Математическое моделирование радиотехнических устройств и систем

(ММРУС)

Парамонов Александр Иванович

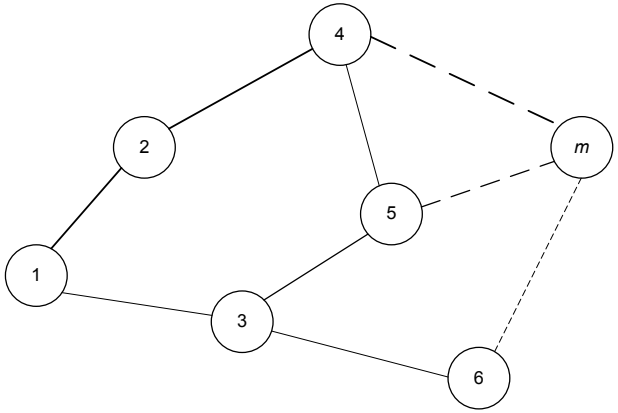
alex-in-spb@yandex.ru

Содержание

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.

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

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

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

Целевая функция

$$x_j = \arg\min_{x_j} \left\{ \overline{p}\left(x_j\right) \right\} \qquad \text{при} \qquad \sum_{j=1}^{n} x_j = N$$

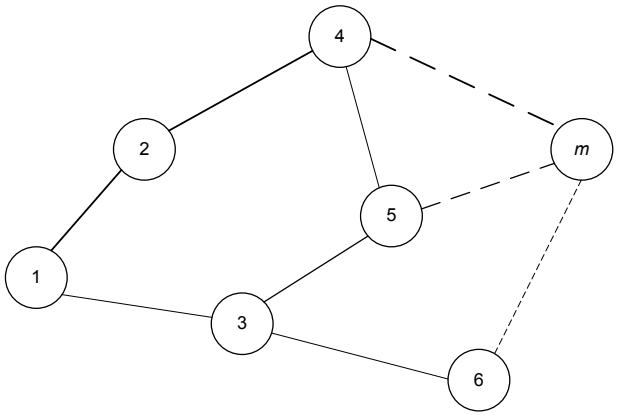
5. Выбор метода оптимизации целевой функции

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* направлений связи

w_j Надежность канала в *j* направлений связи (вероятность исправного состояния

v_j Значимость *j* направления связи

$$v_j = 0 \ldots 1 \qquad \sum_{j=1}^{n} v_j = 1$$

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

$$W_j = 1 - \left(1 - w_j\right)^{x_j}; \qquad j = 1 \ldots n$$

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

$$\overline{w} = \sum_{j=1}^{n} v_j \cdot W_j; \qquad j = 1 \ldots n \qquad \text{где} \qquad \sum_{j=1}^{n} v_j = 1$$

Целевая функция

$$x_j = \arg\max_{x_j} \left\{ \overline{w}\left(x_j\right) \right\} \qquad \text{при} \qquad \sum_{j=1}^{n} x_j = N$$

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 $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ является **абсолютным** (**глобальным**) экстремумом *f(x)*, если $f(x^*) \le (\ge) f(x) \quad для \quad \forall x$

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

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

5.2 Безусловная оптимизация

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

1. Необходимые условия существование 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.

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

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.

$$a_{ik} = \begin{Vmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{Vmatrix}$$ Гессиан *f(x)*

Собственные значения можно найти как корни уравнения

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

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 $f(x) = f(x_1, x_2, \ldots x_n)$

требуется найти экстремумы *f(x)* при заданных ограничениях:

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

1. Необходимые условия существование локального экстремума (НУ)

Функция Лагранжа

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

ψ_i множители Лагранжа

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

такой, что $\Phi'_x(x^*, \psi^*) = 0$

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

1. Достаточные условия существование 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

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

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

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

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

$$f(t \cdot x + (1-t) \cdot y) \le t \cdot f(x) + (1-t) \cdot f(y)$$

$$t \in 0 \ldots 1 \quad x \ne y \qquad\qquad n$$

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 линейна 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 $f(x) = f(x_1, x_2, \ldots x_n)$

требуется найти экстремумы *f(x)* при заданных ограничениях:

$$\varphi_j(x) = \varphi_j(x_1, x_2, \ldots x_n) \le 0; \qquad x \ge 0; \qquad j = 1 \ldots m$$

$\varphi_j(x)$ выпуклые функции

Необходимые и достаточные условия существование локального экстремума

Функция Лагранжа

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

ψ_i множители Лагранжа

1) $\dfrac{\partial \Phi(x, \psi)}{\partial x_i} \ge 0$

2) $\dfrac{\partial \Phi(x, \psi)}{\partial x_i} x_i = 0$

3) $x_i \ge 0$

$i = 1 \ldots n$

4) $\dfrac{\partial \Phi(x, \psi)}{\partial \psi_j} \ge 0$

5) $\dfrac{\partial \Phi(x, \psi)}{\partial \psi_j} x_i = 0$

6) $\psi_j \ge 0$

$j = 1 \ldots m$

6. Численные методы оптимизации

Безусловная оптимизация

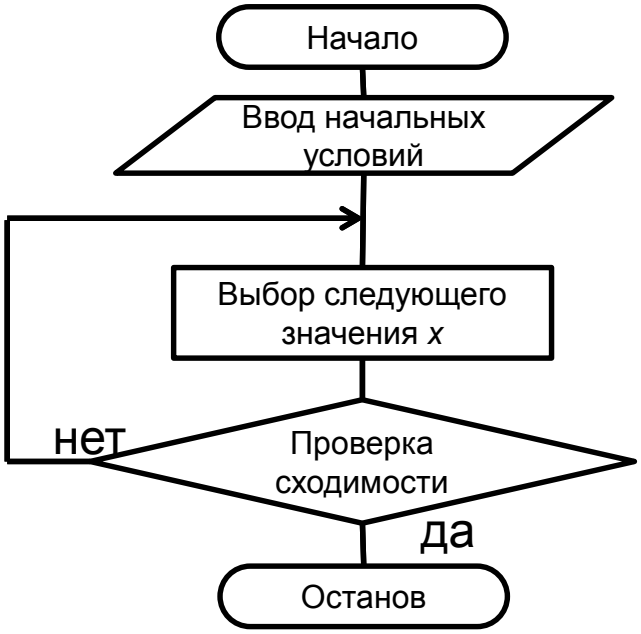
-функции одной переменной

-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*

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

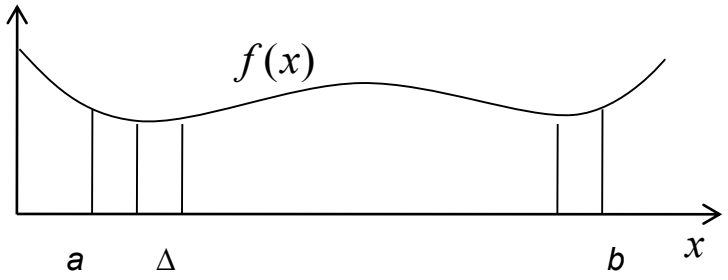
$\frac{\Delta}{2}$ – *абсолютная погрешность*

Требуется найти экстремум

$x^* = \arg\min_x \{f(x)\}$ *или*

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

2. Поиск экстремума



$n = \frac{b-a}{\Delta}$ $\qquad x^* = \arg\min_{x_i} = \{f(x_i)\}, \quad i = 1 \ldots n$

6.2.2 Дихотомия

Заданы

Функция 1 *переменной*

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

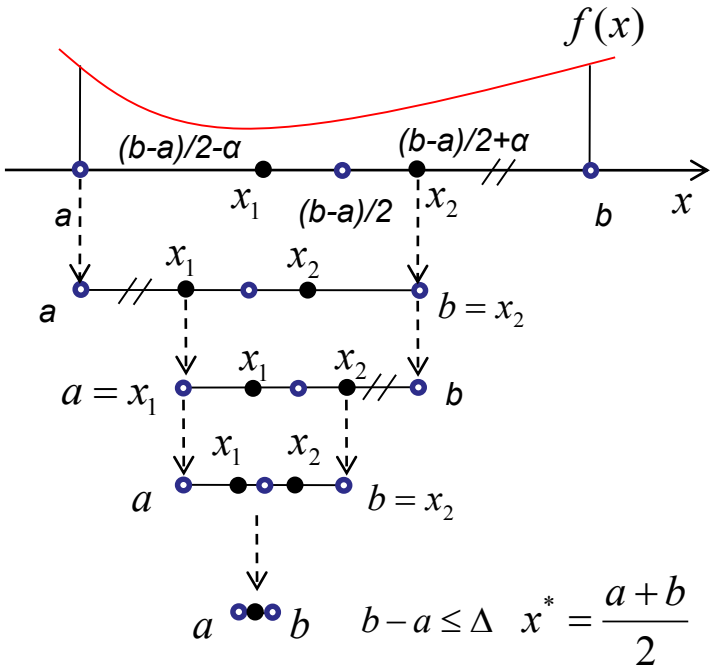
$\frac{\Delta}{2}$ – *абсолютная погрешность*

Требуется найти экстремум

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

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

2. Поиск экстремума



-Алгоритм поиска представляет собой много итерационную процедуру
-На каждой итерации интервал поиска экстремума сужается путем исключения правого (*x2*, *b*) или левого отрезка (*a, x1*)
-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*|>Δ
-результат *x**=(*a*+*b*)/2

6.2.3 Метод золотого сечения

Заданы

Функция 1 *переменной*

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

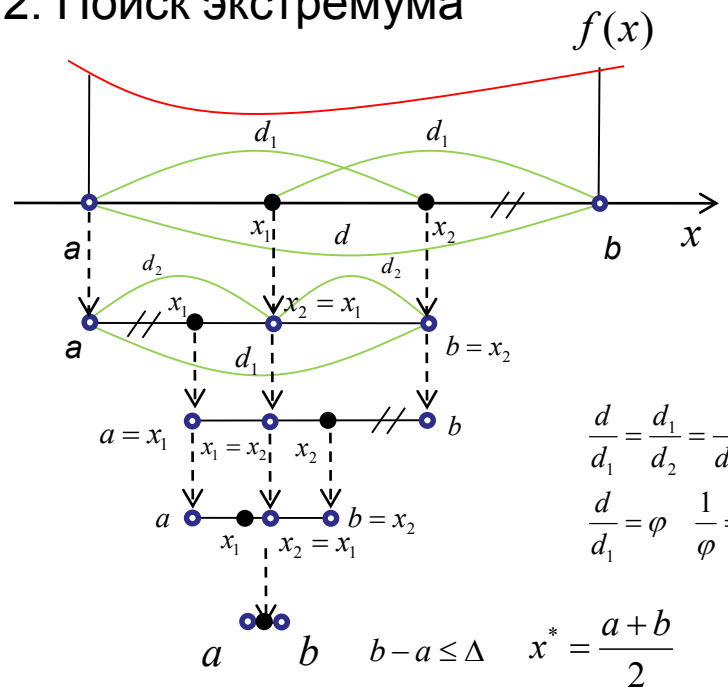
$\frac{\Delta}{2}$ – *абсолютная погрешность*

Требуется найти экстремум

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

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

2. Поиск экстремума



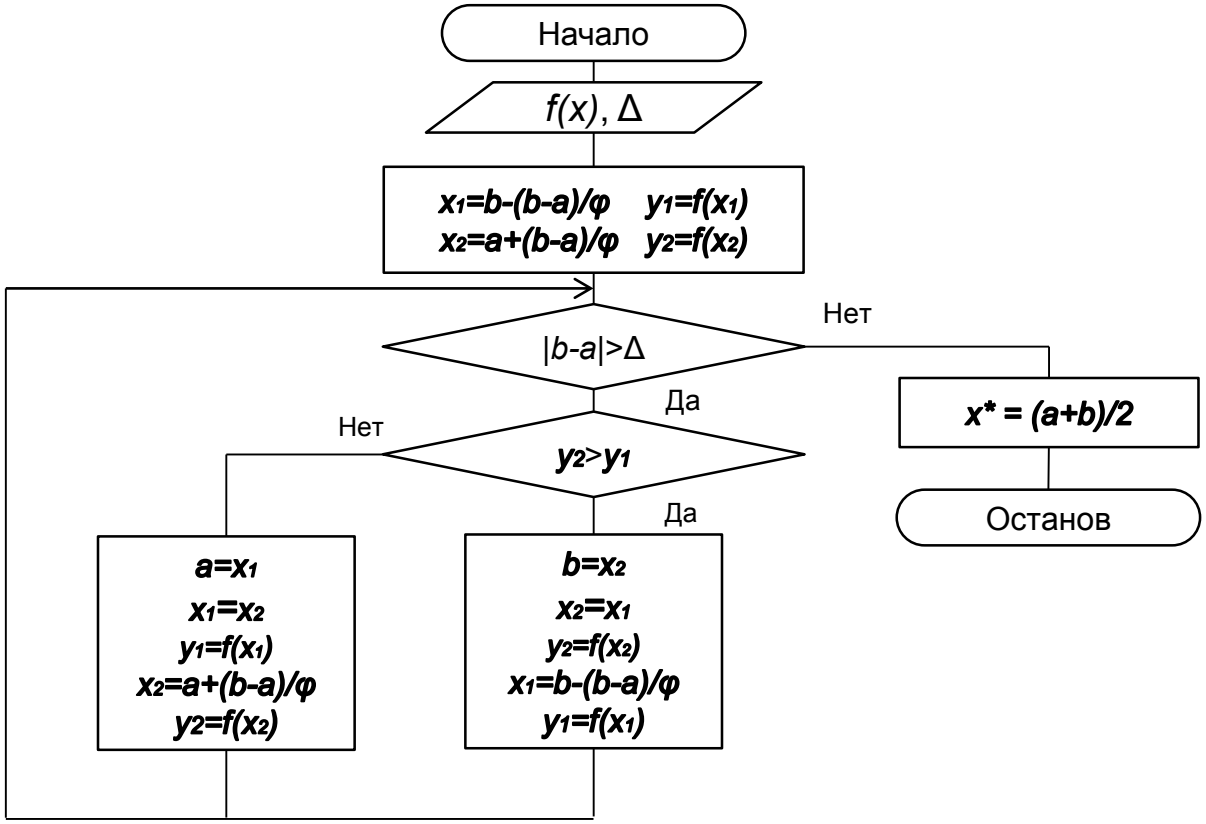
$b - a \le \Delta \qquad x^* = \frac{a+b}{2}$

-Алгоритм поиска представляет собой много итерационную процедуру
-На каждой итерации интервал поиска экстремума сужается путем исключения правого (*x*2, *b*) или левого отрезка (*a, x1*)
-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*|>Δ
-результат *x**=(*a*+*b*)/2

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

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

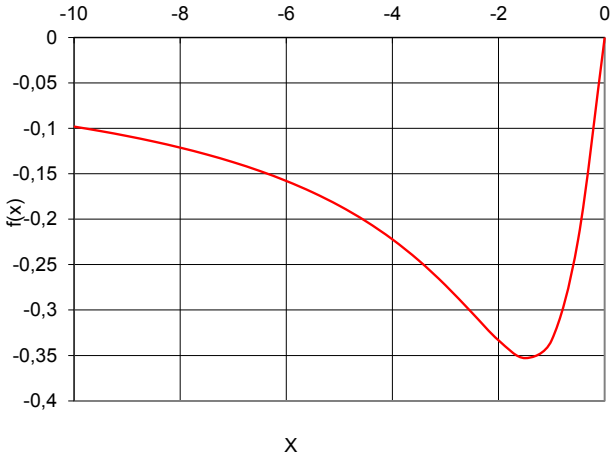
Метод золотого сечения (алгоритм)



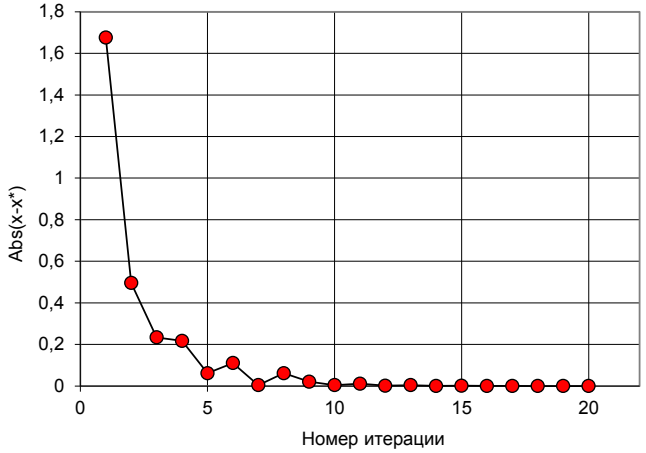
Метод золотого сечения (пример)

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

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*

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

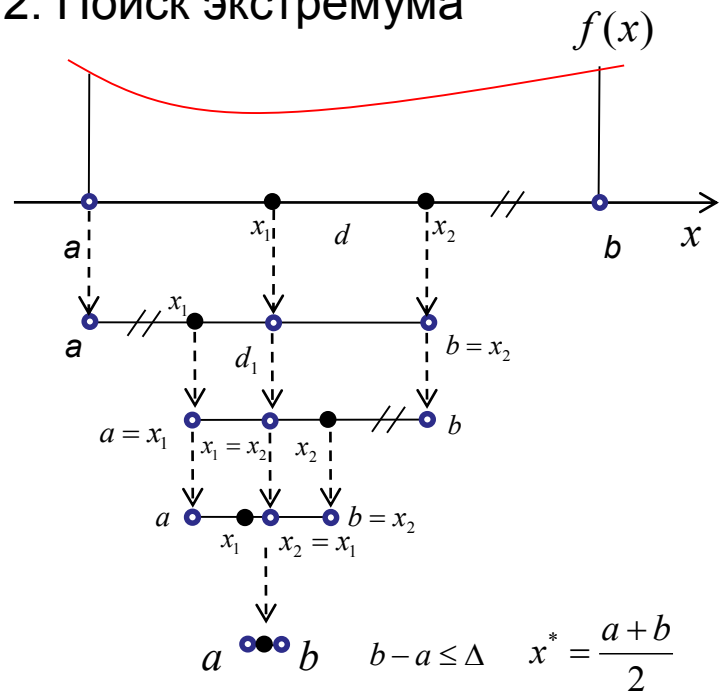
n – *число итераций*

Требуется найти экстремум

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

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

2. Поиск экстремума

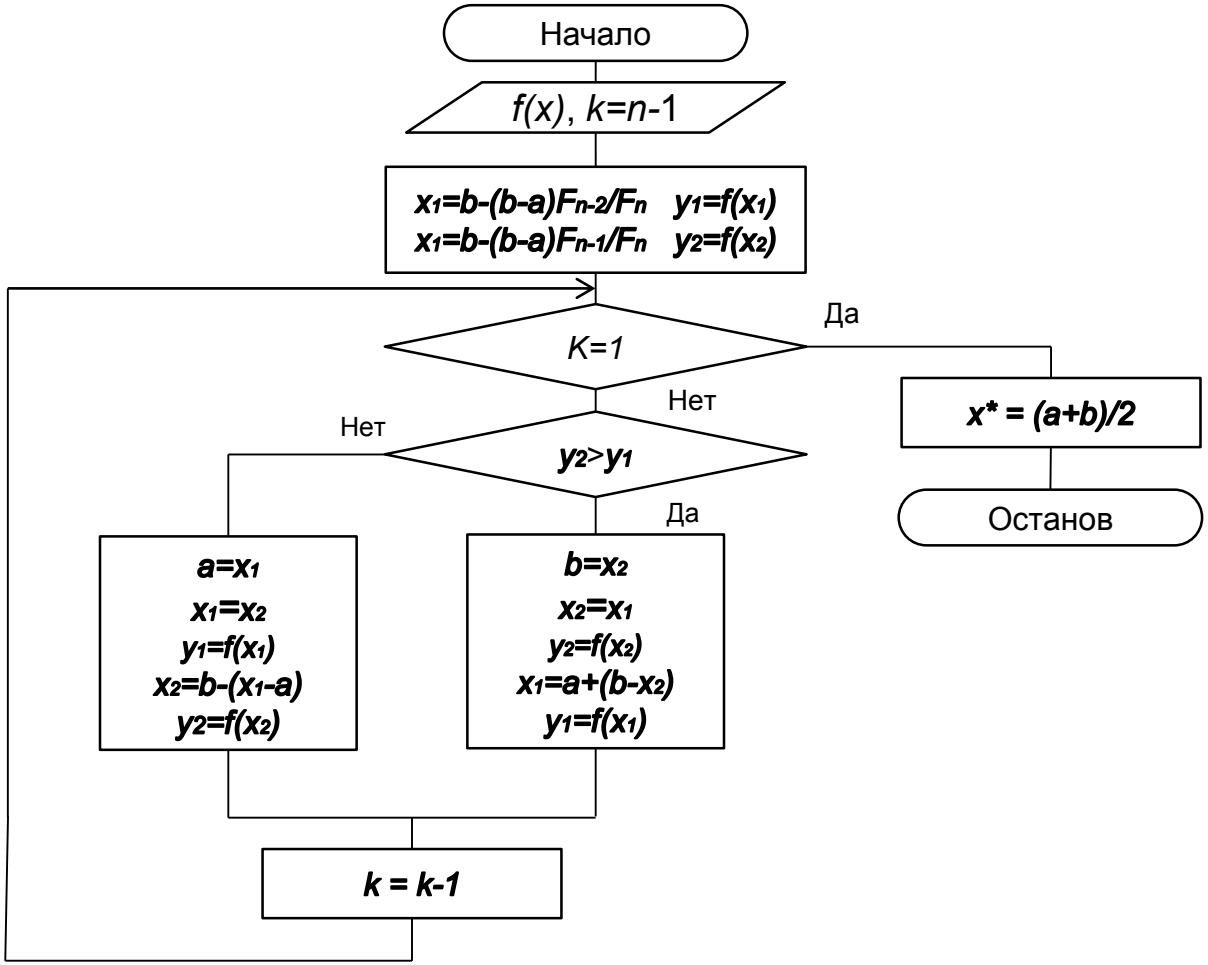


-Алгоритм поиска представляет собой много итерационную процедуру
-На каждой итерации интервал поиска экстремума сужается путем исключения правого (*x*2, *b*) или левого отрезка (*a, x1*)
-Правый отрезок исключается если *f*(*x*2)>*f*(*x*1)
-Левый отрезок исключается если *f*(*x*1)>*f*(*x*2)
-Итерации повторяются пока не выполнено заданное число итераций
-результат *x**=(*a*+*b*)/2

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

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

Метод чисел Фибоначчи (алгоритм)



6.2.5 Метод квадратичной интерполяции

Заданы

Функция 1 *переменной*

$f(x)$ $h-$*величина* *шага*

Требуется найти экстремум

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

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

-Метод относится к группе методов аппроксимации

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

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

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

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

Коэффициенты *a, b, c* могут быть вычислены, если известны значения функции в 3 точках, например α, β, γ.

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

$$\begin{cases} a \cdot \alpha^2 + b \cdot \alpha + c = f_\alpha \\ a \cdot \beta^2 + b \cdot \beta + c = f_\beta \\ a \cdot \gamma^2 + b \cdot \gamma + c = f_\gamma \end{cases} \Rightarrow$$

$$a = \left[(\gamma-\beta)f_\alpha + (\alpha-\gamma)f_\beta + (\beta-\alpha)f_\gamma\right]/\Delta$$

$$b = \left[(\beta^2-\gamma^2)f_\alpha + (\gamma^2-\alpha^2)f_\beta + (\alpha^2-\beta^2)f_\gamma\right]/\Delta$$

$$c = \left[\beta\gamma(\gamma-\beta)f_\alpha + \gamma\alpha(\alpha-\gamma)f_\beta + \alpha\beta(\beta-\alpha)f_\gamma\right]/\Delta$$

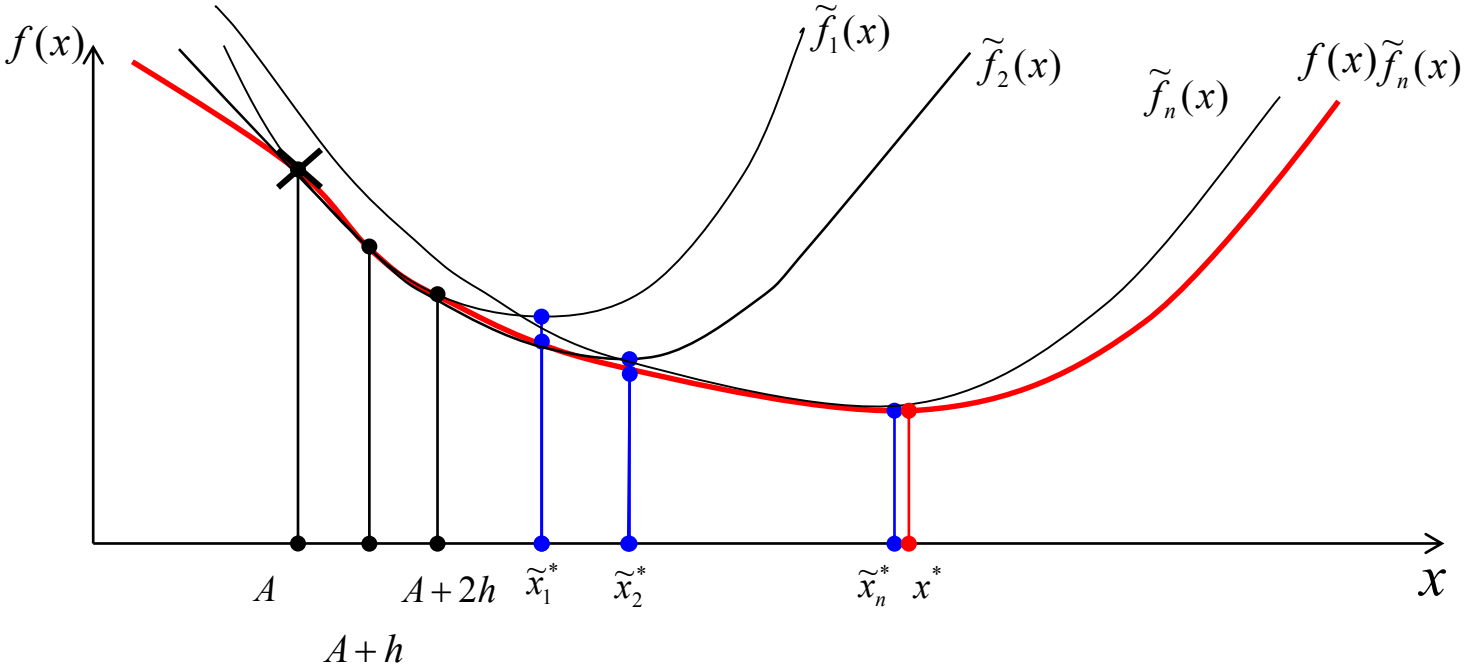
$$\Delta = (\alpha-\beta)(\beta-\gamma)(\gamma-\alpha)$$

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

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.

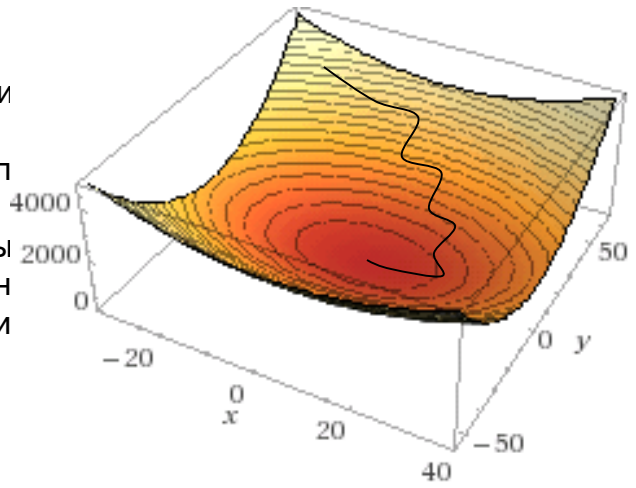


6.3 Оптимизация функция нескольких переменных
6.3.1 Покоординатный спуск

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

Данный метод предполагает следующие шаги:

1. Фиксируются все значения переменных кроме одной
2. Производится поиск экстремума по одной переменной одним и известных 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.



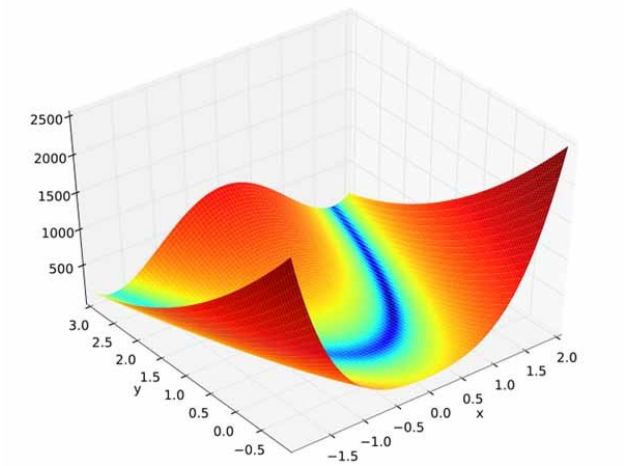
Данный метод может быть эффективен только при оптимизации относительно «простых» функций

Пример графика функции получен с помощью сервиса www.wolframalpha.com

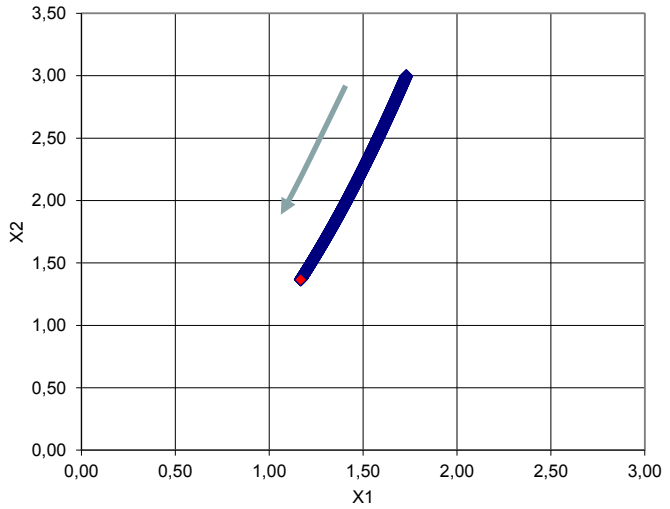
Покоординатный спуск (пример)

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)

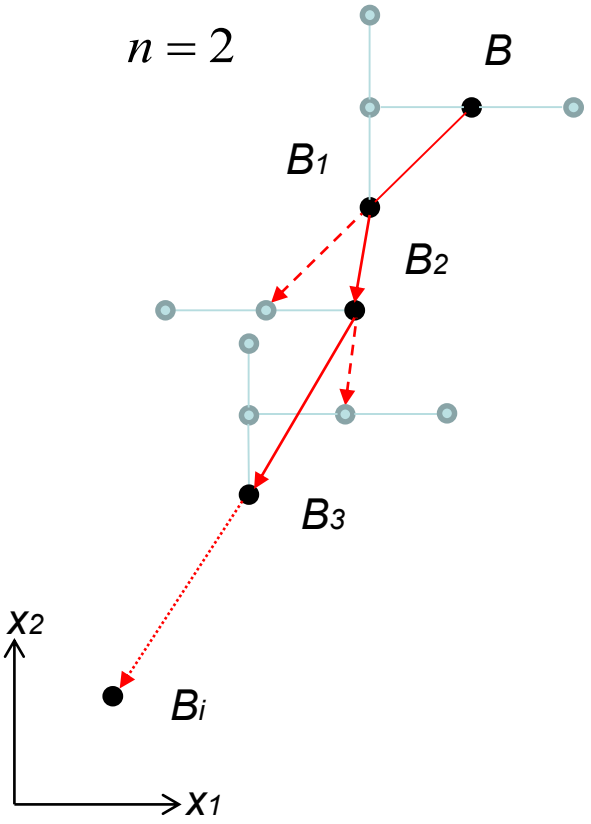


Результат x*=(1,16; 1,36), потребовалось около 1138x20 вычислений функции

6.3.2 Метод Хука-Дживса (поиск по образцу)

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

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 *Bi.*

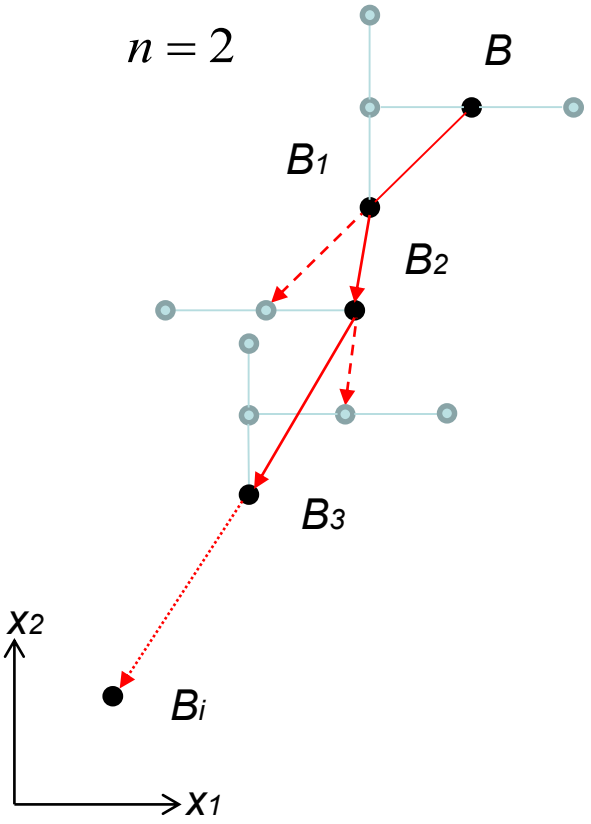
Поиск по образцу:
После нахождения базисной точки *Bi* производится поиск по образцу, который заключается в том, что в направлении от *Bi-1* к *Bi* делается шаг из точки *Bi* величиной, например, равной расстоянию между *Bi-1* и *Bi* .

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 $f(x) = f(x_1, x_2, \ldots x_n)$

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 *Bi.*

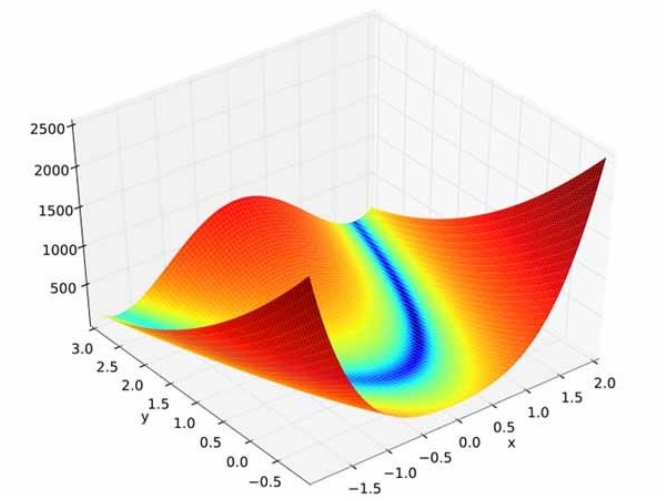
Поиск по образцу:
После нахождения базисной точки *Bi* производится поиск по образцу, который заключается в том, что в направлении от *Bi-1* к *Bi* делается шаг из точки *Bi* величиной, например, равной расстоянию между *Bi-1* и *Bi* .

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.

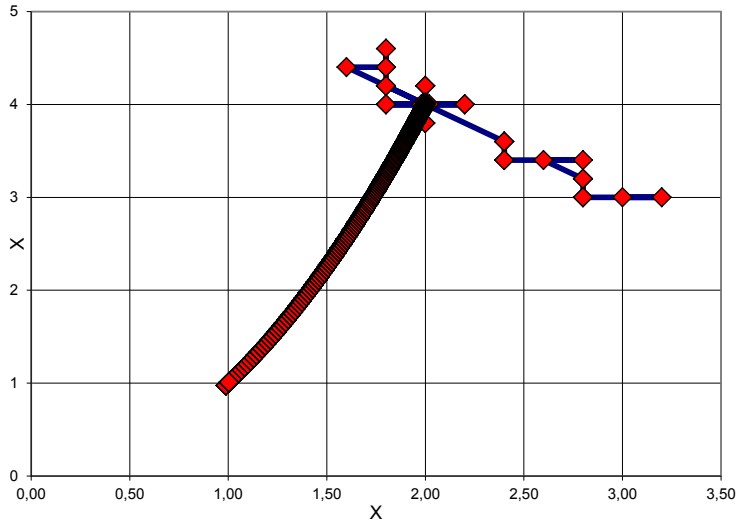
Метод Хука-Дживса (пример)

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)



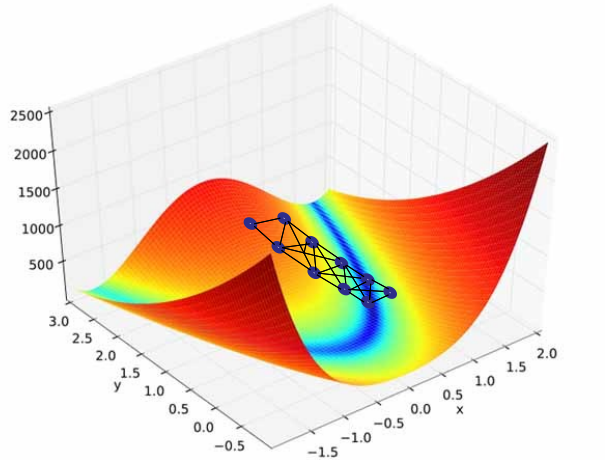
Результат (1,004; 1,008); 1261 вычисления функции

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

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

Этапы метод предполагает следующие шаги:

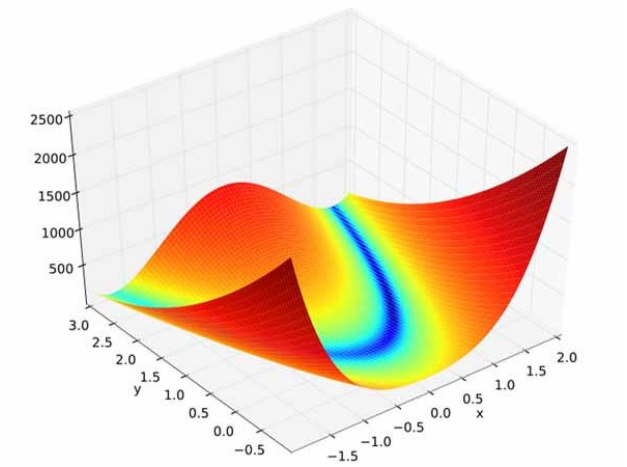
1. *Подготовка*. Задается исходный многогранник, содержащий n+1 вершину (т.е. *n*+1 точка - симплекс)
2. *Оценка*. Вычисляются значения функции в вершинах многогранника. Находят «худшую» H, «лучшую» L, и предшествующую «худшей» (по величине) вершину K.
3. *Отражение*. Относительно «наихудшей» точки определяется 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.



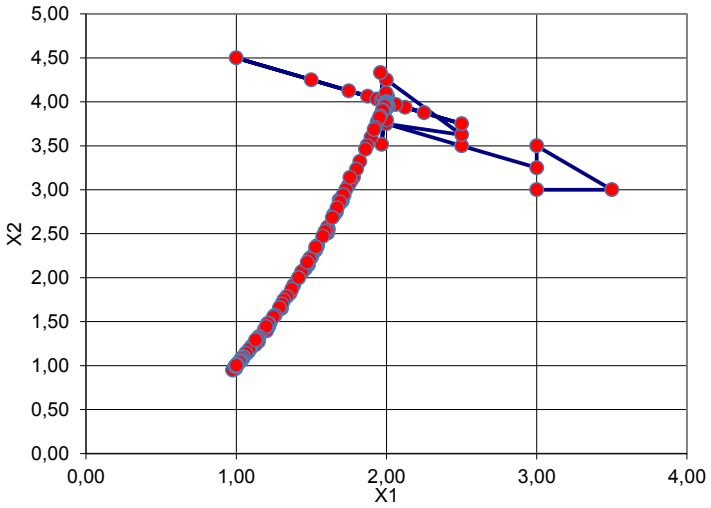
Симплекс метод Нелдера-Мида (пример)

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 n $f(x) = f(x_1, x_2, \ldots x_n)$ и ограничения в виде явных и неявных ограничений

$$l_i \le x_i \le u_i \quad i = 1 \ldots n \qquad g_j(x) \le b_j \quad j = 1 \ldots m$$

(Данный метод представляет собой модификацию симплекс метода Нелдера-Мида, в отличии 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_H.
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_H 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*.

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

6. Условие сходимости достигается, если σ и *dm* меньше некоторых заданных величин.

6.3.5 Метод штрафных функций (Условная оптимизация)

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

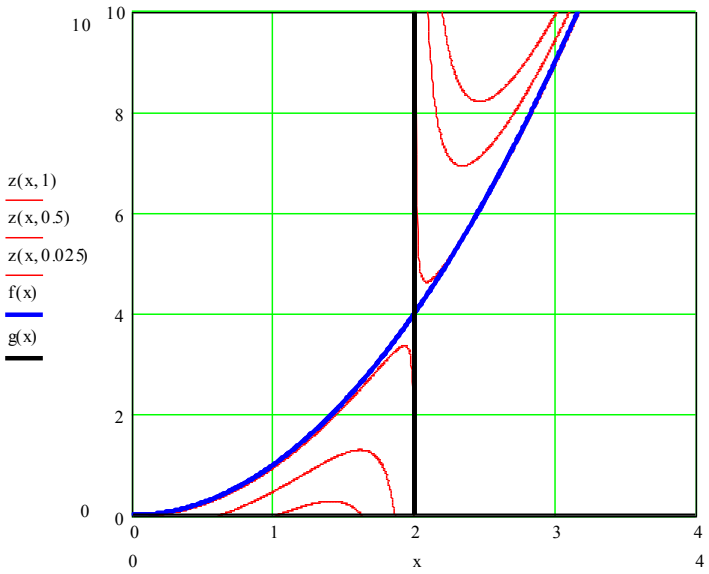
Метод заключается в замене исследуемой функции некоторой модифицированной функцией, которая в области допустимых значений близка к исходной функции, а вблизи области ограничений ее значение резко увеличивается.

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

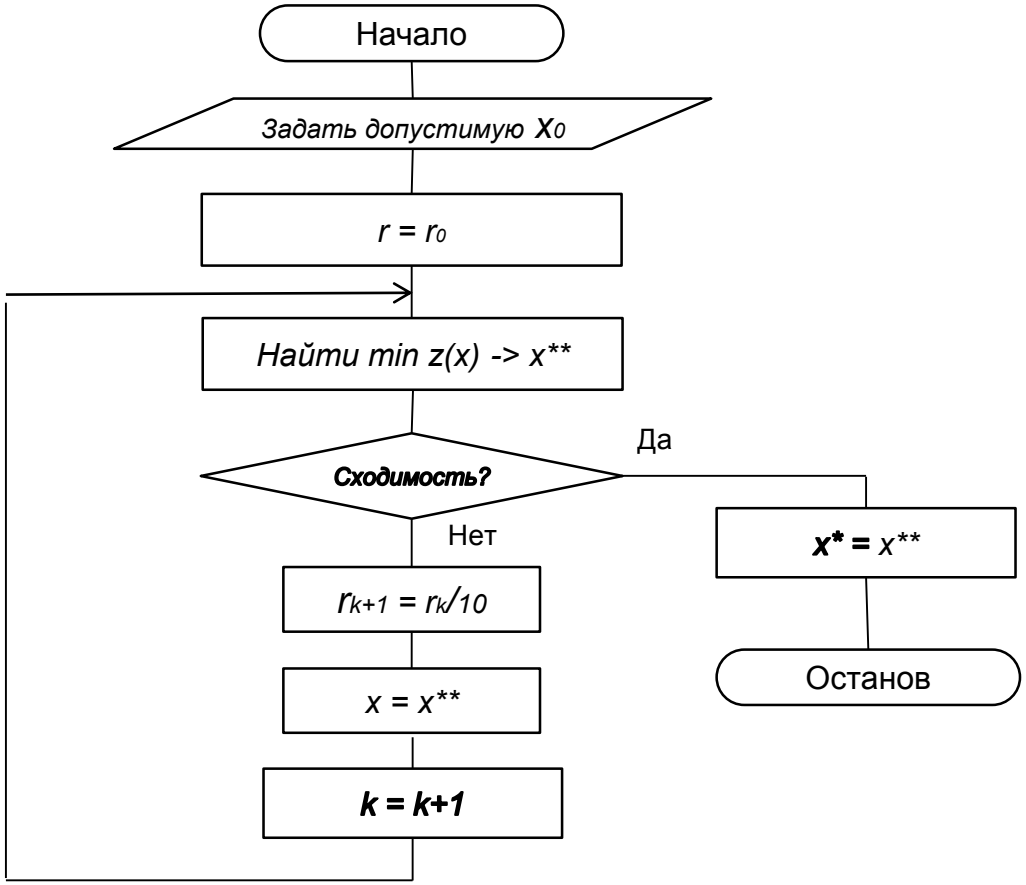
Например, $P(x) = r\sum_{j=1}^{m} \frac{1}{C_j(x)}; \quad r > 0 \Rightarrow \quad z(x,r) = f(x) + r\sum_{j=1}^{m} \frac{1}{C_j(x)};$

Пример.

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



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

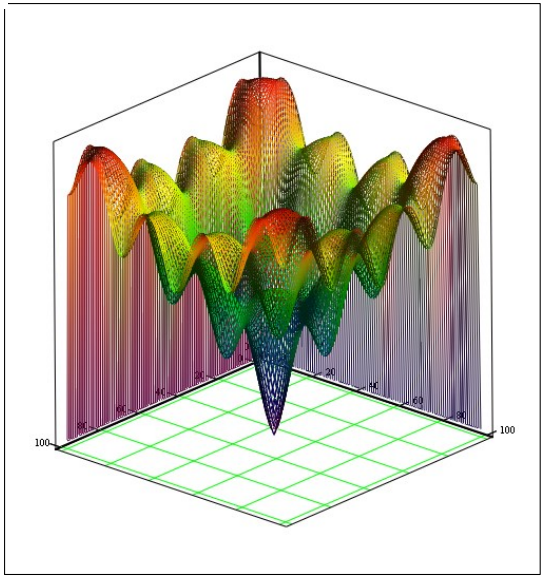
Целью разработки и использования данных методов является оптимизация невыпуклых функций.
Общего метода численной оптимизации невыпуклых функций не существует. Существуют лишь подходы, позволяющие на основании имеющихся сведений об оптимизируемой функции использовать те или иные методы, позволяющие с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

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

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).



Пример: функция Эккли

6.4.1 Слепой случайный поиск

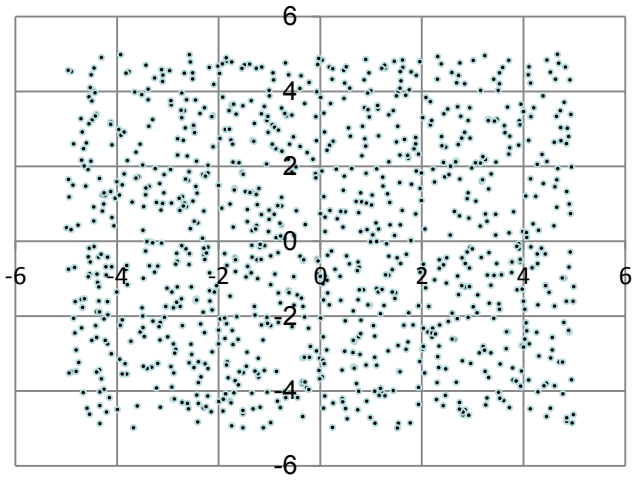
Пусть задана невыпуклая функция нескольких n переменных $f(x) = f(x_1, x_2, \ldots x_n)$

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 Δ - заданное значение абсолютной погрешности

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



Пример: Слепой случайный поиск, n=2

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

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

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.

6.4.2 Эволюционный метод (генетический алгоритм)

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

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 производится 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:

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

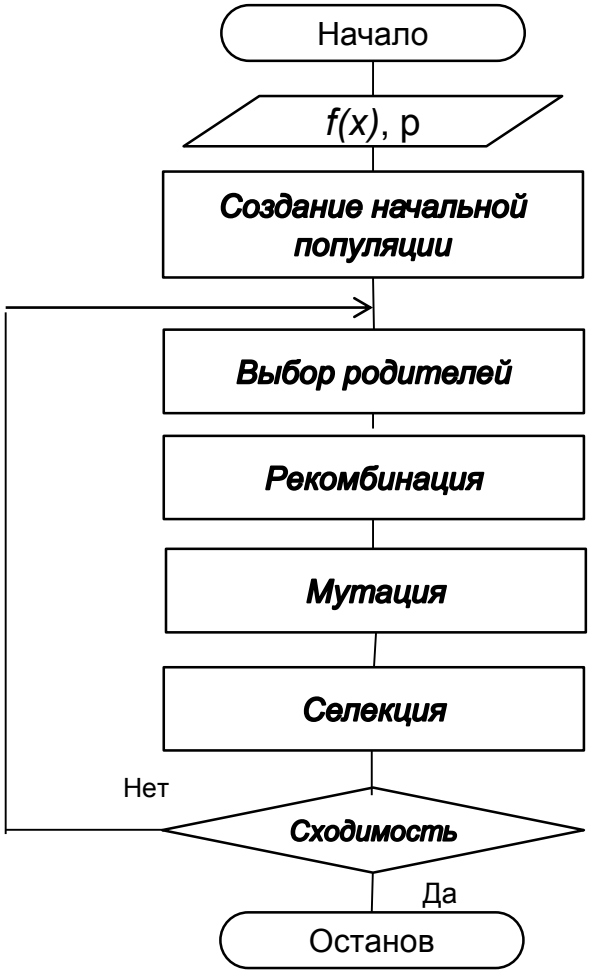
Шаг 4. **Мутация**. Случайное изменение наследственных признаков. Например: $x_i = \zeta \cdot x_i$

где ζ 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.

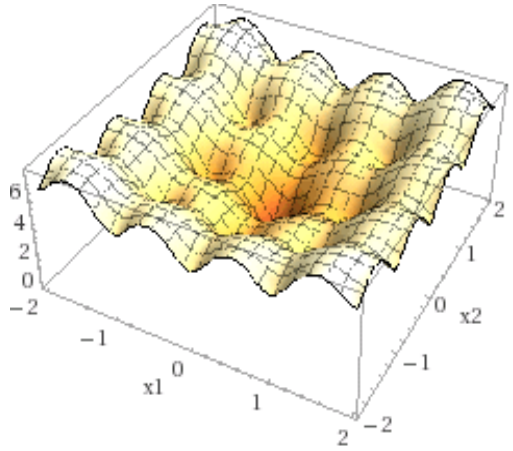
Генетический алгоритм

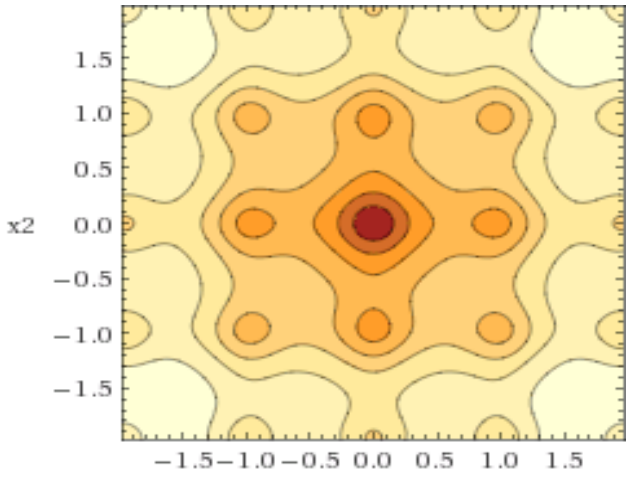


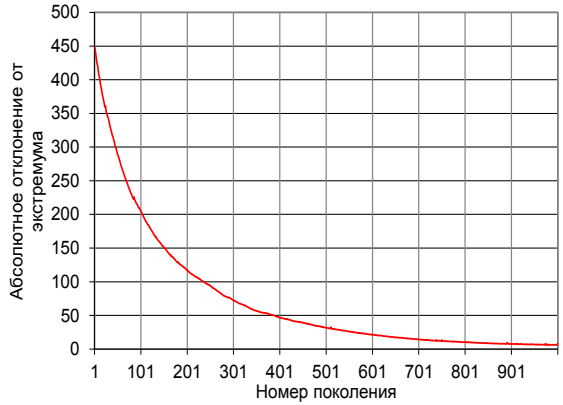
Сходимость. Процесс смены поколений повторяется пока на протяжении некоторого (заданного) числа поколений не будет происходить уменьшение (увеличение) значения целевой функции.

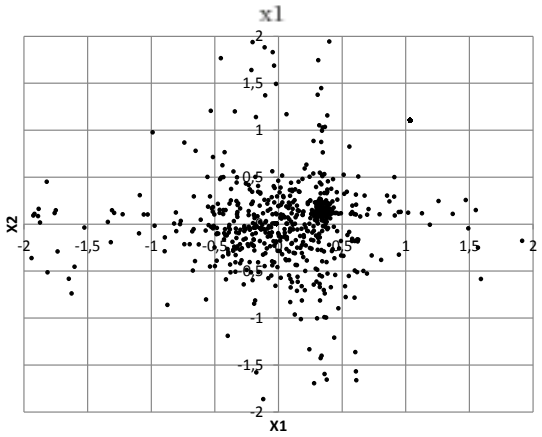
Генетический алгоритм (пример)

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









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

Графики функции Эккли получены с помощью сервиса www.wolframalpha.com

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.

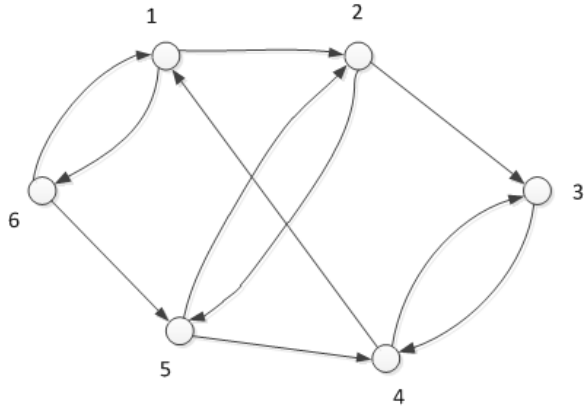
14. Модели теории графов
14.1 Граф

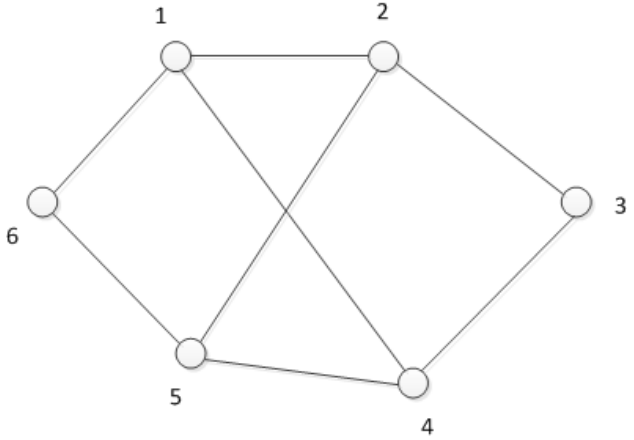
Граф G задается множеством точек или *вершин* $x_1, x_2, \ldots, x_n$ (которое обозначается через *X*) и множеством линий или *ребер* $a_1, a_2. \ldots, a_m$ (которое обозначается символом A), соединяющих между собой все или часть этих точек.
Таким образом, граф *G* полностью задается (и обозначается) парой (*X, A*).

Если ребра из множества *A* ориентированы, что обычно показывается стрелкой, то они называются дугами, и граф с такими ребрами называется *ориентированным* графом. Если ребра не имеют ориентации, то граф называется *неориентированным*.

В случае когда *G* = (*X, A*) является ориентированным графом и хотят пренебречь направленностью дуг из множества *A*, то неориентированный граф, соответствующий *G*, будем обозначают как $\bar{G}$ = (*X*, *A*).

Другое, часто употребляемое описание ориентированного графа *G* состоит в задании множества вершин *X* и соответствия *Г*, которое показывает, как между собой связаны вершины. Соответствие *Г* называется *отображением* множества *X* в *X*, а граф в этом случае обозначается парой *G* = (*X, Г*).





14.2 Пути, маршруты, веса, длина пути

Путем (или ориентированным *маршрутом*) ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальначальной вершиной следующей.

Маршрут - последовательность ребер, в которой каждое ребро за исключением, возможно, первого и последнего ребер, связано со своими двумя концевыми вершинами.

Путь или маршрут можно изображать также последовательностью вершин такое представление часто оказывается более полезным в тех случаях, когда осуществляется поиск простых орцепей или простых цепей.

Иногда дугам графа G сопоставляются (приписываются) числа — дуге (x_i, x_j) ставится в соответствие некоторое число c_{ij}, называемое ***весом***, или ***длиной***, или ***стоимостью*** (***ценой***) дуги. Тогда граф G называется графом *со* ***взвешенными дугами***.

Иногда веса (числа v_i) приписываются вершинам x_i графа, и тогда получается граф со ***взвешенными вершинами***.

Если в графе веса приписаны и дугам, и вершинам, то он называется просто ***взвешенным*** графом.

При рассмотрении пути μ, представленного последовательностью дуг ($a_1, a_2, \ldots, a_q$), за его ***вес*** (или ***длину***, или ***стоимость***) принимается число $L(\mu)$, равное сумме весов всех дуг, входящих в путь

$$L(\mu) = \sum_{(x_i, x_j) \in \mu} c_{ij}$$

Таким образом, когда слова «длина», «стоимость», «цена» и «вес» применяются к дугам, то они эквивалентны по содержанию, и в каждом конкретном случае выбирается такое слово, которое ближе подходит по смыслу и совпадает с принятым в литературе.
Длиной (или мощностью) пути называется число дуг, входящих в него

14.3 Некоторые определения

Остовный подграф. Пусть дан граф G = (X, A). Остовным подграфом *) G_p графа G называется граф (X, A_p), для которого A_p принадлежит A. Таким образом, остовный подграф имеет то же самое множество вершин, что и граф G, но множество дуг подграфа G_p является подмножеством множества дуг исходного графа.

Порожденный подграф. Пусть дан граф G = (X, Г). Порожденным подграфом G_S называется граф $(X_S, Г_S)$, для которого X_S принадлежит X и для каждой вершины X_i, приннадлежащей X_S, $Г_s(x_i) \cap X_S$. Таким образом, порожденный подграф состоит из подмножества вершин X_S множества вершин исходного графа и всех таких дуг графа G, у которых конечные и начальные вершины принадлежат подмножеству X_S.

Граф G = (X, A) называют полным, если для любой пары вершин x_i и x_j в X существует ребро (x_i, x_j) в G = (X, A), т. е. для каждой пары вершин графа G должна существовать по крайней мере одна дуга, соединяющая их.

Ориентированный граф называется ***сильно связным*** или ***сильным***, если для любых двух различных вершин x_i и x_j существует по крайней мере один путь, соединяющий x_i с x_j. Это определение означает также, что любые две вершины такого графа взаимно достижимы.

Ориентированный граф называется ***односторонне связным*** или ***односторонним***, если для любых двух различных вершин x_i и x_j существует по крайней мере один путь из x_i в x_j или из x_j в x_i (или оба одновременно).

Ориентированный граф называют ***слабо связным*** или ***слабым***, если для любых двух различных вершин графа существует по крайней мере один маршрут, соединяющий их.

Если для некоторой пары вершин орграфа не существует маршрута, соединяющего их, то такой орграф называется ***несвязным***.

14.4 Матричные представления

Пусть дан граф G, его матрица ***смежности*** обозначается через $A=[a_{ij}]$ и определяется следующим образом:

a_{ij} = 1, если в G существует дуга (x_i, x_j),
a_{ij} = 0, если в G нет дуги (x_i, x_j).

Таким образом, матрица ***смежности*** графа, может иметь имеет вид

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

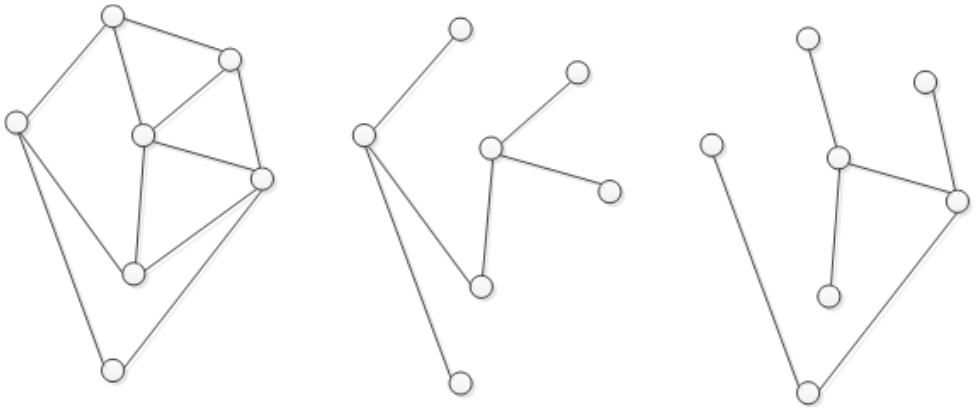
Пусть дан граф G с n вершинами и m дугами. Матрица ***инциденций*** графа G обозначается через $B = [b_{ij}]$ и является матрицей размерности ***n* X *m***, определяемой следующим образом:
b_{ij} = 1, если x_i является начальной вершиной дуги a_i,
b_{ij} = -1, если x_i является конечной вершиной дуги a_i,
b_{ij} = 0, если x_i не является концевой вершиной дуги a_i или если a_i является петлей.

14.5 Деревья, остов графа

Неориентированное дерево есть:
1) связный граф, содержащий *n* вершин и *n* - 1 ребер, либо
2) связный граф, не имеющий циклов, либо
3) граф, в котором каждая пара вершин соединена одной и только одной простой цепью.

Если G = (X, A) — неориентированный граф с *n* вершинами, то остовным деревом (или остовом) графа G называется всякий остовный подграф графа G, являющийся деревом (в смысле приведенного выше определения).

Остов графа G можно также расрассматривать как минимальный связный остовный подграф графа G, где «минимальность» понимается в том смысле, как говорилось, т. е. никакое собственное подмножество ребер этого остова не образует связный остовный подграф графа G.



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)

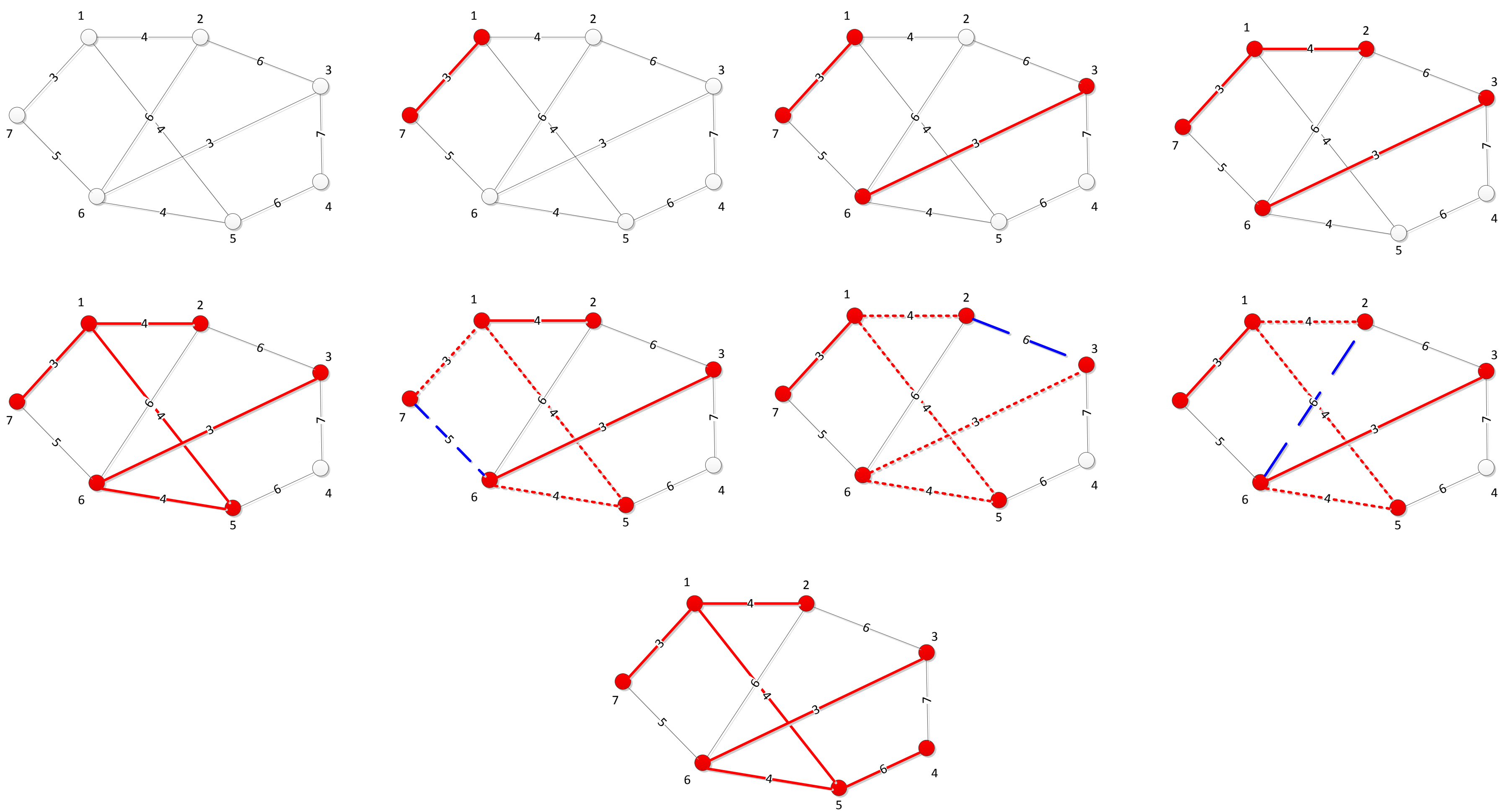
Шаг 1. Начать с вполне несвязного графа T, содержащего *n* вершин.
Шаг 2. Упорядочить ребра графа G в порядке неубывания их весов.
Шаг 3. Начав с первого ребра в этом списке, добавлять ребра в графе T, соблюдая условие: такое добавление не должно приводить к появлению цикла в T.
Шаг 4. Повторять шаг 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 Ts 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 Кратчайший остов (SST) графа (алгоритм Прима)

Шаг 1. Пусть $T_s = \{x_s\}$, где x_s — произвольно выбранная вершина, и $A_s = \varnothing$ (A_s является множеством ребер, входящих в SST).

Шаг 2. Для каждой вершины x_j не принадлежащей T_s найти вершину a_j принадлежащую T_s, такую, что

$$c(a_j, x_j) = \min_{x_j \in T_S} \left[c\left(x_i, x_j\right) \right] = \beta_j$$

приписать вершине x_j пометку $[\alpha_j \beta_j]$. Если такой вершины α_j нет, т. е. при $\Gamma(x_j) \cap T_s = \varnothing$,

приписать вершине x_j пометку $[0, \infty]$.

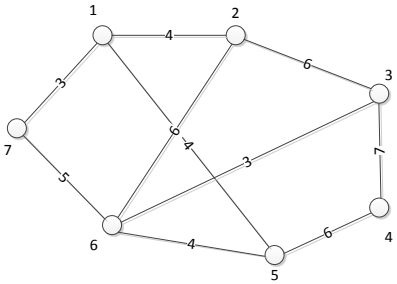
Шаг 3. Выбрать такую вершину x_j^*, что $\beta_{j^*} = \min_{x_j \in T_S} \left[\beta_j \right]$

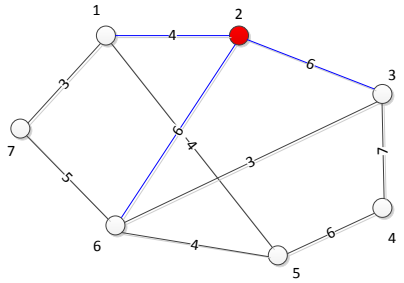
Обновить данные: $T_s = T_s \cup \{x_{j^*}\}$, $A_s = A_s \cup \{(a_{j^*}, x_{j^*})\}$.

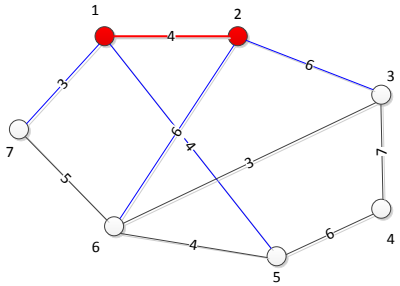
Если $|Ts| = n$, то остановиться. Ребра в A_s образуют SST.

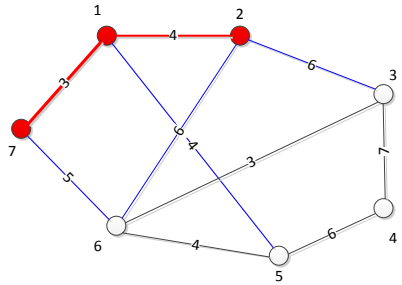
Если $|Ts| \neq n$, то перейти к шагу 4.

Шаг 4. Для всех $x_j \notin T_s$, таких, что $x_j \in \Gamma(x_{j^*})$, обновить пометки следующим образом.
если $\beta_j > C(x_{j^*}, x_j)$, то положить $\beta_j = C(x_{j^*}, x_j)$, $\alpha_j = x_{j^*}$ и вернуться к шагу 3.
если $\beta_j \leq C(x_{j^*}, x_j)$, то перейти к шагу 3.







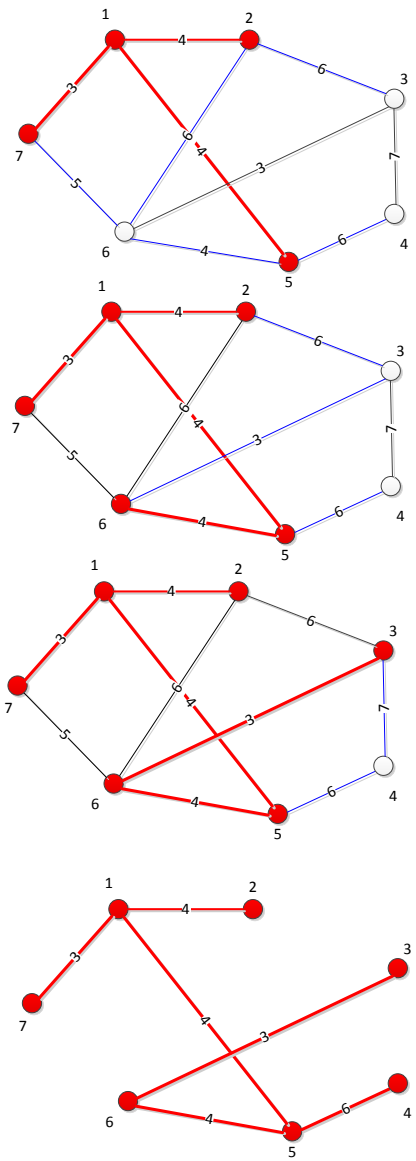


14.5 Пример алгоритма Прима

$T_s=\{2\}$ $A_s=\{\varnothing\}$

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 (6,5)=4 (4,5)=6 (3,2)=6	(1,2) (7,1) (5,1) (6,5)	3 [2, 6] 4 [0, ∞] 6 [5, 4]
1,2,5,6,7	(3,6)=3 (3,2)=6 (4,5)=6	(1,2) (7,1) (5,1) (6,5) (3,6)	3 [6, 3] 4 [5, 6]
1,2,3,5,6,7	(4,3)=7 (4,5)=6	(1,2) (7,1) (5,1) (6,5) (3,6) (4,5)	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 с содержит номера вершин - k-й элемент C[k] есть номер предпоследней вершины на текущем кратчайшем пути из Vi в Vk.

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".

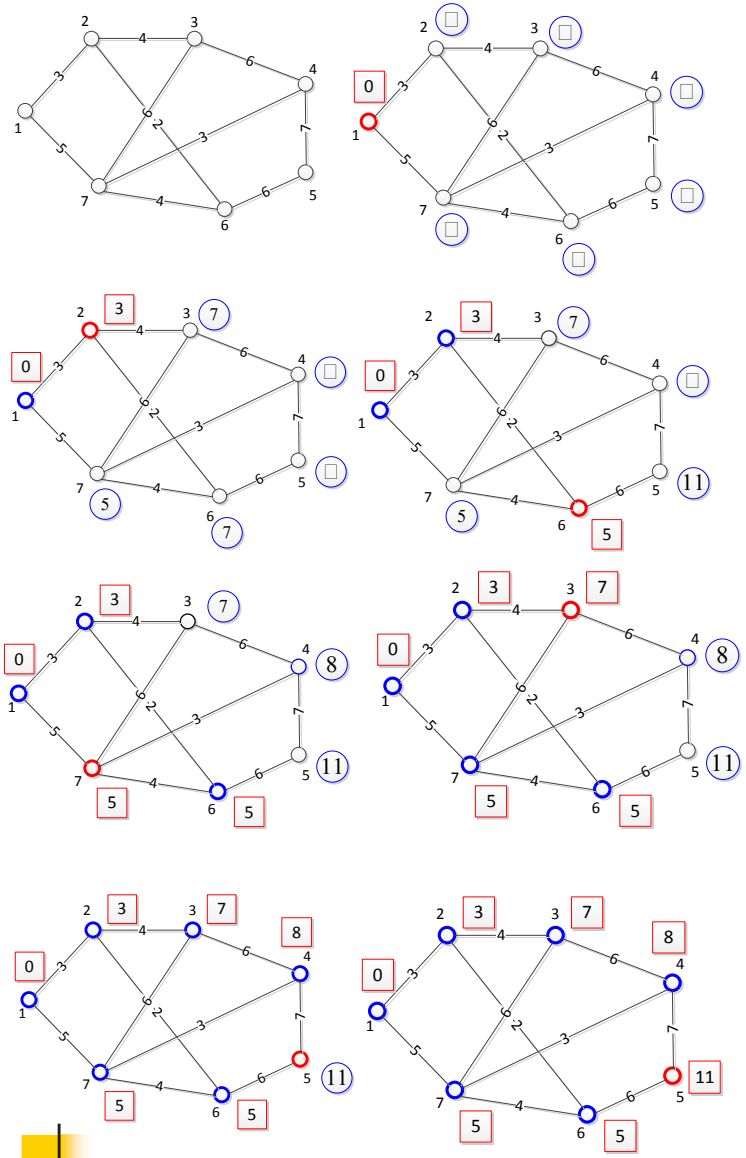
1 (инициализация). В цикле от 1 до N заполнить нулями массив S; заполнить числом i массив C; перенести i-ю строку матрицы A в массив B,
S[i]:=1; C[i]:=0 (i - номер стартовой вершины)

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]<=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 Vi ... Vk длиннее, чем путь Vi...Vj Vk).
(Если все S[k] отмечены, то длина пути от Vi до Vk равна B[k]. Теперь надо) перечислить вершины, входящие в кратчайший путь).

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

Алгоритм Дейкстры (пример)



1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞
	0+3=3	∞	∞	∞	∞	5
		3+4=7	∞	∞	3+2=5	5
		~~∞~~ 7	∞	5+6=11		~~5+4=9~~ 5
		~~5+6=11~~ 7	5+3=8	~~∞~~ 11		
			~~7+6=13~~ 8	~~∞~~ 11		
				~~8+7=15~~ 11		

Алгоритм Дейкстры (поиск кратчайшего пути)

```
#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;
}
```

Алгоритм Флойда-Уоршалла

Задана матрица весов c=[c_{ij}] i,j =1…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…k. Очевидно, что C_{0ij} — длина (вес) ребра (i, j), если таковое существует (в противном случае его длина может быть обозначена как ∞).

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

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 Ckij, ∀ i,j для k = 1… n. Полученные значения C_{nij} являются длинами кратчайших путей между вершинами i, j.

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

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

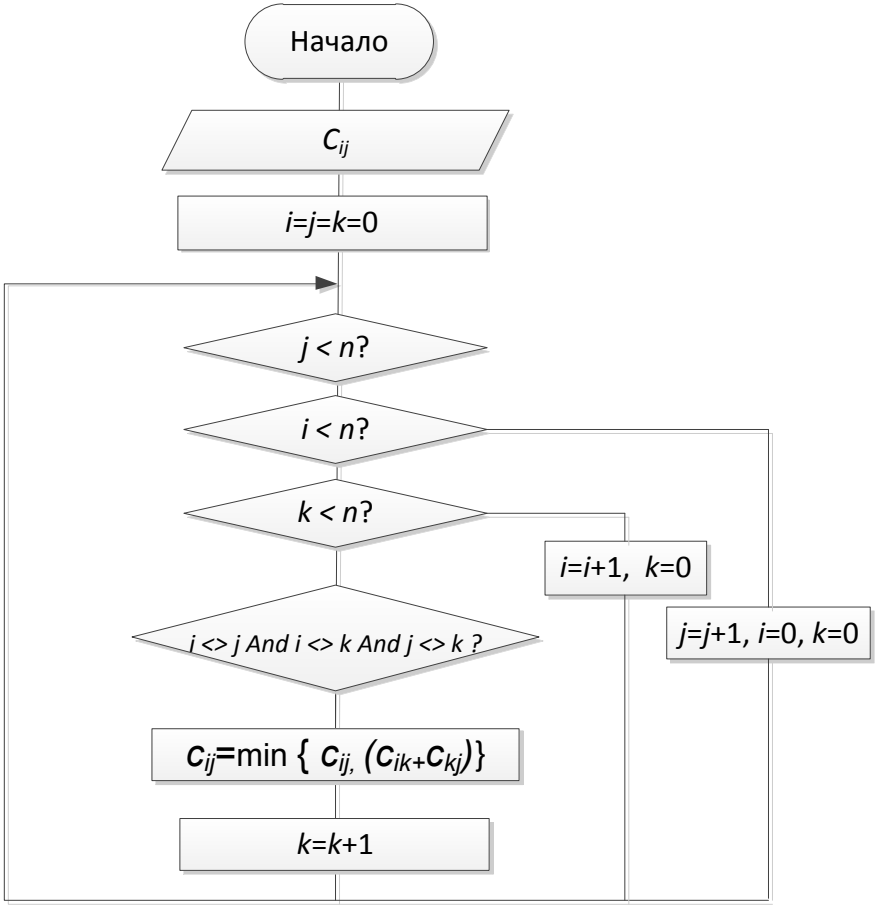
Шаг 4.

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

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

(в) Если все $c_{ii} \geq 0$ 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 I, 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
```

Алгоритмы Йена (Yen's algorithm) (поиск k-кратчайших путей между двумя вершинами)

Присвоение начальных значений.

Шаг 1. Найти P^1. Присвоить k=2. Если существует только один кратчайший путь P^1, включить его в список L_0 и перейти к шагу 2.

Если таких путей несколько, но меньше, чем К, включить один из них в список L_0, а остальные в список L_1. Перейти к шагу 2.

Если существует К или более кратчайших путей P^1, то задача решена. Остановка.

Нахождение отклонений.

Шаг 2. Найти все отклонения P_i^k(k-1)-го кратчайшего пути P^{k-1} для всех i=1, 2, …, q_{k-1}, выполняя для каждого i шаги с 3-го по 6-й.

Шаг 3. Проверить, совпадает ли подпуть, образованный первыми i вершинами любого из P^j путей (j=1, 2,…,k-1). Если да, то присвоить $c(x_i^{k-1}, x_{i+1}^j)$=; иначе ничего не изменять. (При выполнении алгоритма вершина x_1 обозначается s).

Шаг 4. Используя алгоритм кратчайшего пути, нужно найти кратчайшие пути S_i^k от x_i^{k-1} к t, исключая из рассмотрения вершины s, x_i^{k-1}, x_3^{k-1},..., x_i^{k-1}. Если существует несколько кратчайших путей, то взять в качестве S_i^k один из них.

Шаг 5. Построить P_i^k, соединяя R_i^k(=s, x^{-1}, x^{-1},…, x^{-1}) с S и поместить P в список L_1.

Шаг 6. Заменить элементы матрицы весов, измененные на шаге их первоначальными значениями и возвратиться к шагу 3.

Выбор кратчайших отклонений.

Шаг 7. Найти кратчайший путь в списке L_1. Обозначить этот путь P^k и переместить его из L_1 в L_0.

Если k=K, то остановка. Список L_0-список К-кратчайших путей.

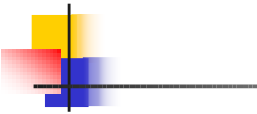
Если k<K, то присвоить k=k+1, перейти к шагу 2.

Если в L_1 имеется более чем один кратчайший путь (h-путей), то поместить в L_0 любой из них и продолжать как выше до тех пор, пока увеличенное на h число путей, уже находящихся в L_1, не совпадет с К или не превысит его.

Тогда остановка.

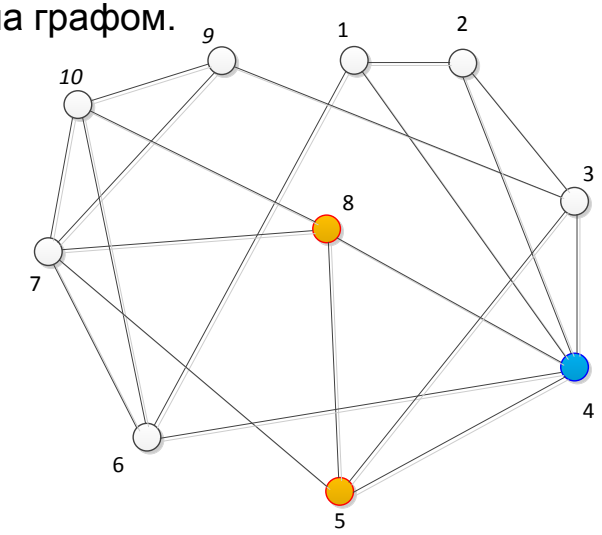
Некоторые алгоритмы поиска путей

-Алгоритм Дейкстры (поиск кратчайших путей (пути) от заданной вершины графа)
-Алгоритм Беллмана-Форда (поиск кратчайшего пути от заданной вершины графа)
-Алгоритм Йена (поиск k-кратчайших путей между двумя вершинами графа)
-Алгоритм Флойда-Уоршелла (поиск кратчайших путей между всеми вершинами графа)
-Эвристический алгоритм A* (A звезда)



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

Таким образом,
$R_{\lambda 0}(x_i) = \{x_j \mid v_j d(x_i, x_j) \le \lambda\ ,\ x_j \in X\}$
$R_{\lambda t}(x_i) = \{x_j \mid v_j d(x_j, x_i) \le \lambda\ ,\ x_j \in X\}$

Для каждой вершины x_i определим следующие два числа:

$S_0(x_i) = \max\{v_j d(x_i, x_j)\}\ x_j \in X$ Число внешнего разделения
$S_t(x_i) = \max\{v_j d(x_j, x_i)\}\ x_j \in X$ 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.

$S_0(x_i^*) = \min\{\ S_t(x_i)\ \}\ x_i \in X$ - внешний центр графа
$S_t(x_t^*) = \min\{\ S_t(x_i)\ \}\ x_i \in X$ - 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

Решение задачи 1
Минимум максимумов
по строкам

3
3
3
3
2
3
3
2
3
3

3	3	3	3	2	3	3	2	3	3

Числа внутреннего разделения

Числа внешнего разделения

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, Г). Для каждой вершины $x_i \in X$ определим два числа, которые назовем передаточными числами:

$$\sigma_0(x_i) = \sum_{x_j \in X} v_i d(x_i, x_j)$$

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

(внешнее и внутреннее передаточные числа)

Вершина x_0 для которой $\sigma_0(x_0) = \min_{x_i \in X}\{\sigma_0(x_i)\}$ - внешняя медиана графа

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

размещение медиан графа

Матрица длин кратчайших путей

	1	2	3	4	5	6	7	8	9	10	Решение задачи 2 Минимум сумм по строкам
1	0	1	2	1	2	1	2	2	3	2	16
2	1	0	1	1	2	2	3	2	2	3	17
3	2	1	0	2	1	3	2	2	1	2	16
4	1	1	2	0	1	1	2	1	3	2	14
5	2	2	1	1	0	2	1	1	2	2	14
6	1	2	3	1	2	0	1	2	2	1	15
7	2	3	2	2	1	1	0	1	1	2	15
8	2	2	2	1	1	2	1	0	2	1	14
9	3	2	1	3	2	2	1	2	0	1	37
10	2	3	2	2	2	1	2	1	1	0	25
	16	17	16	14	14	15	15	14	37	25	

Внутренние передаточные числа

Внешние передаточные 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	3	3	3	3	2	3	3	2	3	3
Внутренние передаточные числа	16	17	16	14	14	15	15	14	37	25

Числа внешнего разделения

Внешние передаточные 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 медианя 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	∞	1	∞	1	∞	∞	∞	∞
2	1	0	1	1	∞	∞	∞	∞	∞	∞
3	∞	1	0	∞	1	∞	∞	∞	1	∞
4	1	1	∞	0	1	1	∞	1	∞	∞
5	∞	∞	1	1	0	∞	1	1	∞	∞
6	1	∞	∞	1	∞	0	1	∞	∞	1
7	∞	∞	∞	∞	1	1	0	1	1	∞
8	∞	∞	∞	1	1	∞	1	0	∞	1
9	∞	∞	1	∞	∞	∞	1	∞	0	1
10	∞	∞	∞	∞	∞	1	∞	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

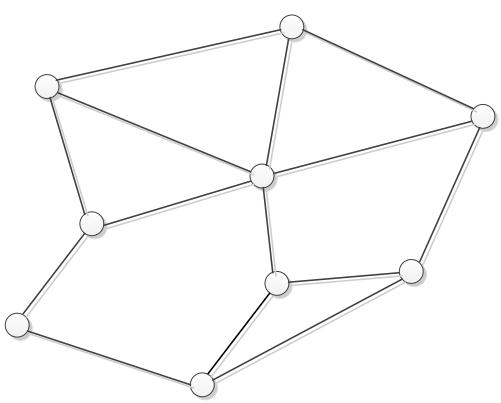
1. Результат – матрица кратчайших путей.

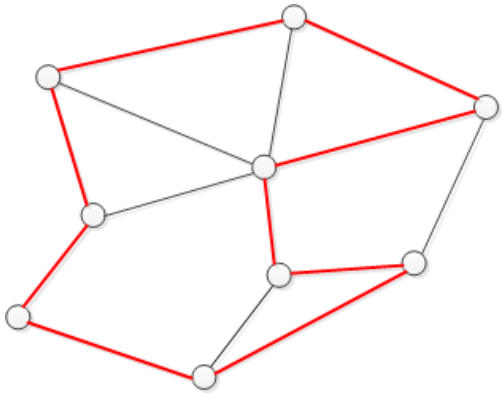
Задача коммивояжера

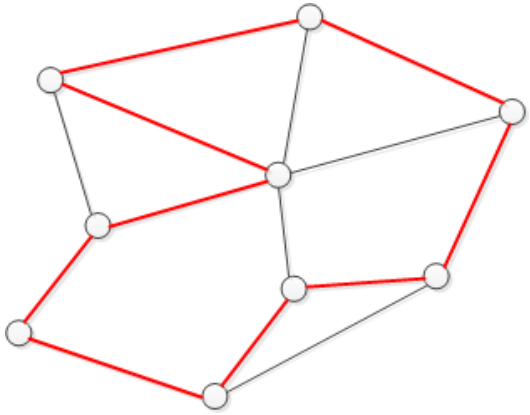
(TSP - Travelling salesman problem)

Гамильтонов цикл
Гамильтонова цепь (путь)

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



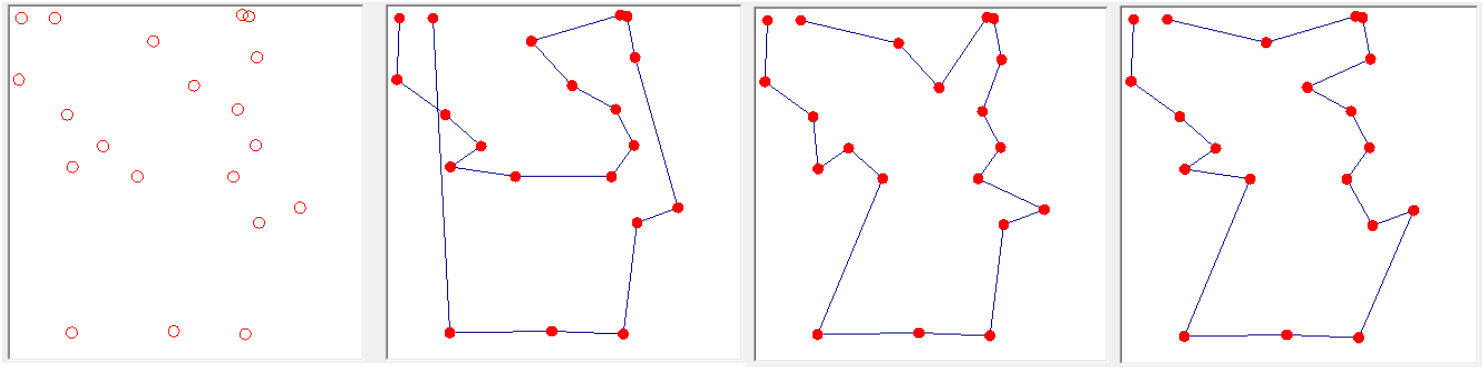




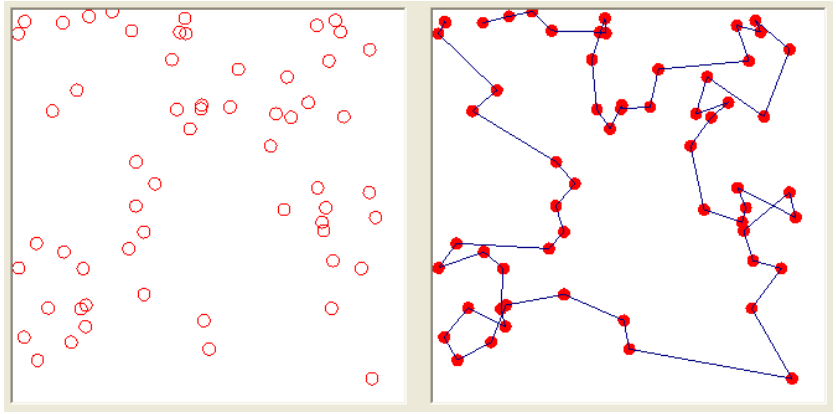
Задача коммивояжера: нахождение Гамильтонова цикла (цепи) наименьшей длины.
NP – полная задача

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.

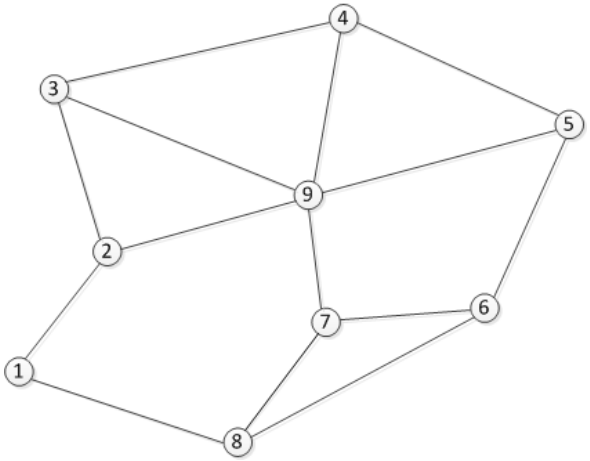
$$p_{ij} = \frac{\eta_{ij}^{\beta}\tau_{ij}^{\alpha}}{\sum_{k\in D}\left(\eta_{ik}^{\beta}\tau_{ik}^{\alpha}\right)}$$

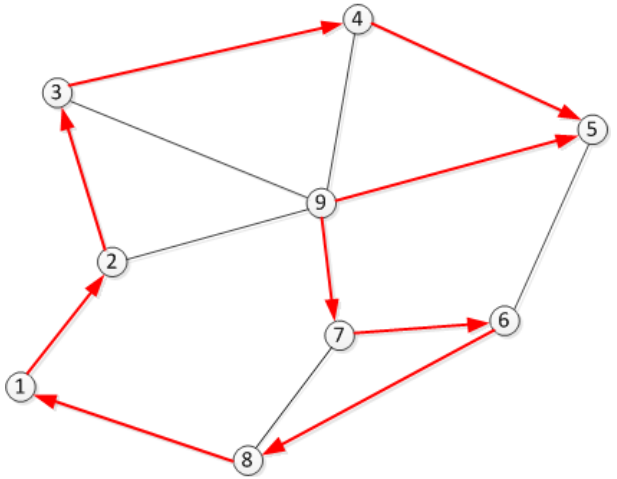
D – множество номеров достижимых вершин (из вершины *i*).

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

τ_{ij}^{α} – 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

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

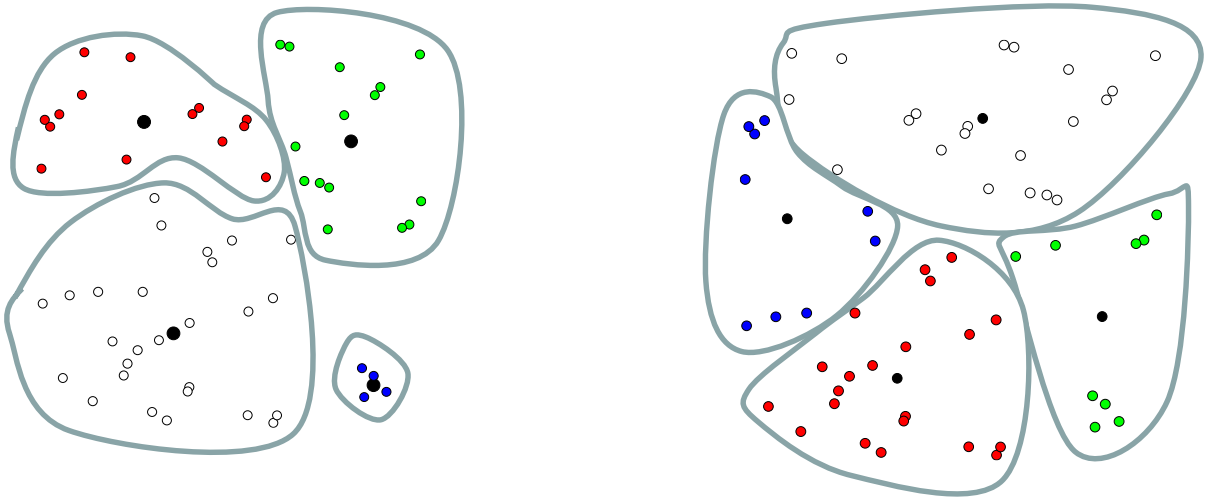
$$\Delta\tau^{(r)} = \frac{Q}{L_0^{(r)}}$$

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

Кластерный анализ (кластеризация)

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

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 служит разбиение множества точек на плоскости на подмножества, по признаку близости их координат.

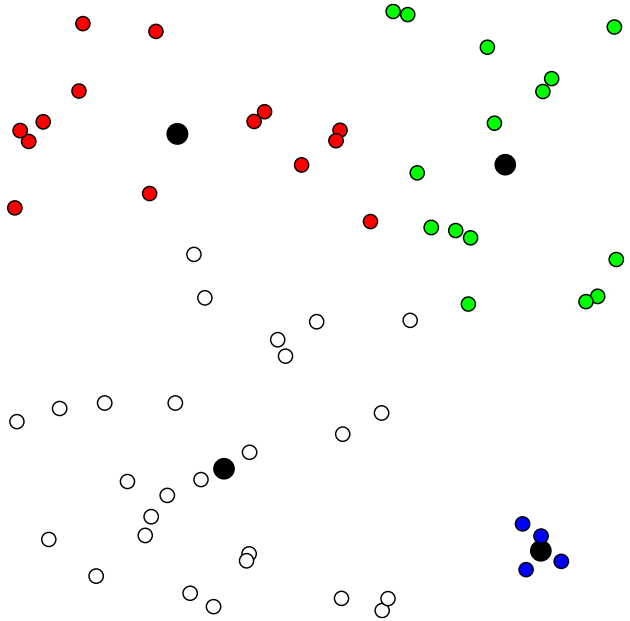


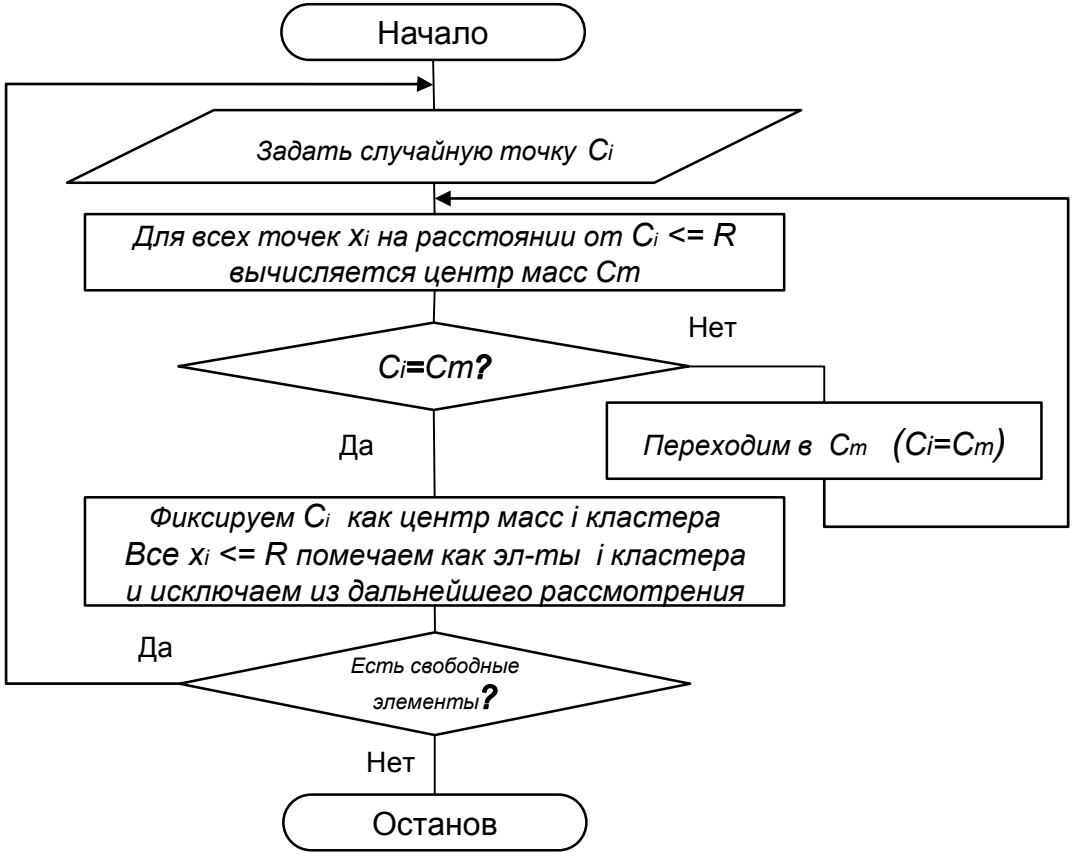
Решение задачи заключается в минимизации суммарного отклонения расстояний (метрик) объектов от центров кластеров (центров масс)

Алгоритм кластеризации FOREL (произвольный элемент)

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

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



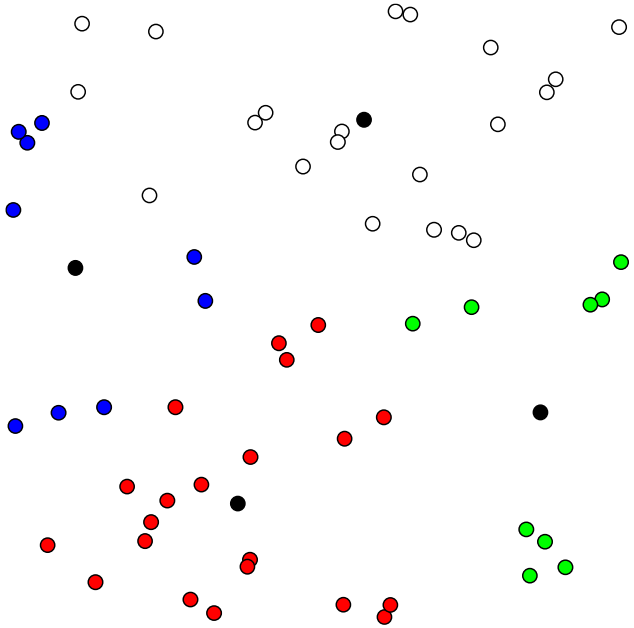


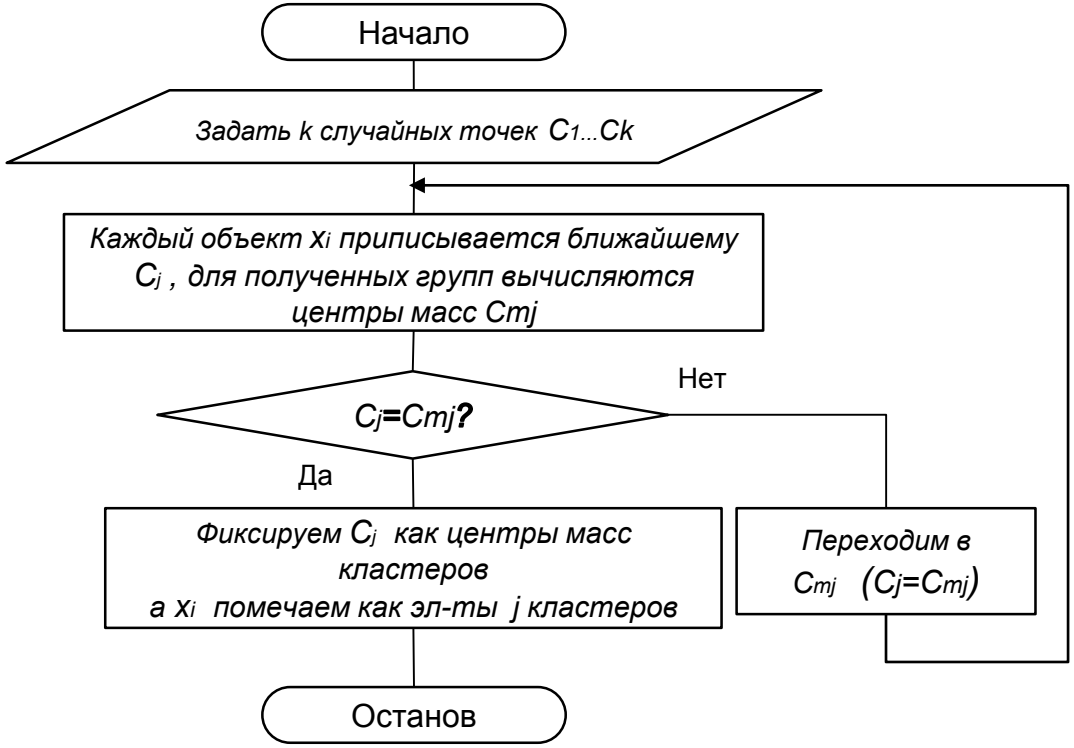
В результате решения получается некоторое число кластеров, средний размер которых близок к *R*

Алгоритм кластеризации *k*-средних (*k*-means)

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

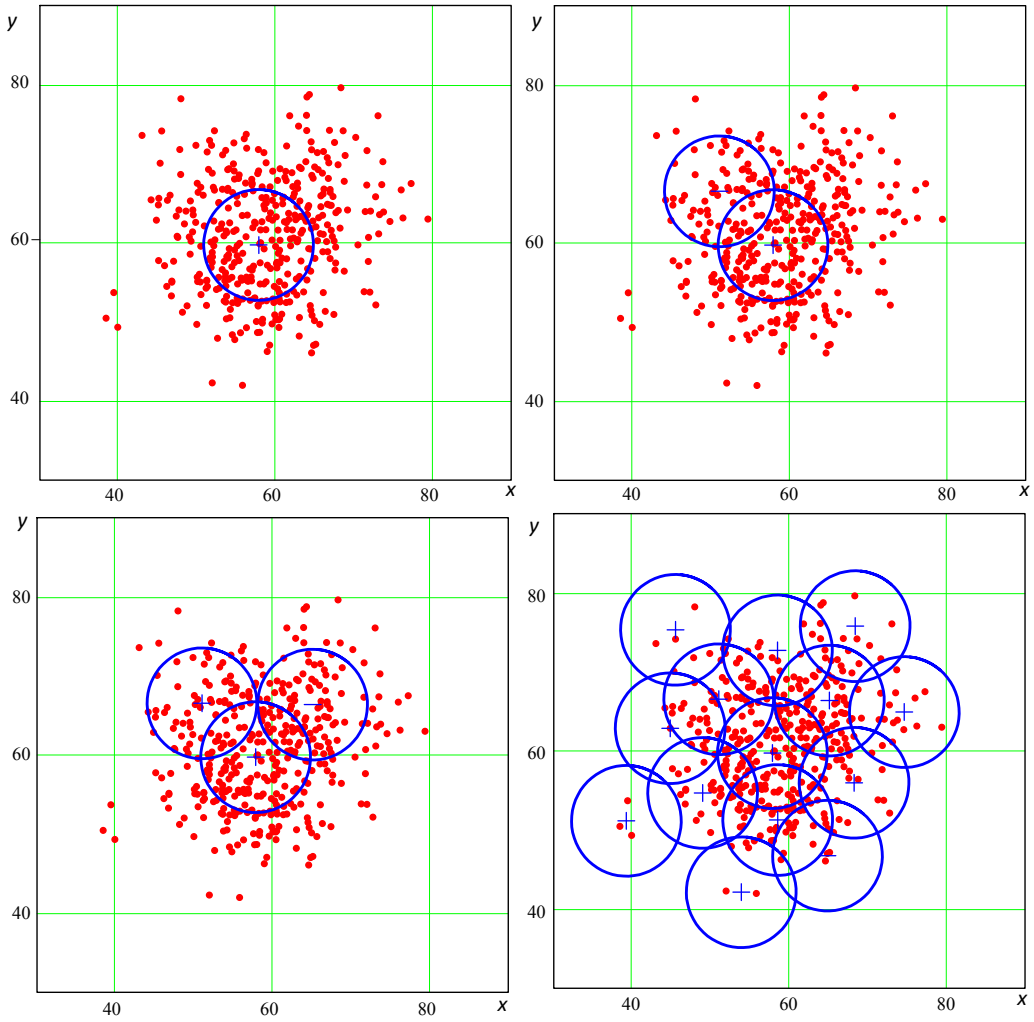
Объекты имеют некоторые характеристики (например, координаты на плоскости *x* и *y*).
Задано число кластеров *k*.





В результате решения получает *k* кластеров

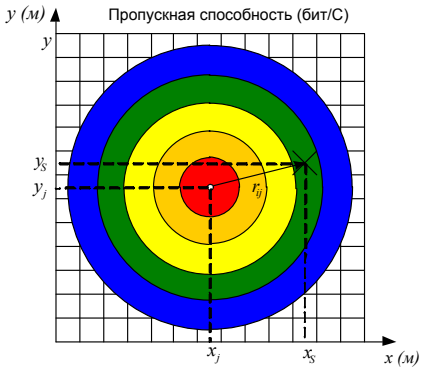
Пример кластеризации

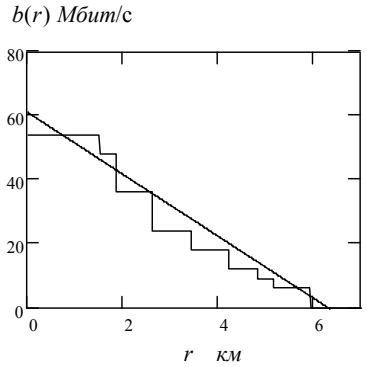


Применение кластерного анализа для выбора структуры сети

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

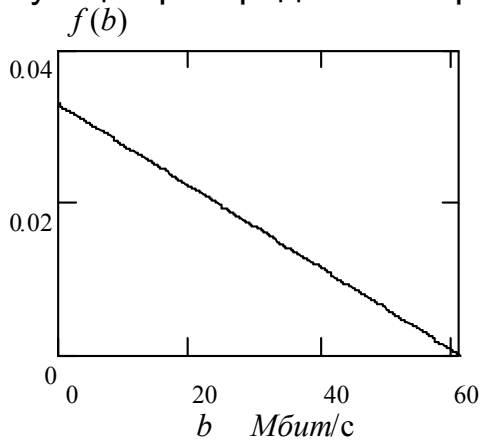
Пропускная способность БС



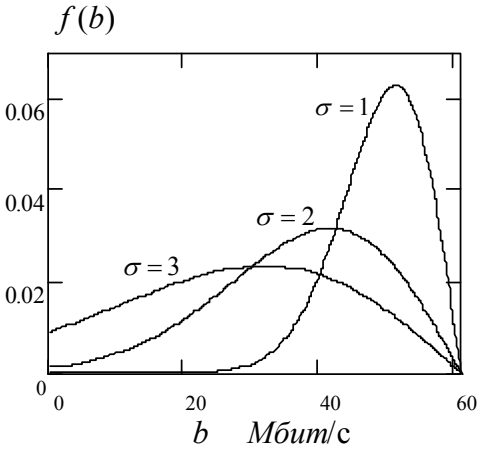


$$b(r) = b_0\left(1 - \frac{r}{R}\right)$$

Функция распределения пропускной способности БС



Равномерное распределение трафика по территории



Нормальное распределение трафика по территории

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

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

Результаты

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.

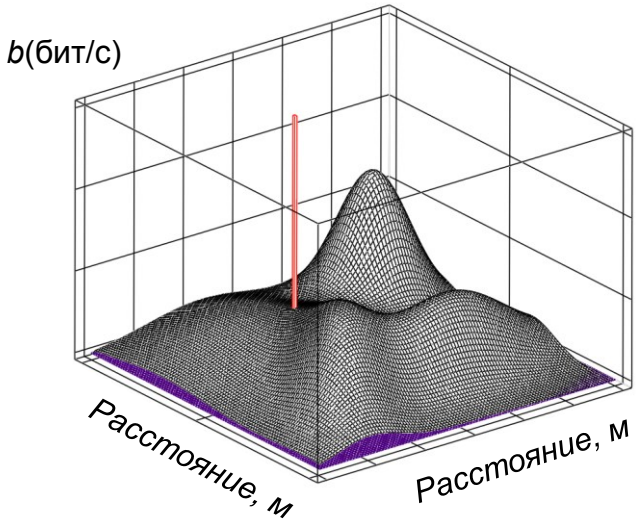
$$x_0 = \frac{1}{A}\iint_S y \cdot f_a(x,y)dxdy \quad \text{(м)}$$

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

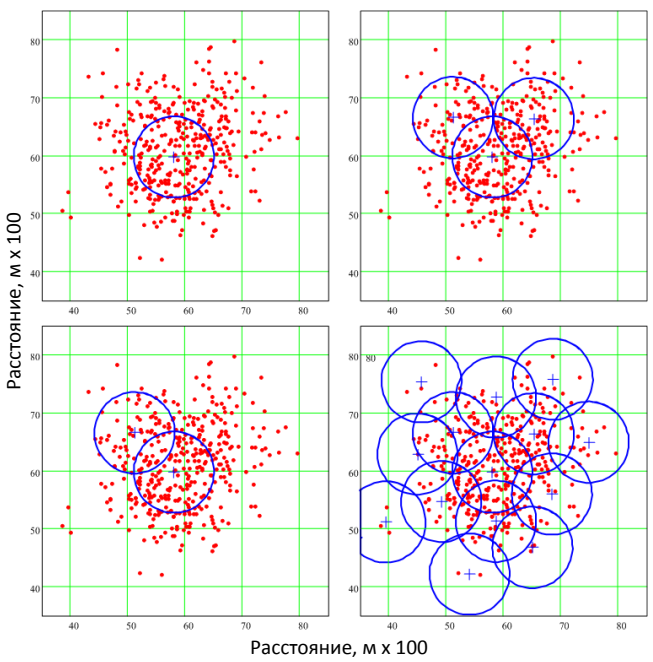
$$A = \iint_S f_a(x,y)dxdy \quad \text{(бит/с)}$$

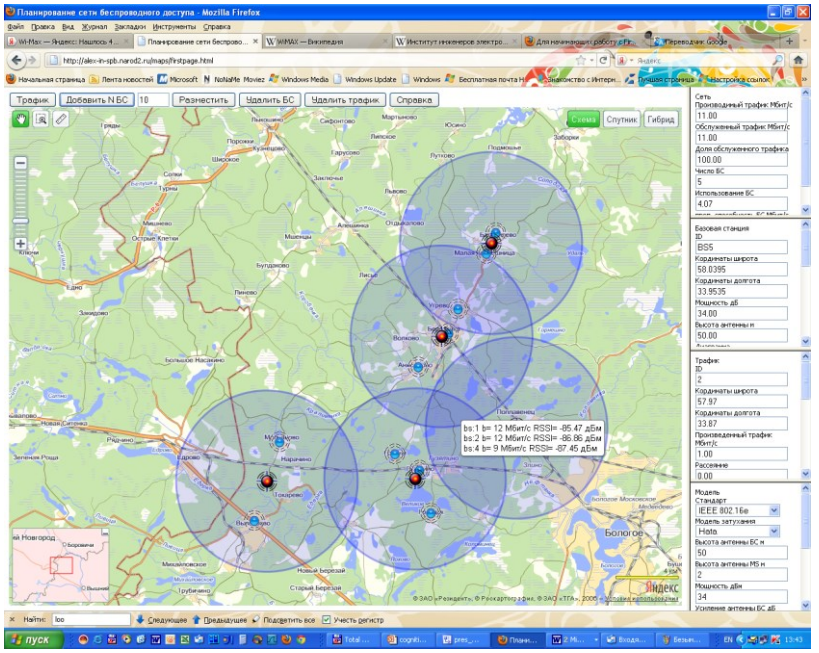
$$x_0 = \frac{1}{A}\sum_{i=1}^{N} a_i \cdot x_i \text{(м)} \qquad y_0 = \frac{1}{A}\sum_{i=1}^{N} a_i \cdot y_i \quad \text{(м)}$$

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



4.4 Моделирование и реализация, публикации

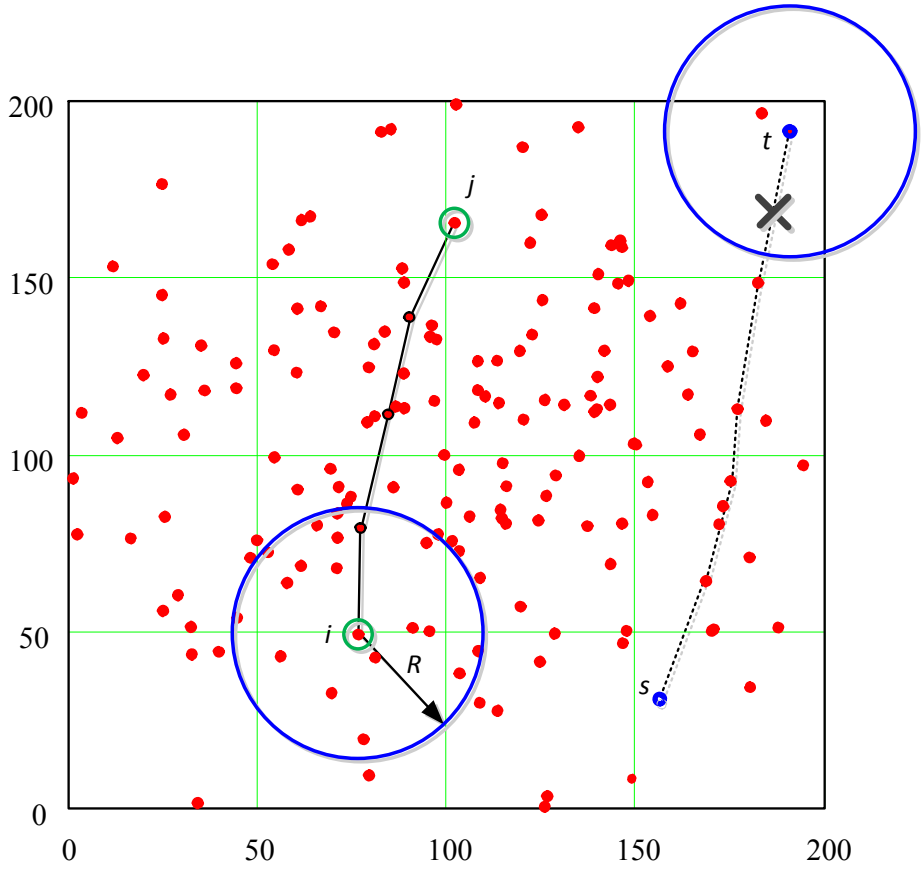




СПб ГУТ)))

Случайные графы
(модели сети беспроводной связи)

Пуассоновское поле



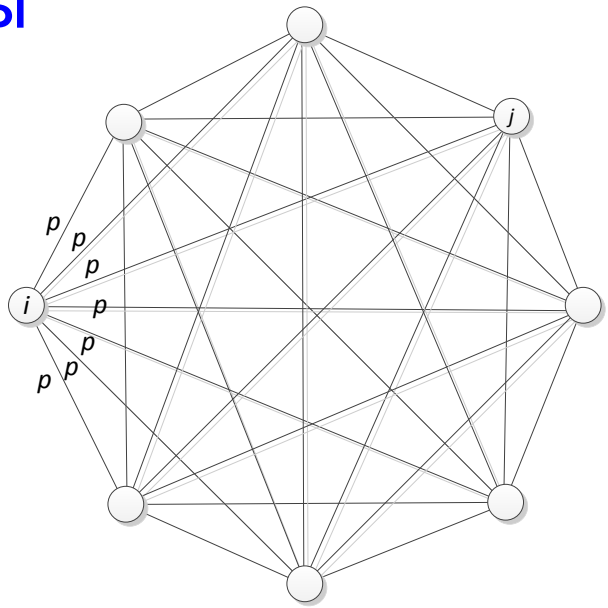
Вероятность связности узла

$$p_n = 1 - e^{-\rho\pi R^2}$$

Вероятность связности сети

$$p_c \approx \frac{n^{(con)}_{ij}}{n^2}, \qquad i,j = 1 \ldots n$$

Случайные графы



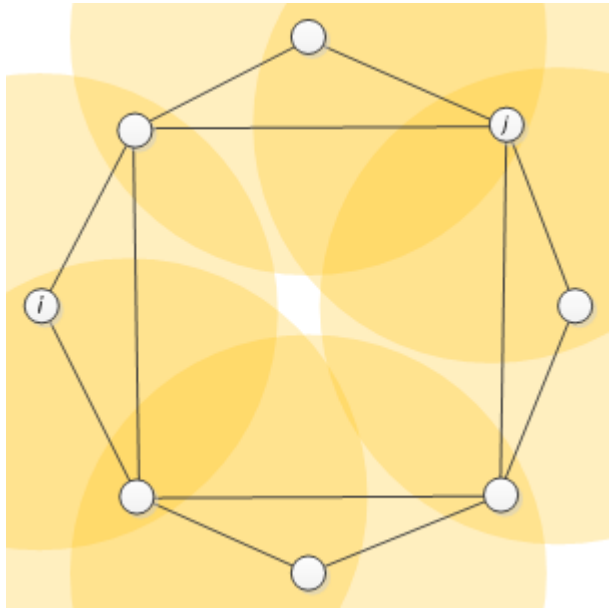
Пуассоновское поле

$$p = \rho\pi R^2 / n \qquad p = \rho\frac{3}{4}\pi R^3 / n$$

$$p = \frac{c \cdot \ln n}{n}$$

$c > 1,$

$c < 1$

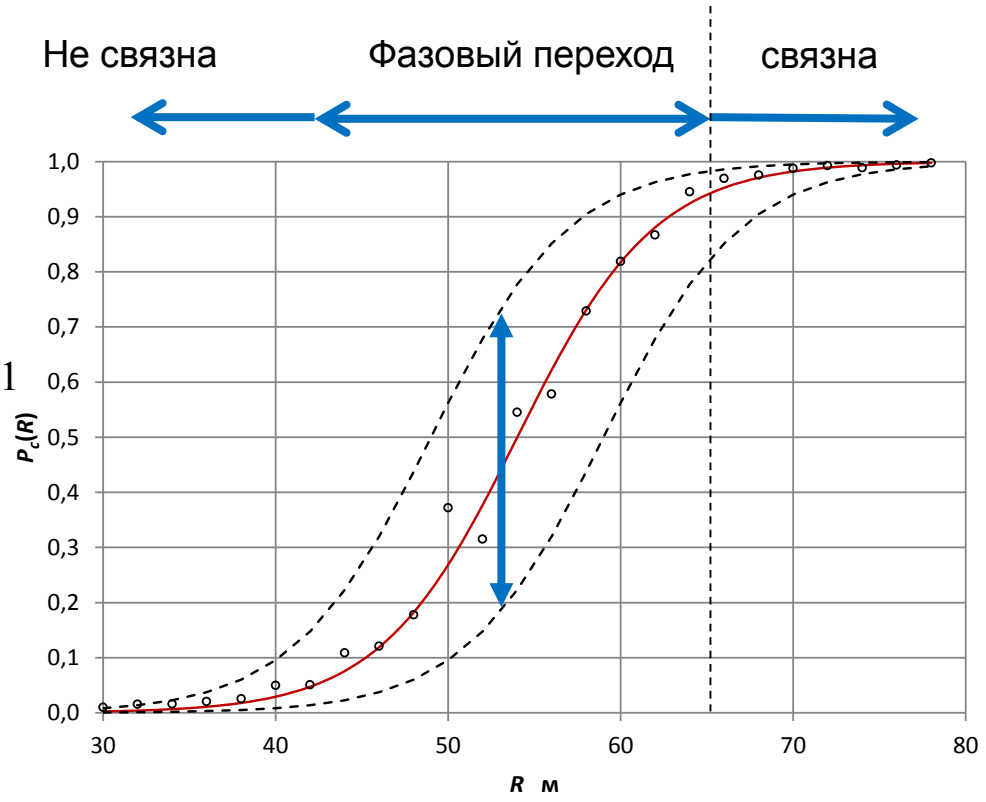


Случайные графы

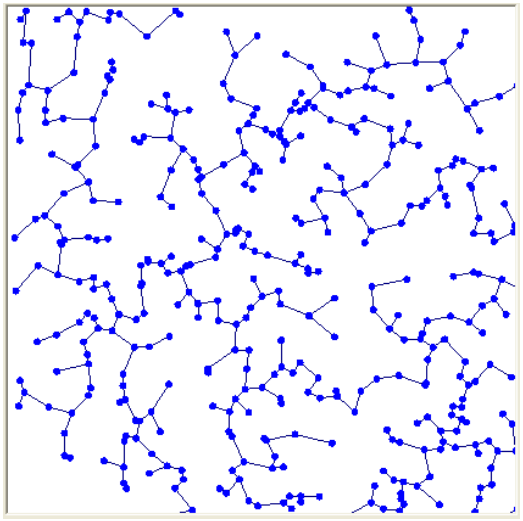
Пуассоновское поле

$$\tilde{p}_c(R) = \frac{1}{1 + e^{-\frac{R - R_0}{n_0}}}$$

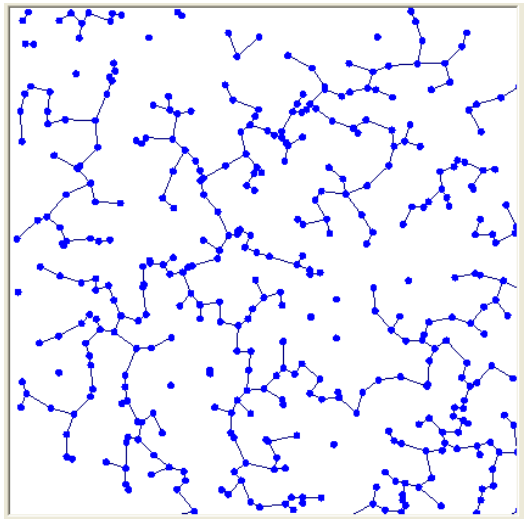
$$p_0 = \rho \pi R_0^2 / n, \quad n_0 = \rho \pi R_0^2, \quad p_0 \approx e^{-1}$$



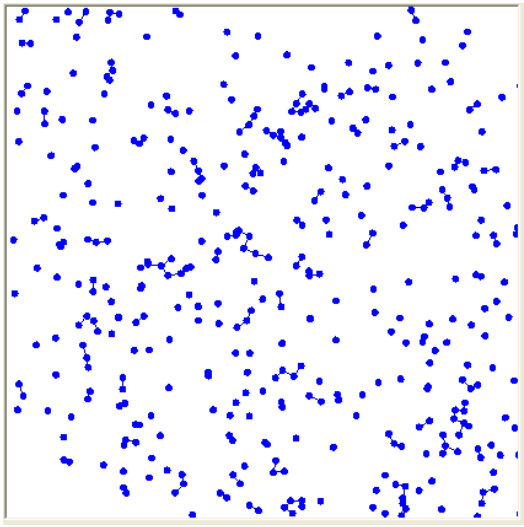
Изменение связности сети



Связность близка к 1

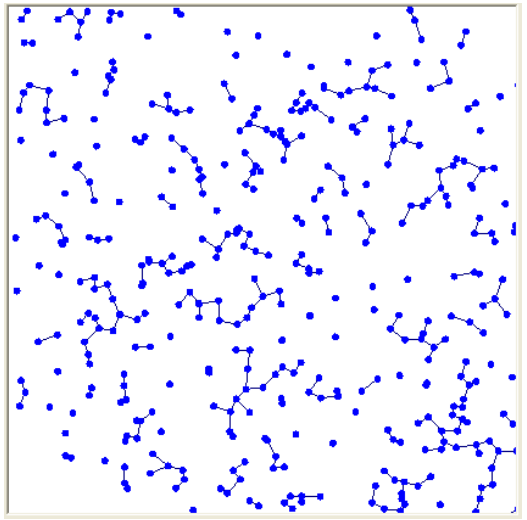


Фазовый переход
(p>1/n гигантская компонента ~n)



Связность близка к 0

Фазовый переход
(p<1/n компоненты
~ln(n))



Random graph with 400 nodes. Prim's minimum spanning tree algorithm used for network connectivity illustration.

Влияние числа узлов сети на дисперсию связности

-СКО (<10%) если число узлов сети более 50.

-В диапазоне значений от 2 до 50 узлов дисперсия связности может достигать 40%.

-Высокая дисперсия характерна для состояния фазового перехода .

The model applicability depends on the number of nodes. For practical purposes *n* must be more then 50 (for given network parameters).
In case of lower nodes number the model gives rough result of connectivity estimation.

Модели надежности сети связи

3.3 Модели надежности сети связи

3.3.1 Общие определения

В отрасли связи действует ГОСТ Р 53111-2008, определяющий устойчивость функционирования сети связи общего пользования. Ниже приведены определения согласно данному документу.

1 устойчивость функционирования сети электросвязи: Способность сети электросвязи выполнять свои функции при выходе из строя части элементов сети в результате воздействия дестабилизирующих факторов.

2 дестабилизирующий фактор: Воздействие на сеть электросвязи, источником которого является физический или технологический процесс внутреннего или внешнего по отношению к сети электросвязи характера, приводящее к выходу из строя элементов сети.

3 коэффициент готовности: Вероятность того, что объект находится в работоспособном состоянии в любой момент времени, кроме планируемых периодов, в течение которых применение объекта по назначению не предусматривается.

4 коэффициент оперативной готовности: Вероятность того, что объект находится в работоспособном состоянии в любой момент времени, кроме планируемых периодов, в течение которых применение объекта по назначению не предусматривается, и, начиная с этого момента, будет работать безотказно в течение заданного интервала времени.

5 надежность сети электросвязи: Свойство сети электросвязи сохранять способность выполнять требуемые функции в условиях воздействия внутренних дестабилизирующих факторов (т.е. сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в заданных режимах и условиях применения и технического обслуживания).

Общие определения

6 живучесть сети электросвязи: Свойство сети электросвязи сохранять способность выполнять требуемые функции в условиях, создаваемых воздействиями внешних дестабилизирующих факторов.

7 работоспособное состояние: Состояние объекта, при котором значения всех параметров, характеризующих способность выполнять им заданные функции, соответствуют требованиям или нормам.

8 средняя наработка на отказ: Отношение суммарной наработки восстанавливаемого объекта к математическому ожиданию числа его отказов в течение этой наработки.

9 вероятность связности (связность) направления электросвязи: Вероятность того, что на заданном направлении электросвязи существует хотя бы один путь, по которому возможна передача информации с требуемыми качеством и объемом.

10 внутренний дестабилизирующий фактор: Дестабилизирующий фактор, источник которого расположен внутри сети электросвязи или ее элементов.

11 внешний дестабилизирующий фактор: Дестабилизирующий фактор, источник которого расположен вне сети электросвязи.

12 направление связи (основное направление связи): Совокупность линий передачи и узлов связи, обеспечивающая связь между двумя пунктами сети для обеспечения деятельности органов государственного управления, обороны, безопасности, охраны правопорядка, мобилизационной готовности при чрезвычайных ситуациях.

В качестве показателя надежности каналов электросвязи применяется коэффициент готовности Кг канала электросвязи, определяемый выражением:

$Кг = To/(To + Tв)$, (1)

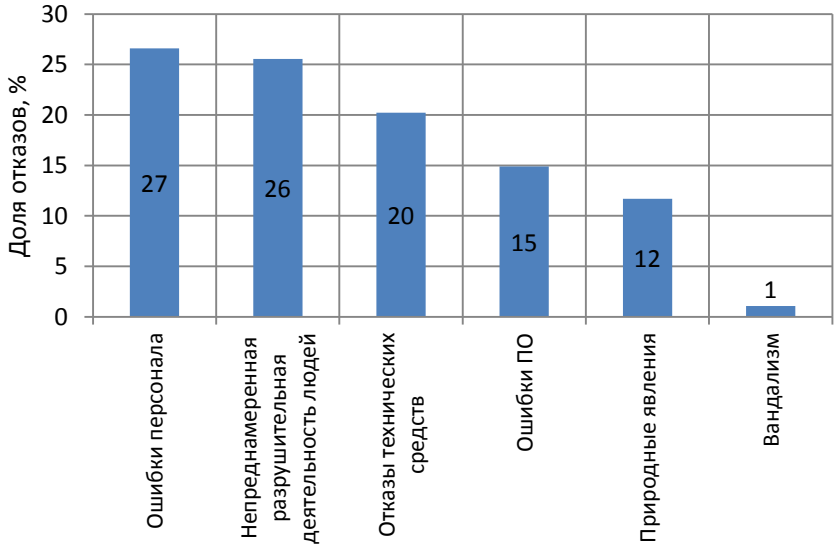
где To - среднее время наработки на отказ канала электросвязи;

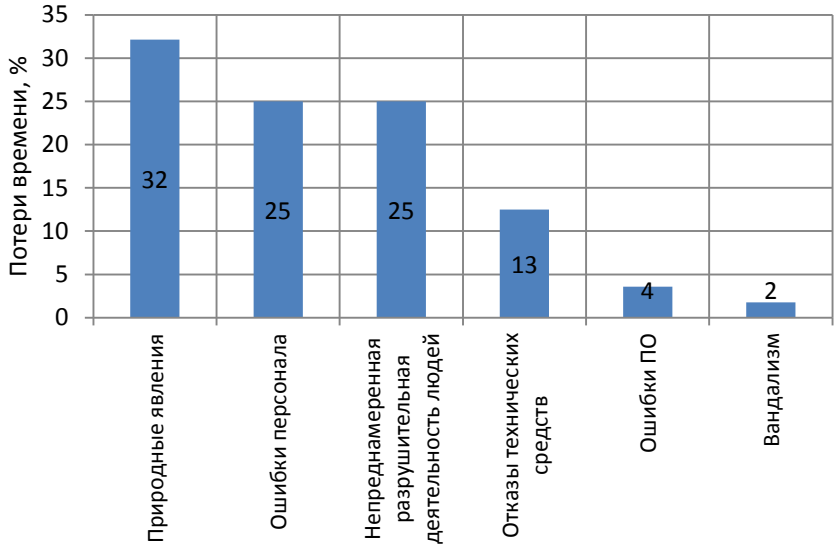
Тв - среднее время восстановления работоспособности канала электросвязи.

Иллюстрации (надежность сетей связи)

Alcatel-Lucent Bell Labs

Уязвимости системы телекоммуникаций

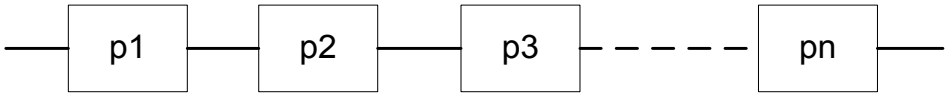




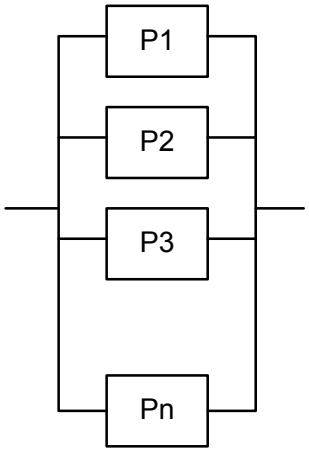
США (FCC)

N	Группа факторов	Доля от общего количества отказов, %	Потери пользователь-ского 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

3.3.2 Надежность простейших структур (последовательное и параллельное включение элементов)

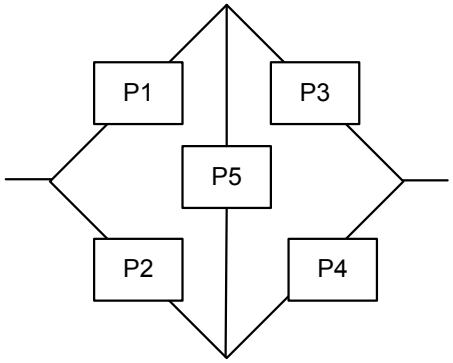


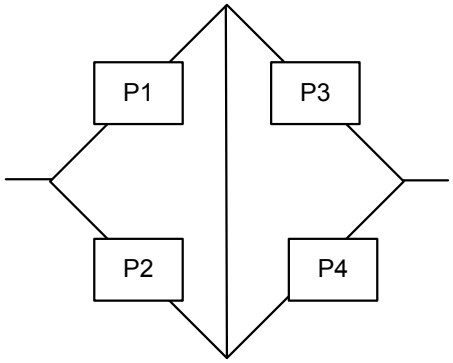
$$P = \prod_{i=1}^{n} p_i$$



$$P = 1 - \prod_{i=1}^{n} (1 - p_i)$$

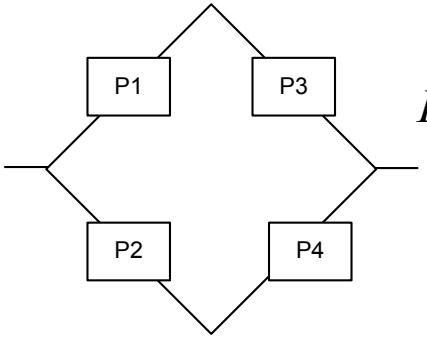
3.3.3 Мостовая схема включения





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

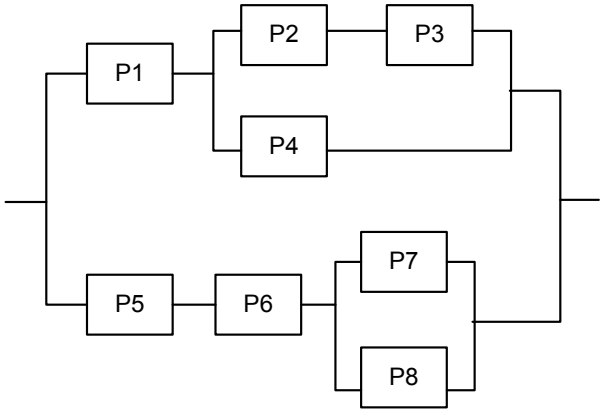
$$P = p_5 P(W \mid e_5) + q_5 P(W \mid \bar{e}_5)$$

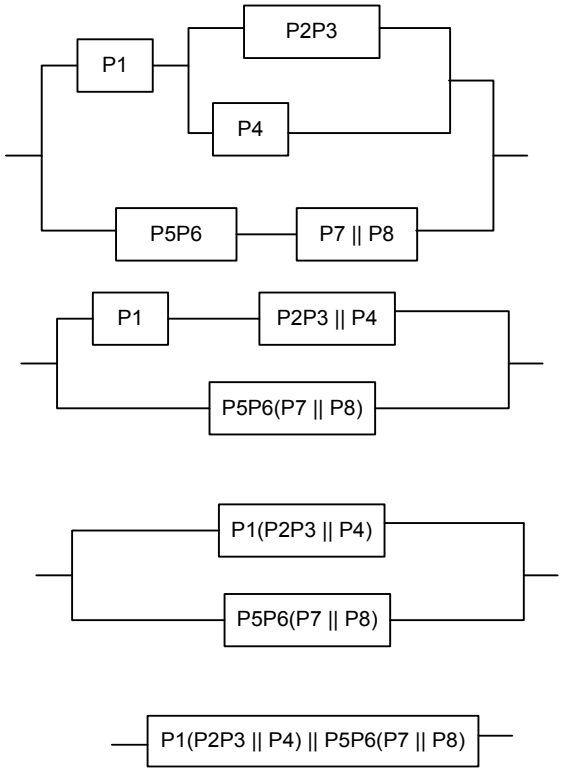


$$P(W \mid \bar{e}_5) = 1 - (1 - p_1 p_3)(1 - p_2 p_4)$$

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

3.3.4 Метод декомпозиции





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

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

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

3.3.5 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 *Tj* 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)

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

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

$$S_k = \sum_{1 \le i1 \le i2 \le \ldots \le ik \le l} P_r\left(E_{i1} \cap E_{i2} \cap \ldots \cap E_{ik}\right)$$

Неравенства Бонферрони

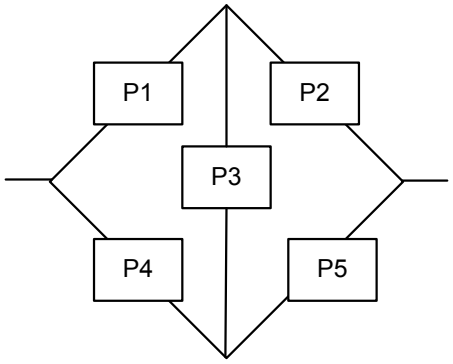
$$P \le S_1$$

$$P \ge S_1 - S_2$$

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

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

Метод включения-исключения (пример)



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

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

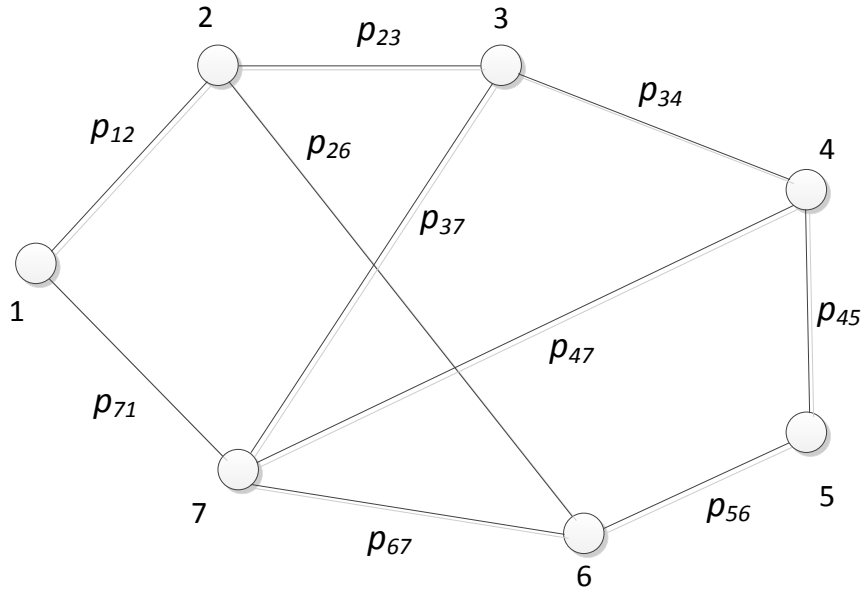
$$S_2 = P_r(MP_1 MP_2) + P_r(MP_1 MP_3) + P_r(MP_1 MP_4) + P_r(MP_2 MP_3) + P_r(MP_2 MP_4) + P_r(MP_3 MP_4) =$$
$$= p_1 p_2 p_4 p_5 + p_1 p_2 p_3 p_5 + p_1 p_2 p_3 p_4 + p_1 p_3 p_4 p_5 + p_2 p_3 p_4 p_5 + p_1 p_2 p_3 p_4 p_5$$

$$S_3 = P_r(MP_1 MP_2 MP_3) + P_r(MP_1 MP_2 MP_4) + P_r(MP_1 MP_3 MP_4) + P_r(MP_2 MP_3 MP_4) =$$
$$= 4 p_1 p_2 p_3 p_4 p_5$$

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

$$P = S_1 - S_2 + S_3 - S_4 = p_1 p_2 + p_4 p_5 + p_1 p_3 p_5 + p_2 p_3 p_4 - (p_1 p_2 p_5 + p_1 p_2 p_3 p_5 + p_1 p_2 p_3 p_4 +$$
$$+ p_2 p_3 p_4 p_5 + p_1 p_2 p_3 p_4 p_5) + 2 p_1 p_2 p_3 p_4 p_5$$

Самый надежный путь



$$p(P) = \prod_{(x_i, x_j) \in P} p_{ij}$$

$$\log p = \sum_{(x_i, x_j) \in P} \log p_{ij} = - \sum_{(x_i, x_j) \in P} \log C_{ij}$$

$$C_{ij} = -\log(p_{ij})$$

Трехместная операция, в алгоритме Флойда-Уоршалла

$$p_{ij} = \max\left\lfloor p_{ij}, p_{ik} + p_{kj} \right\rfloor$$

Измерение параметров трафика
1 Проведение измерений

Цель измерений трафика – получение сведений о характеристиках и параметрах процесса поступления и обслуживания заявок (вызовов или пакетов)

Цель измерений QoS – получение сведений о характеристиках процесса обслуживания заявок (вызовов или пакетов)

1. Объект измерений
2. Измеряемые параметры
3. План проведения измерений

2 Объект измерений

Например

-абонент (группа абонентов); - измерения абонентского трафика

-линия связи (пучок соед. линий); - измерения нагрузки СЛ

-направление связи; - измерения по направлениям связи

-коммутатор; - измерения на коммутаторе

др. сетевые элементы

3 Анализируемые параметры

Трафик:

-Интенсивность заявок (интенсивность вызовов, интенсивность поступления пакетов);

-Продолжительность обслуживания (продолжительность вызова, длина пакета);

-Интервалы времени между заявками;

-Зависимость нагрузки от времени;

-Зависимость продолжительности обслуживания от времени (суток, недели, сезона, года);

-Зависимость продолжительности обслуживания от вида услуги;

-Зависимость объема трафика от времени;

QoS

-вероятность потерь (коэффициент потерь);

-задержка доставки пакета данных;

-вариация задержки доставки пакета данных (джиттер);

-(коэффициент ошибок);

-время установления соединения;

-др. специфичные для услуги параметры.

…….

4 План проведения измерений

1. Время проведения измерений;
(дни, сезоны)

2. Продолжительность измерений;
(непрерывные, периодические)

3. Объем выборки;
(репрезентативность результата)

5 Обработка результатов

1.Точечные оценки

2. Интервальные оценки

3. Распределения

4. Тренды

5. Другие зависимости

5.1 Точечные оценки

$$X = \{x_1, x_2, \ldots x_n\}$$

1. Среднее значение $$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. Несмещенная (исправленная) дисперсия $$S^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

3. Несмещенное среднеквадратическое отклонение $\sigma_x = S_x$

4. Моменты случайной величины

Коэффициент вариации $$V = \frac{\sigma_x}{\bar{x}}$$

5.2 Интервальные оценки

1. Доверительный интервал для среднего

$$x = \bar{x} \pm \Delta \qquad \Delta = t_{n-1,1-\frac{\alpha}{2}} \cdot \frac{\sigma_x}{\sqrt{n}}$$

$$\widetilde{\Delta} = Z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma_x}{\sqrt{n}}$$

$Z_{1-\frac{\alpha}{2}}$ квантиль стандартного нормального распределения

2. Доверительный интервал для дисперсии

$$\frac{(n-1)\cdot S^2}{\chi^2_{1-\frac{\alpha}{2}}} \le D_x \le \frac{(n-1)\cdot S^2}{\chi^2_{\frac{\alpha}{2}}}$$

5.3 Доверительный интервал для вероятности

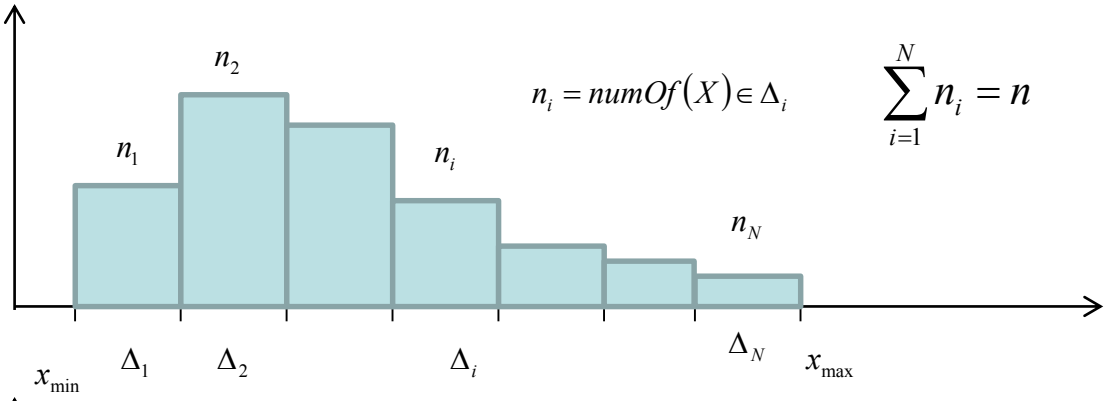
$$p = \bar{p} \pm \Delta \qquad \Delta = t_{n-1,1-\frac{\alpha}{2}} \cdot \sqrt{\frac{\bar{p} \cdot (1 - \bar{p})}{n}}$$

5.4 Гистограммы, функции распределения

$$X = \{x_1, x_2, \ldots x_n\} \qquad x_{\min} = \min\{x_1, x_2, \ldots x_n\} \qquad x_{\max} = \max\{x_1, x_2, \ldots x_n\}$$

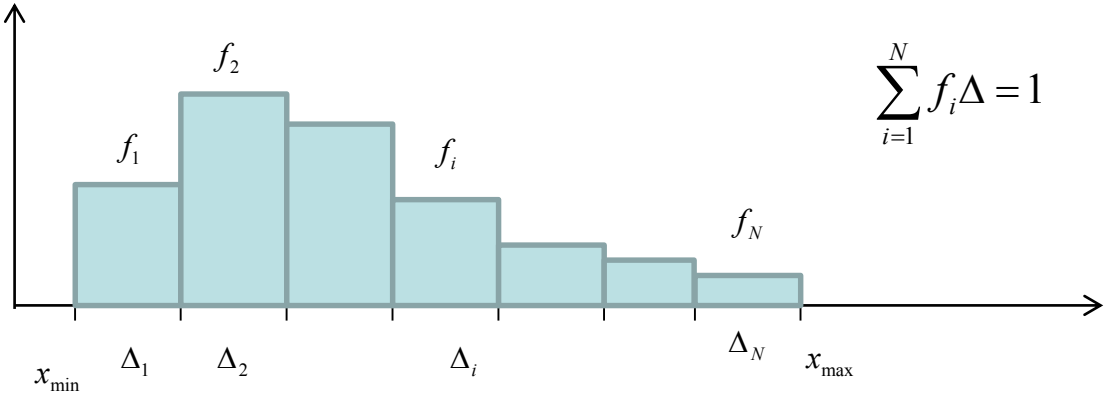
$$\Delta = \frac{x_{\max} - x_{\min}}{N}$$

ширина интервала «кармана»



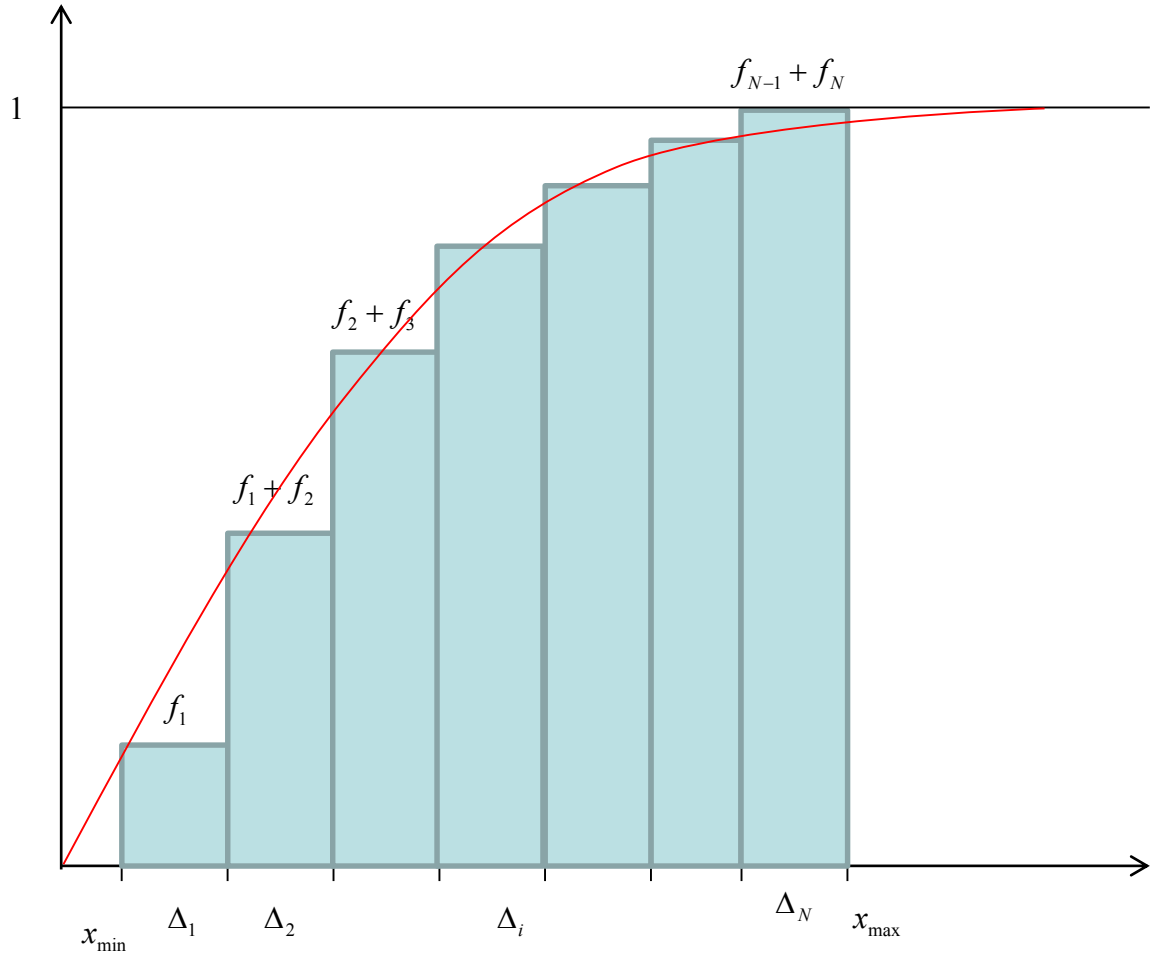
$$n_i = numOf(X) \in \Delta_i \qquad \sum_{i=1}^{N} n_i = n$$

$$f_i = \frac{n_i}{n \cdot \Delta}$$



$$\sum_{i=1}^{N} f_i \Delta = 1$$

5.5 Функция распределения



Имитационное моделирование (надежность)

$\{p_1 \dots p_N\}$ Надежность элементов сети

N Число элементов сети

На каждом цикле генерируется N случайных чисел с равномерным законом распределения $\{\xi_1 \dots \xi_N\}$

$0 \le \xi_i < 1$

$$\bar{p} = \frac{n_1}{n}$$

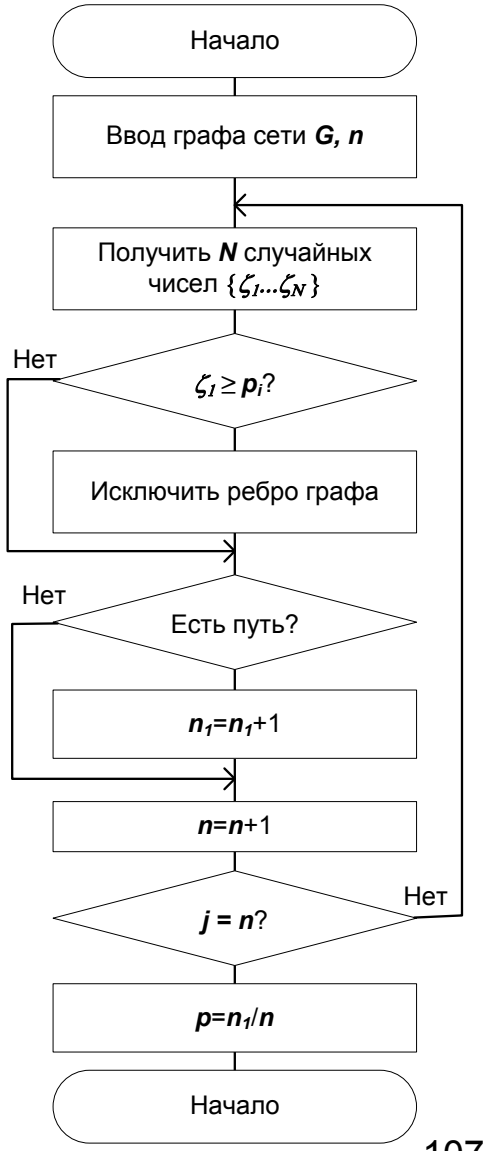
$$p = \bar{p} \pm t(\alpha, r)\sqrt{\frac{\bar{p}(1-\bar{p})}{n}}$$

$$n = \frac{1-p}{\delta^2 p}$$

Например, при $\delta < 5 \cdot 10^{-6}$ $p_i = 0{,}99$ $i = 1 \dots n$

$p = 0{,}99998$

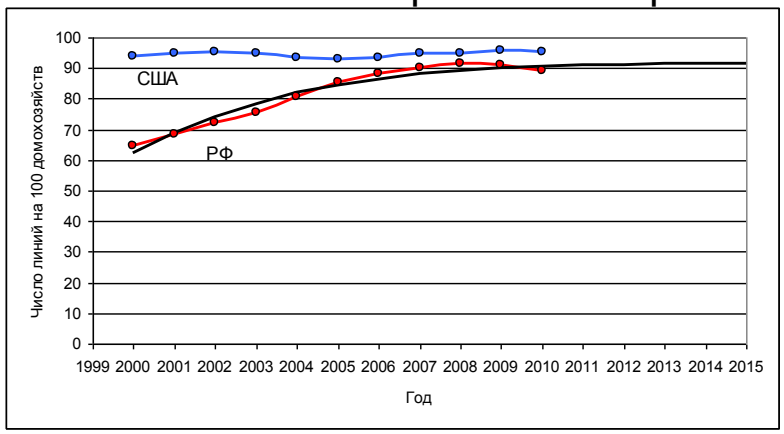
$n > 3.2 \cdot 10^6$



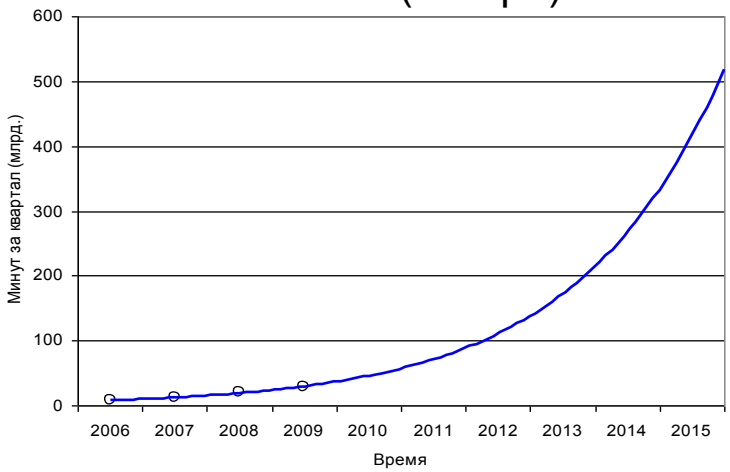
Задачи прогнозирования

Задачи прогнозирования (примеры)

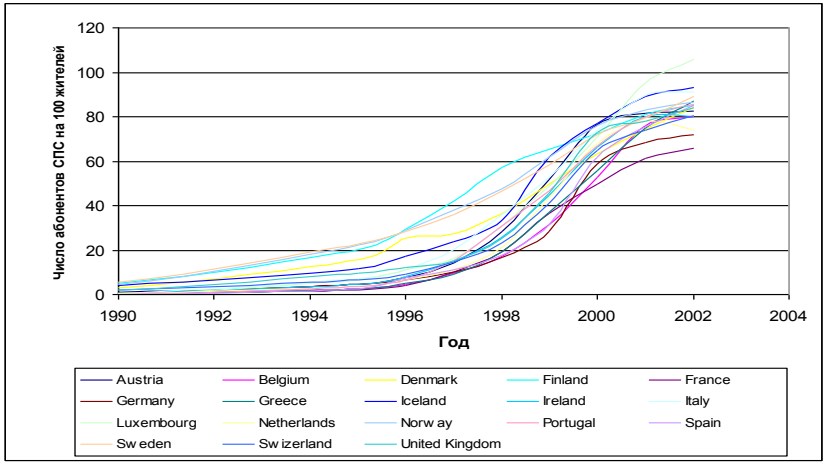
Фиксированная ТфОП



VoIP (в мире)

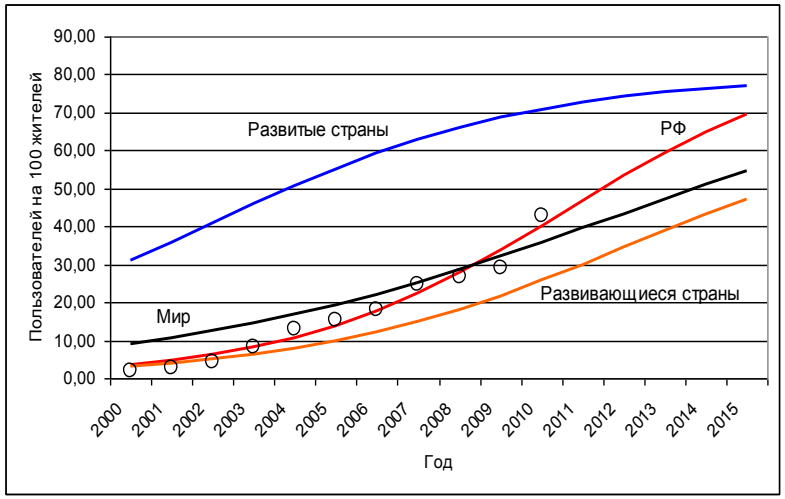


ССПС в Европе

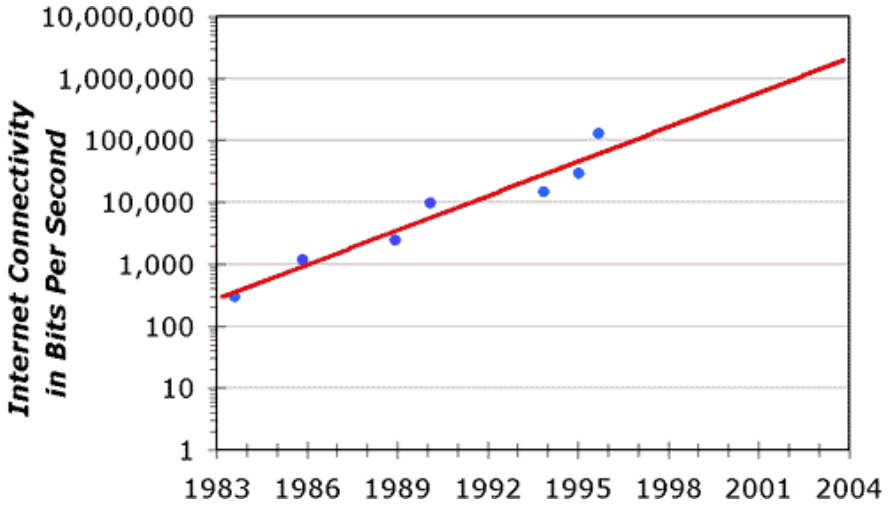


Задачи прогнозирования (примеры)

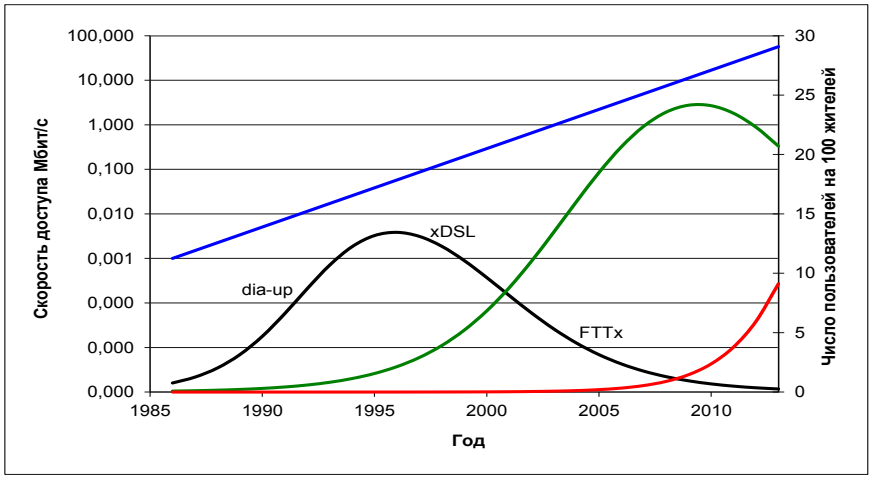
Интернет



Скорость (Закон Нильсена)



Прогноз технологий доступа



Ассоциативный метод

1. Сбор статистических данных о развитии технологии в стране или регионе

$$Y = (y_1, y_2, \dots y_i \dots, y_n)$$

2. Сбор статистических данных о развитии данной технологии или близких технологий, достигших более поздней стадии развития

(за рубежом или в других регионах)

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \dots & \dots & x_{ij} & \dots \\ x_{m1} & x_{m2} & \dots & x_{mk} \end{bmatrix}; \quad k > n$$

3. Выбор наиболее близкого тренда развития на рассматриваемом интервале

$$\min_{i=1\dots m} \sum_{j=1}^{n} (x_{ij} - y_j)^2 \quad \rightarrow i = g \rightarrow \quad X_g = (x_{g1}, x_{g2}, \dots x_{gj} \dots, x_{gn})$$

4. Анализ демографических характеристик страны или региона

$$N, \quad \rho$$

5. Выбор аналитической модели и оценка ее параметров на основе полученных данных

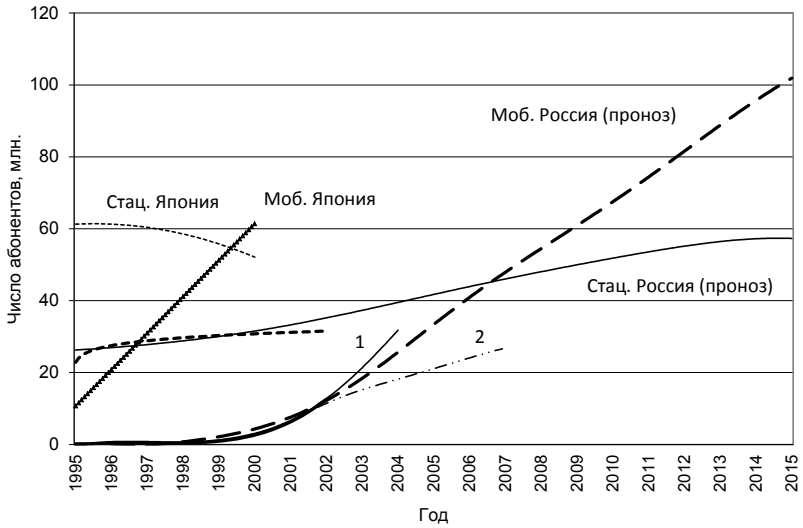
$$n(t) = \frac{N}{1 - e^{-\frac{t - t_0}{b}}}; \quad N, \quad t_0, \quad b$$

6. Получение прогноза развития

$$D = (d_1, d_2, \dots d_i \dots, d_H); \quad H \geq n$$

Результаты(пример)

1. В 2000 г. было спрогнозировано превышение числом сотовых абонентов числа жителей РФ.
2. Относительная ошибка прогноза на 15 лет – в пределах 30%.



Аналитические модели
1. Линейная регрессия

$$\hat{y} = f(b_0, b_1, t) = b_1 \cdot t + b_0$$

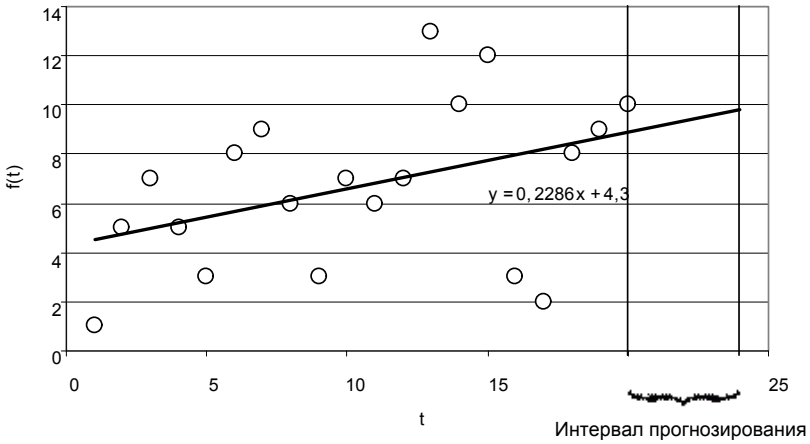
$$b_1 = \frac{\sum_{i=1}^{n} x_i y_i - n \cdot \bar{x} \cdot \bar{y}}{\sum_{i=1}^{n} x_i^2 - n \cdot \bar{x}^2}$$

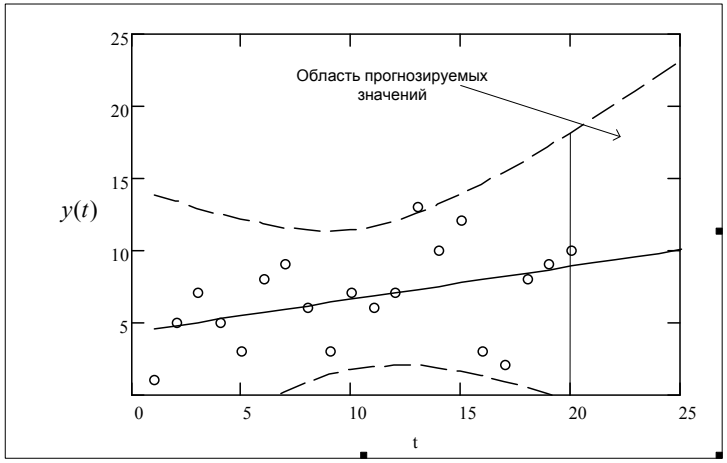
$$b_0 = \bar{y} - b_1 \bar{x}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$y(x_0) = \hat{y}(x_0) \pm S_{y|x}(x_0) t_{n-2,\alpha/2}$$

$$S_{y|x}(x_0) = \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \sqrt{\frac{1}{n-2}\left(\sum_{i=1}^{n}(y_i - \bar{y})^2 - \frac{\left(\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right)}$$





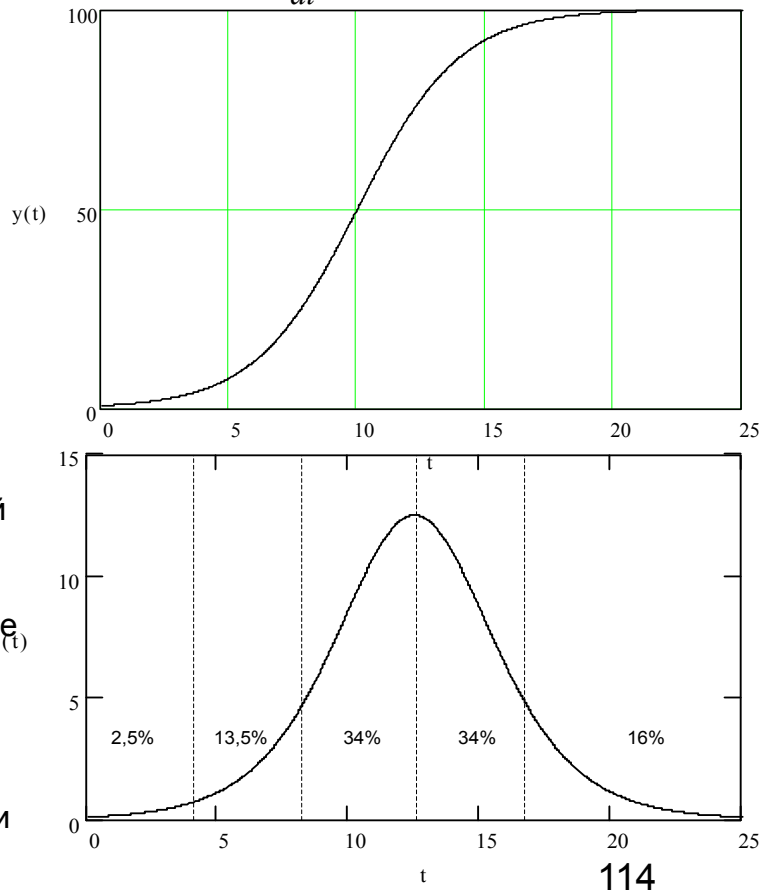
Аналитические модели
2. Логистическая регрессия

$$\frac{dy}{dt} = r \cdot y \cdot \left(1 - \frac{y}{K}\right) \qquad y = y(t) \qquad y(t) = \frac{K \cdot y_0 \cdot e^{rt}}{K + y_0\left(e^{rt} - 1\right)} \qquad \lim_{t \to \infty} y(t) = K$$

Частным случаем логистической функции является S-функция, как решение для $\frac{dy}{dt} = y \cdot (1 - y)$

$$y(x) = \frac{1}{1 + e^{-t}}$$

$$y(t) = \frac{K}{1 + e^{-\frac{t - t_0}{b}}}$$



В 1962 г. Е.М. Роджерсом было предложено использование S-кривой для описания процесса внедрения инноваций (Diffusion of Innovations).

1. Новаторы.
2. Начальный выбор. распространение технологии среди пользователей наиболее быстро откликающихся на предложение (early adopters).
3. Раннее большинство. На данном этапе происходит распространение технологии среди половины основной массы пользователей, наиболее быстро принимающих предложение (early majority).

4. Позднее большинство. На данном этапе происходит распространение технологии среди второй половины основной массы пользователей (late majority).

5. Отстающие. На данном этапе происходит распространение технологии среди пользователей наиболее медленно реагирующих на предложение (laggards).

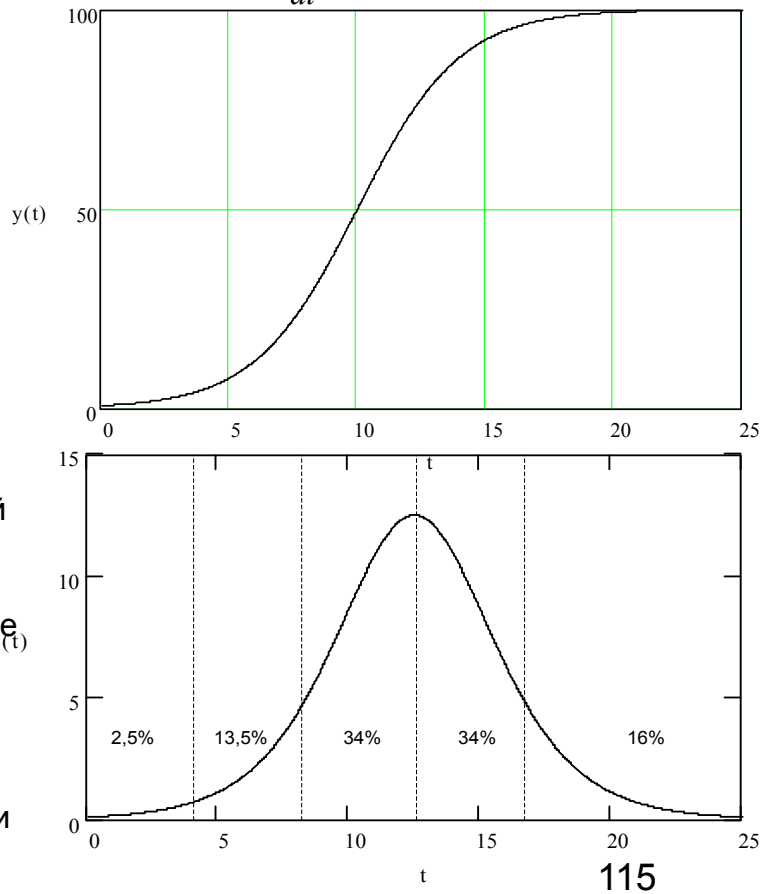
Аналитические модели
2. Логистическая регрессия

$$\frac{dy}{dt} = r \cdot y \cdot \left(1 - \frac{y}{K}\right) \qquad y = y(t) \qquad y(t) = \frac{K \cdot y_0 \cdot e^{rt}}{K + y_0\left(e^{rt} - 1\right)} \qquad \lim_{t \to \infty} y(t) = K$$

Частным случаем логистической функции является S-функция, как решение для $\frac{dy}{dt} = y \cdot (1 - y)$

$$y(x) = \frac{1}{1 + e^{-t}}$$

$$y(t) = \frac{K}{1 + e^{-\frac{t - t_0}{b}}}$$



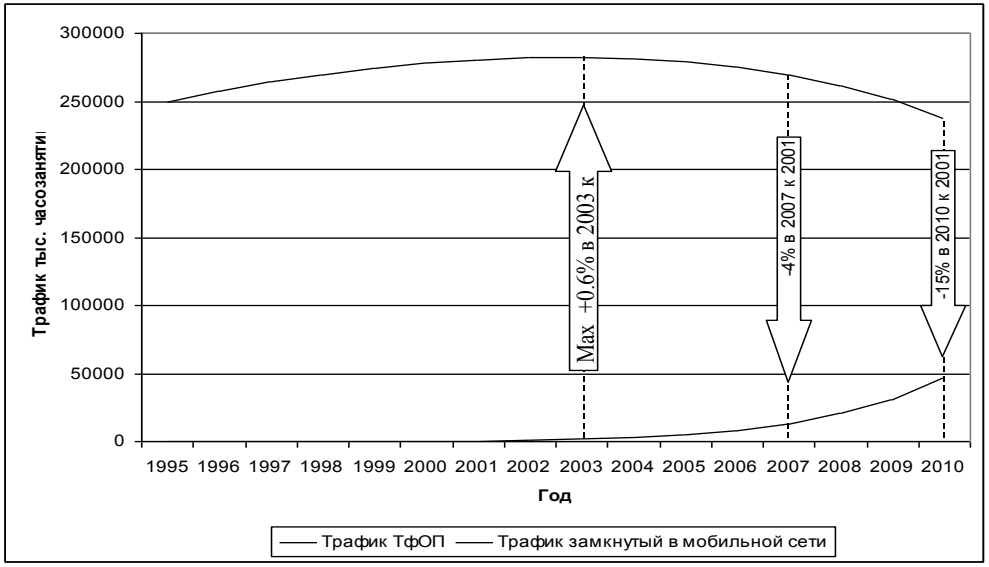
В 1962 г. Е.М. Роджерсом было предложено использование S-кривой для описания процесса внедрения инноваций (Diffusion of Innovations).

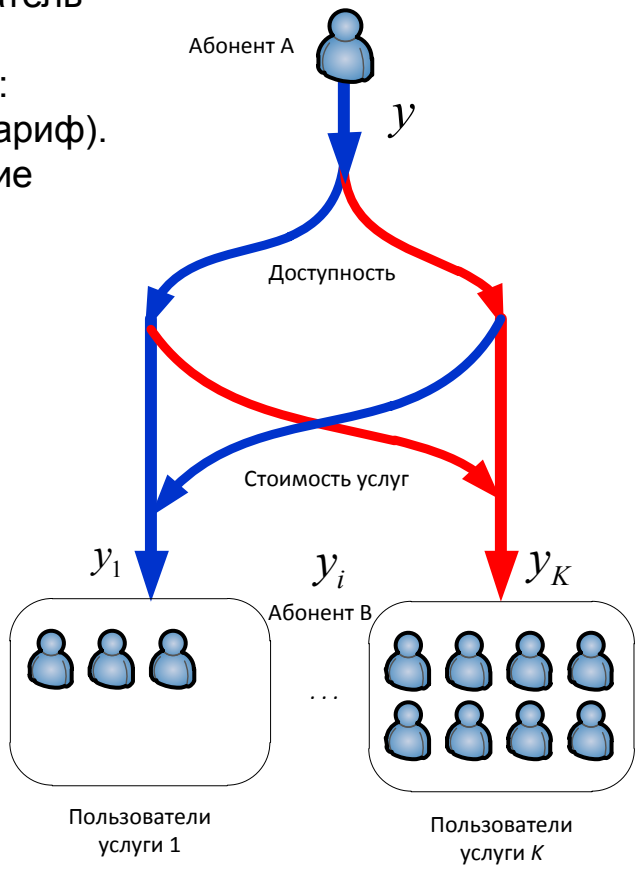
1. Новаторы.
2. Начальный выбор. распространение технологии среди пользователей наиболее быстро откликающихся на предложение (early adopters).
3. Раннее большинство. На данном этапе происходит распространение технологии среди половины основной массы пользователей, наиболее быстро принимающих предложение (early majority).
4. Позднее большинство. На данном этапе происходит распространение технологии среди второй половины основной массы пользователей (late majority).
5. Отстающие. На данном этапе происходит распространение технологии среди пользователей наиболее медленно реагирующих на предложение (laggards).

Миграция трафика

Модель миграции основана на следующих положениях:
1. Трафик, производимый пользователями распределяется между технологиями связи;
2. Доля трафика распределяемого на некоторую технологию пропорциональна доле ее пользователей и доступности услуги;
3. При выборе технологии для реализации услуги связи пользователь учитывает стоимость услуги.

Метод оценки миграции трафика учитывает такие факторы как:
1. Стоимость единицы информации, передаваемой через сеть (тариф).
2. Доступность терминалов (услуг) связи (Доля времени, в течение которой доступна услуга (терминал)).
3. Число абонентов.





Миграция на примере OTT сервисов

Если вероятность того, что представление о соотношении цена/качество пользователя меньше некоторой величины *x* равна

$$p(X < x) = f(x) = 1 - e^{-\bar{c} \cdot x} \qquad p_S(x) = 1 - p(X < x) = e^{-\bar{c} \cdot x}$$

$$p_i = \frac{n_i}{\sum_{i=1}^{k} n_i}; \quad \sum_{i=1}^{k} p_i = 1 \qquad S_i = \frac{p_S(x_i) \cdot p_i}{\sum_{j=1}^{k} p_S(x_j) \cdot p_j}$$

При наличии альтернативного выбора способов предоставления услуги, например голосовой связи (телефонии) в виде основной услуги оператора связи и услуги OTT сервиса (мессенджера) приведенная выше модель трафика будет иметь вид

$$y_B = Y \cdot S_B = Y \cdot \frac{p_S(x_B) \cdot p_B}{p_S(x_B) \cdot p_B + p_S(x_{OTT}) \cdot p_{OTT}}$$

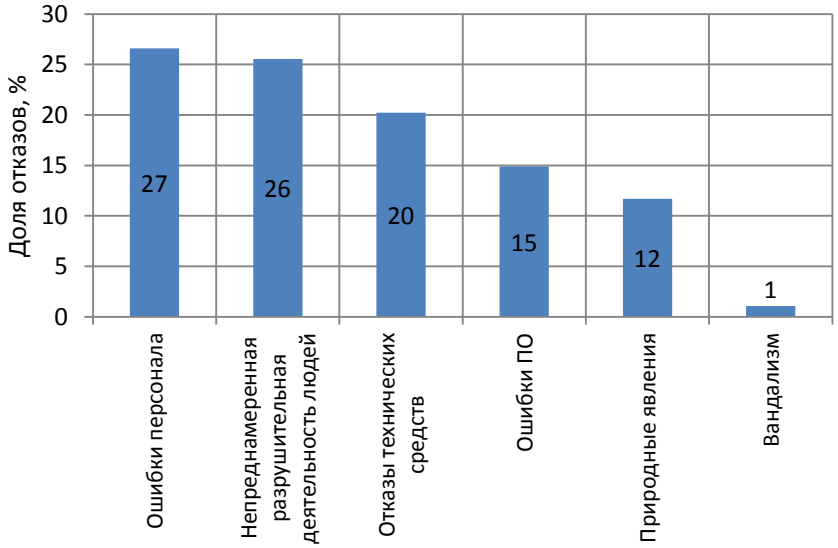
$$y_{OTT} = Y \cdot S_{OTT} = Y \cdot \frac{p_S(x_{OTT}) \cdot p_{OTT}}{p_S(x_B) \cdot p_B + p_S(x_{OTT}) \cdot p_{OTT}}$$

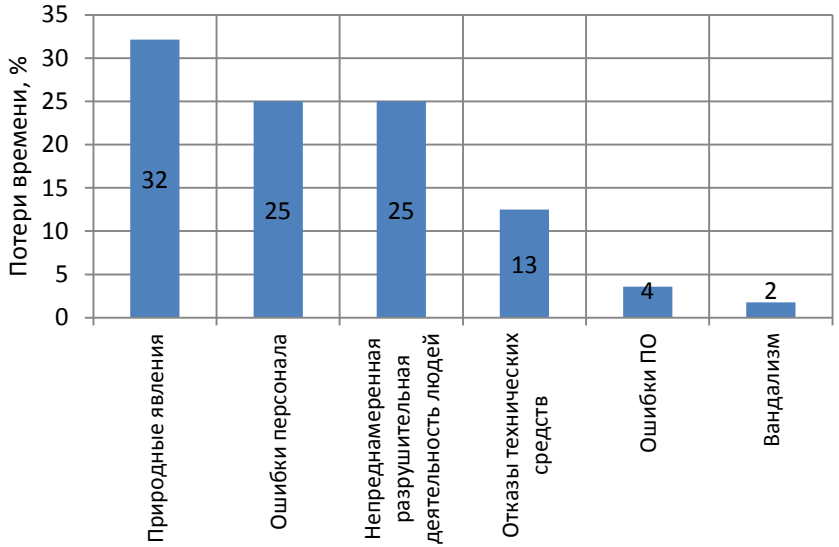
$$y_i = y \cdot S_i$$

Иллюстрации (надежность сетей связи)

Alcatel-Lucent Bell Labs

Уязвимости системы телекоммуникаций



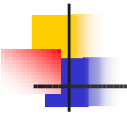


США (FCC)

N	Группа факторов	Доля от общего количества отказов, %	Потери пользователь-ского 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

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.