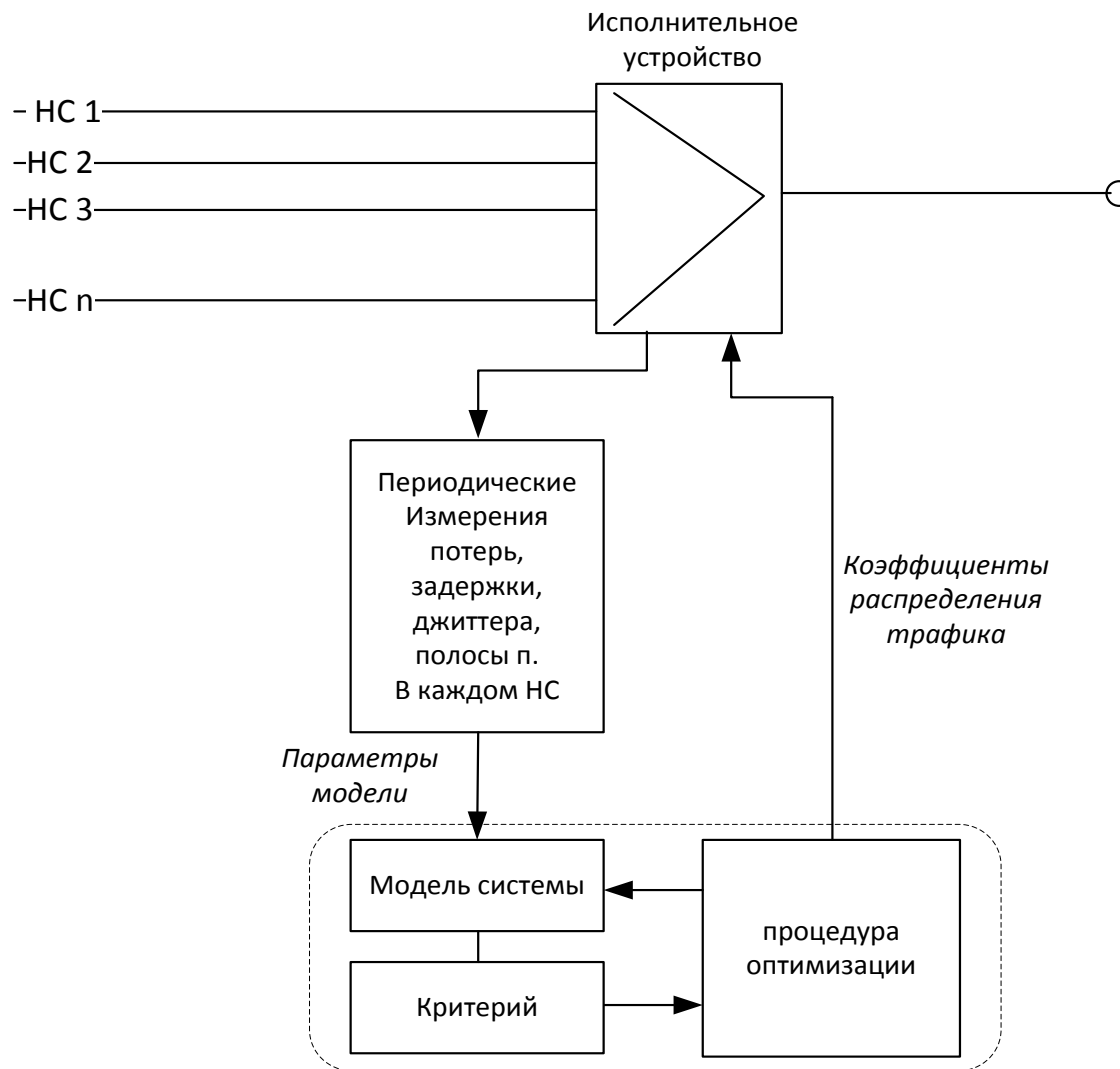


■ Исходный рафик      ■ Трансформированный трафик  
— Управляющее воздействие



■ Исходный рафик      ■ Трансформированный трафик  
— Управляющее воздействие

# Балансировка трафика

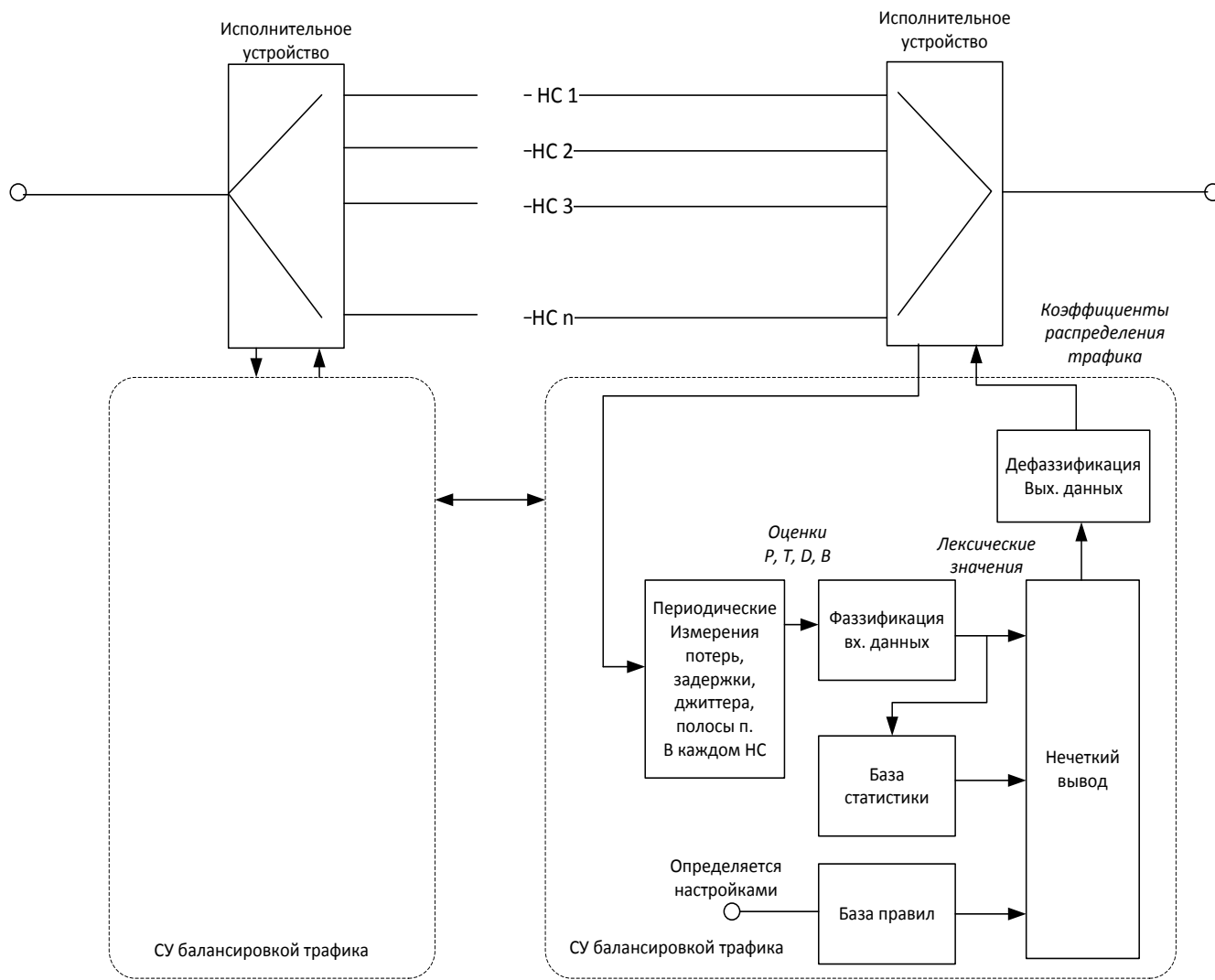


# Балансировка трафика

*Решение задачи оптимизации распределения ресурсов на базе модели системы.* В этом случае используется аналитическая модель системы, например описание НС в виде систем массового обслуживания и алгоритм оптимизации. Решение сводится к поиску значений параметров управления, например, коэффициентов распределения нагрузки, при которых достигается максимум (минимум) критерия эффективности. При этом алгоритм оптимизации вычисляет критерий на основе вычисленных параметров качества НС в соответствие с моделью системы (рис.1).

Достоинства данного подхода: возможность быстрого получения решения, отсутствие влияния на процесс обслуживания трафика (только затраты времени на вычисления).

# (нечеткие методы)



## (нечеткие методы)

Исходными данными для выработки управляющего воздействия являются оценки параметров НС, получаемые в результате периодических измерений, например, оценки потерь, задержки, джиттера, полосы пропускания.

Фаззификация предполагает преобразование (с помощью функций принадлежности) численных оценок параметров в значения, так называемых, лексических переменных, например, «хороший», «средний», «плохой» (набор возможных значений выбирается при разработке). Эти данные сохраняются в базе статистики и поступают на блок нечеткого вывода, который на основе определенного набора правил, а также на основе статистики о предшествующих измерениях вырабатывает решение (выбирает стратегию управления), которая также представлена набором лексических переменных.

В результате дефаззификации определяются численные значения управляющего воздействия, например, коэффициенты распределения трафика.

---

## Выводы

- Задачи оптимизации решаются при комплексном использовании теоретических методов моделирования сетей связи и методов оптимизации.
- Для постановки задачи оптимизации необходима формулировка цели и определение возможностей управления параметрами системы;
- В зависимости от решаемой задачи могут быть применены аналитические или численные методы оптимизации.
- В целях упрощения задачи следует стремиться к тому чтобы целевая функция была, решаемой аналитически. Если это невозможно, то следует пытаться упростить ее, например, построив в виде выпуклой функции.



---

## Литература

### ОСНОВНАЯ ЛИТЕРАТУРА

- Я.С. Дымарский Методы и алгоритмы оптимизации сетей связи. СПб ГУТ, 2005.
- Я.С. Дымарский Задачи и методы оптимизации сетей связи. Методические указания и контрольные задания.
- В.В. Лохмотко, К.И. Пирогов Анализ и оптимизация цифровых сетей интегрального обслуживания. Минск, «Наука и техника», 1991 г.

### ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

- О.И. Шелухин, А.М. Тенякшев, А.В. Осин Фрактальные процессы в телекоммуникациях. М. «Радиотехника», 2003.
- Д. Бертсекас, Р.Галлагер Сети передачи данных.
- Б.Банди Методы оптимизации. Вводный курс. М. Радио и связь. 1988 г.
- Р.Беллман, С.Дрейфус Прикладные задачи динамического программирования. М., Наука, 1965 г.
- Е.С. Вентцель Элементы динамического программирования. М., Наука, 1961 г.
- Н. Кристофидес Теория графов. Алгоритмический подход. М. «Мир», 1978 г.