# Дискретная оптимизация

МФТИ, весна 2018

Александр Дайняк

www.dainiak.com

# Общая постановка задач оптимизации

**Дано:**

- $\mathcal{S}$ — заданное множество (область поиска)
- $f$ — функция на множестве $S$

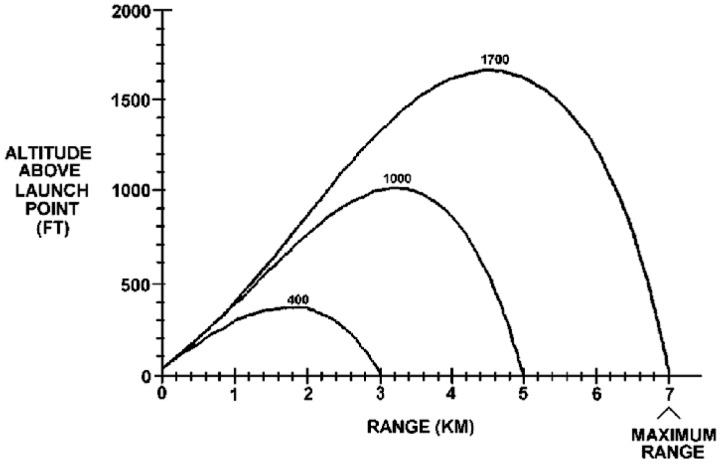
**Найти:**

- $x \in \mathcal{S}$, такой, что

$$\forall y \in \mathcal{S} \quad f(y) \leq f(x)$$

# Непрерывная оптимизация

- Обычно:
  - $\mathcal{S}$ — числовое множество, метрическое пространство, …
  - $f$ — выпуклая/непрерывная/гладкая функция
- Выгодные особенности:
  - $\mathcal{S}$ и $f$ часто задаются явно
  - По множеству $\mathcal{S}$ можно «непрерывно перемещаться»



# Дискретная оптимизация

**Особенность:**

- Множество $S$ конечно или счётно

**Трудности:**

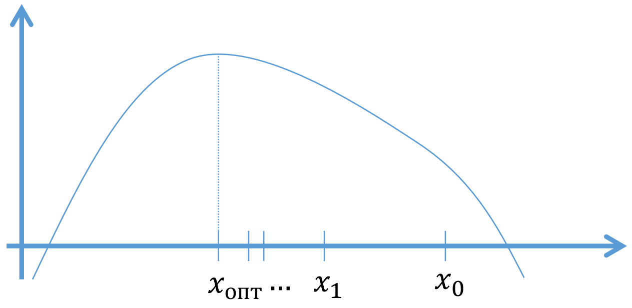
- Нельзя «сдвинуться на сколь угодно малое $\varepsilon$» при поиске
- Множество $S$ может быть сложно/неявно задано
- Функция $f$, как правило, задана неявно, её вычисление в точке требует времени

# Локальный поиск (local search)

- Ищем не глобальный, а *локальный* оптимум (надеемся, что он окажется глобальным)
- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее» (точка, в которую двигаемся, берётся из небольшой окрестности текущей точки; отсюда *локальность*)

# Локальный поиск

- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее»
- Отлично работает в выпуклой оптимизации:



# Локальный поиск

- Отправляемся из произвольной стартовой точки, помалу сдвигаясь туда, где «теплее»
- В дискретной оптимизации всё сложнее:
  - Не можем сдвигаться на «сколь угодно малое $\varepsilon$»
  - Не всегда очевидно, как определять окрестность точки
  - Функции задаются сложно (обычно как суммы специального вида). Неясно, что такое «выпуклая функция».

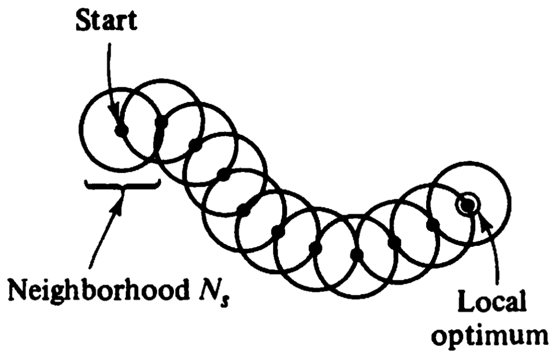
# Локальный поиск: общий алгоритм

- Минимизируем целевую функцию $f$ на множестве $\mathcal{S}$.
- Считаем, что задана *окрестностная функция:*

$$\mathcal{N}: \quad \mathcal{S} \to 2^{\mathcal{S}}$$

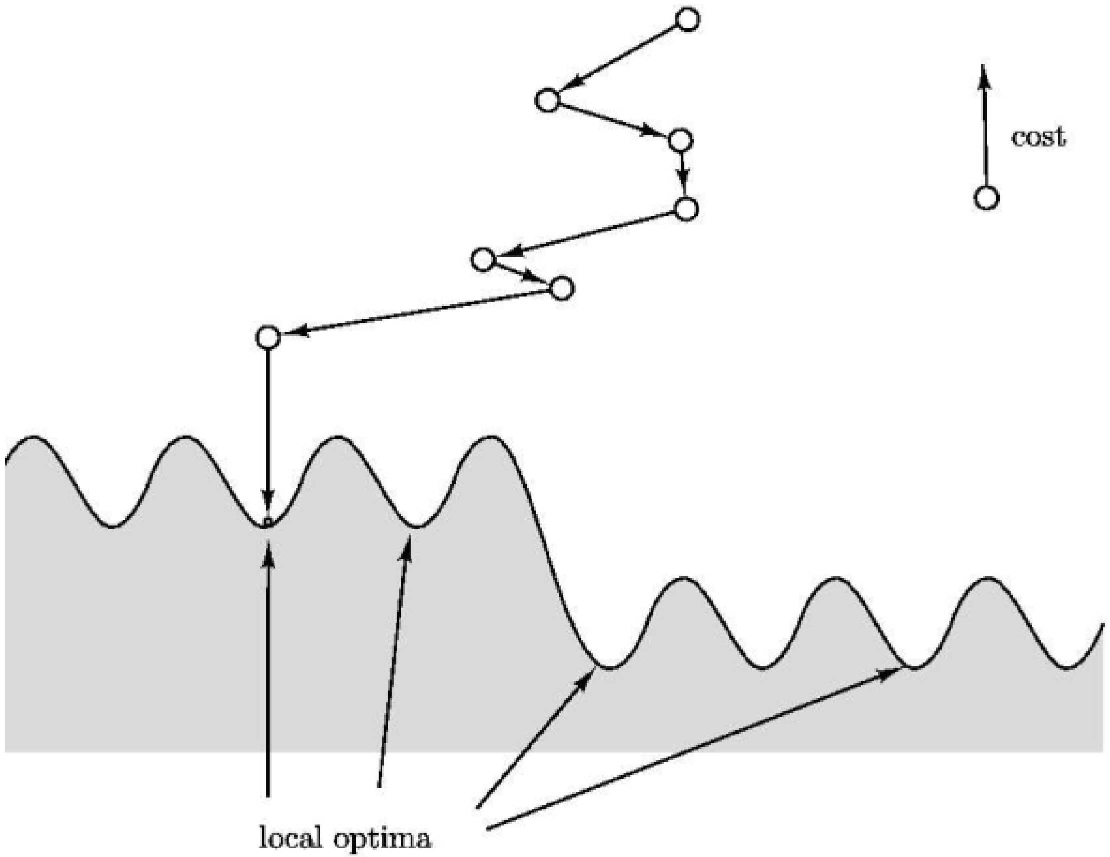
  (также говорят, что задана *система окрестностей*).
- Алгоритм локального поиска (в задаче минимизации):
  1. $x$ := random($\mathcal{S}$)
  2. if $\exists y \in \mathcal{N}(x)$: $f(y) < f(x)$ then $x$ := $y$, goto 2
  3. return($x$)

# Локальный поиск




- Минимизируем целевую функцию $f$ на множестве $\mathcal{S}$.
- Система окрестностей
  - *сильно связная*, если
    $$\forall x', x'' \in \mathcal{S} \quad \exists x_1, x_2, \dots, x_k\colon \quad x_{i+1} \in \mathcal{N}(x_i), x_1 = x', x_k = x'',$$
    то есть из любой точки $\mathcal{S}$ можно попасть в любую другую, перемещаясь по окрестностям
  - *точная*, если, начав из любого начального приближения, алгоритм локального поиска находит глобальный оптимум
  - *полиномиально обозримая (polynomially searchable)*, если для любого $s \in \mathcal{S}$ существует полиномиальный алгоритм для выбора наилучшего элемента в $\mathcal{N}(s)$

# Проблема локальных оптимумов



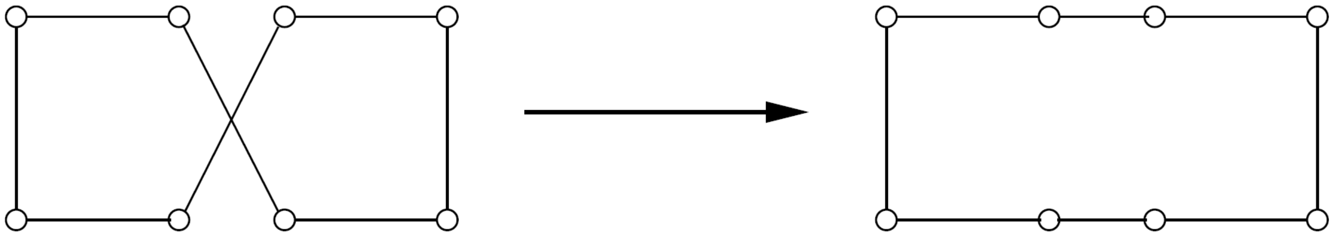
# Локальный поиск в задаче TSP

В задаче TSP:

- $\mathcal{S}$ — множество всех гамильтоновых циклов графа
- $f(H) = \sum_{e \in H} w(e)$

Как определить окрестностную функцию?

- Гамильтоновы циклы «близки», если у них много общих рёбер.
- Удаляем из цикла два ребра, добавляем два новых — получаем другой цикл

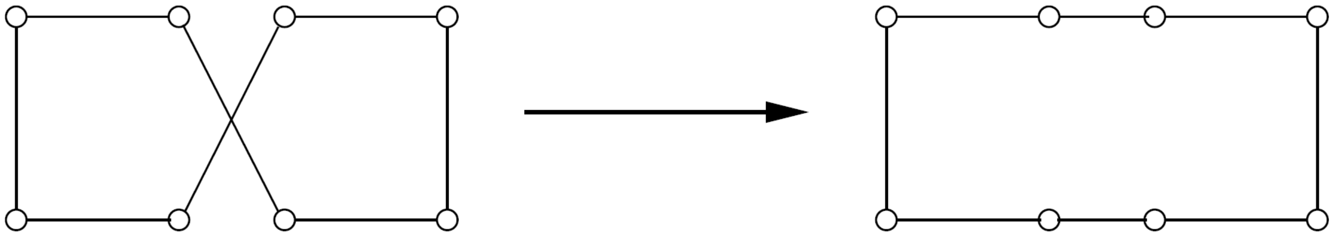


# Локальный поиск в задаче TSP

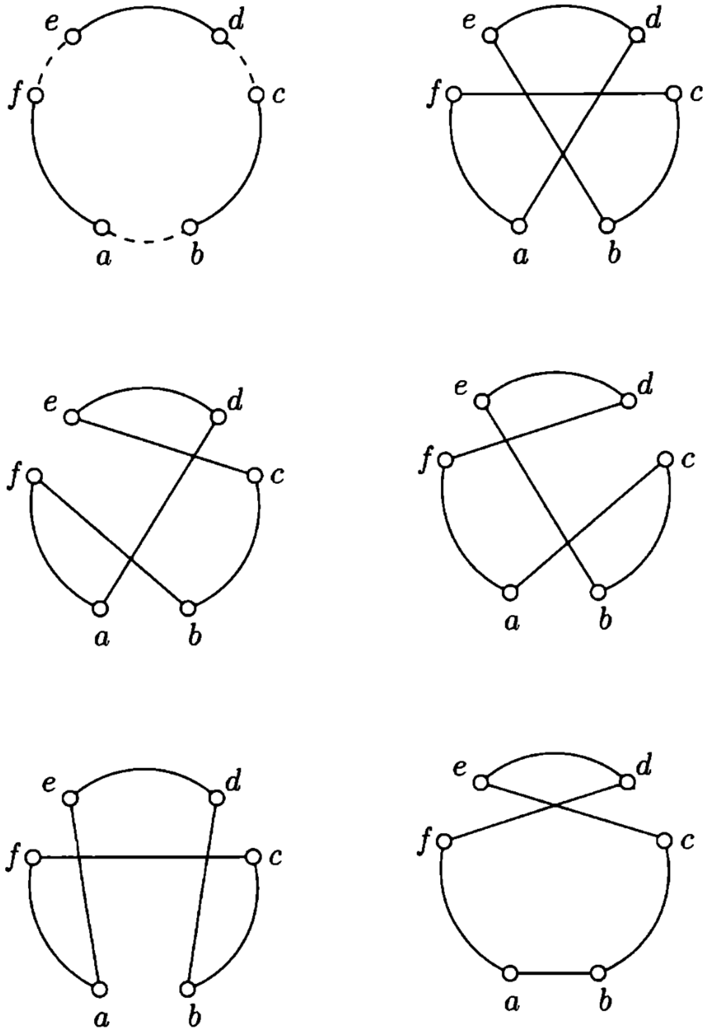
В задаче TSP:

- $\mathcal{S}$ — множество всех гамильтоновых циклов графа
- $f(H) = \sum_{e \in H} w(e)$

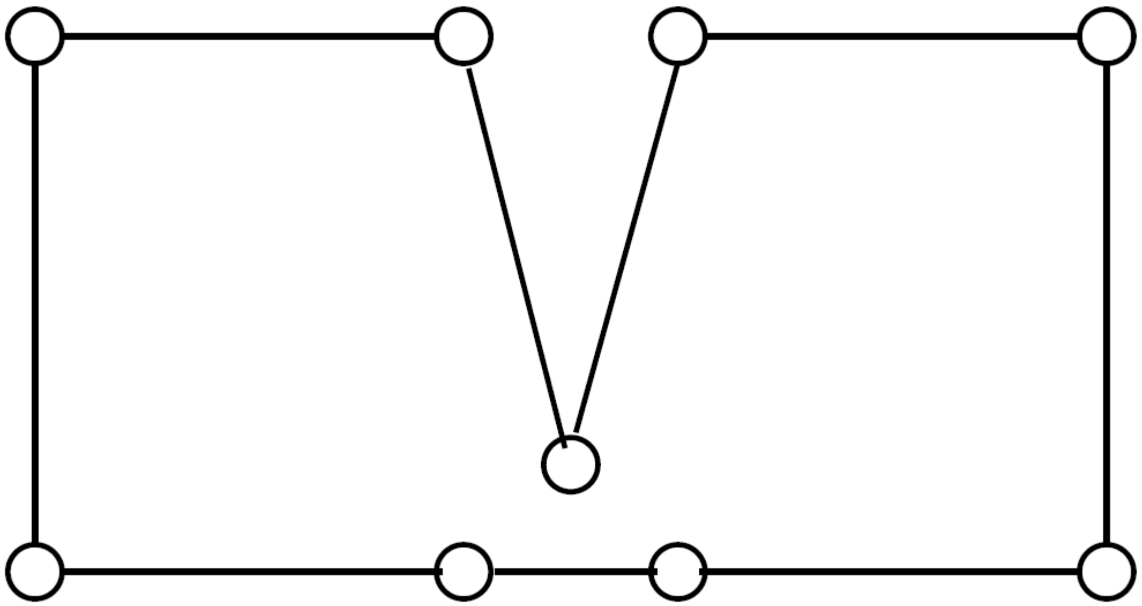
$k$-окрестность $\mathcal{N}_k(H)$ цикла $H$ — множество всех циклов, получаемых из $H$ удалением-добавлением $k$ рёбер



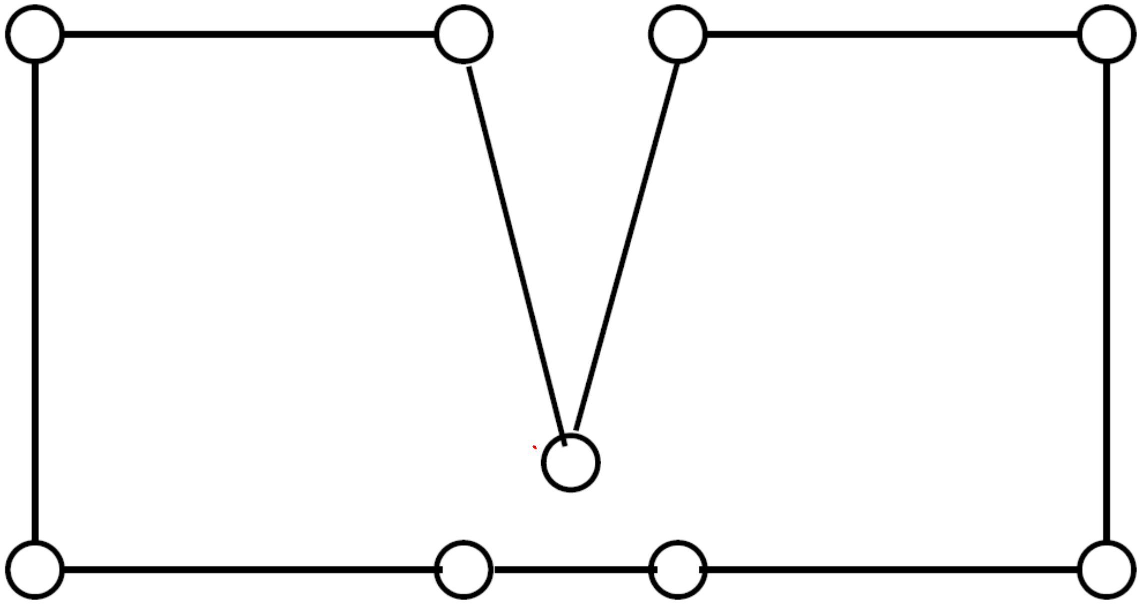
# Возможные варианты 3-замены



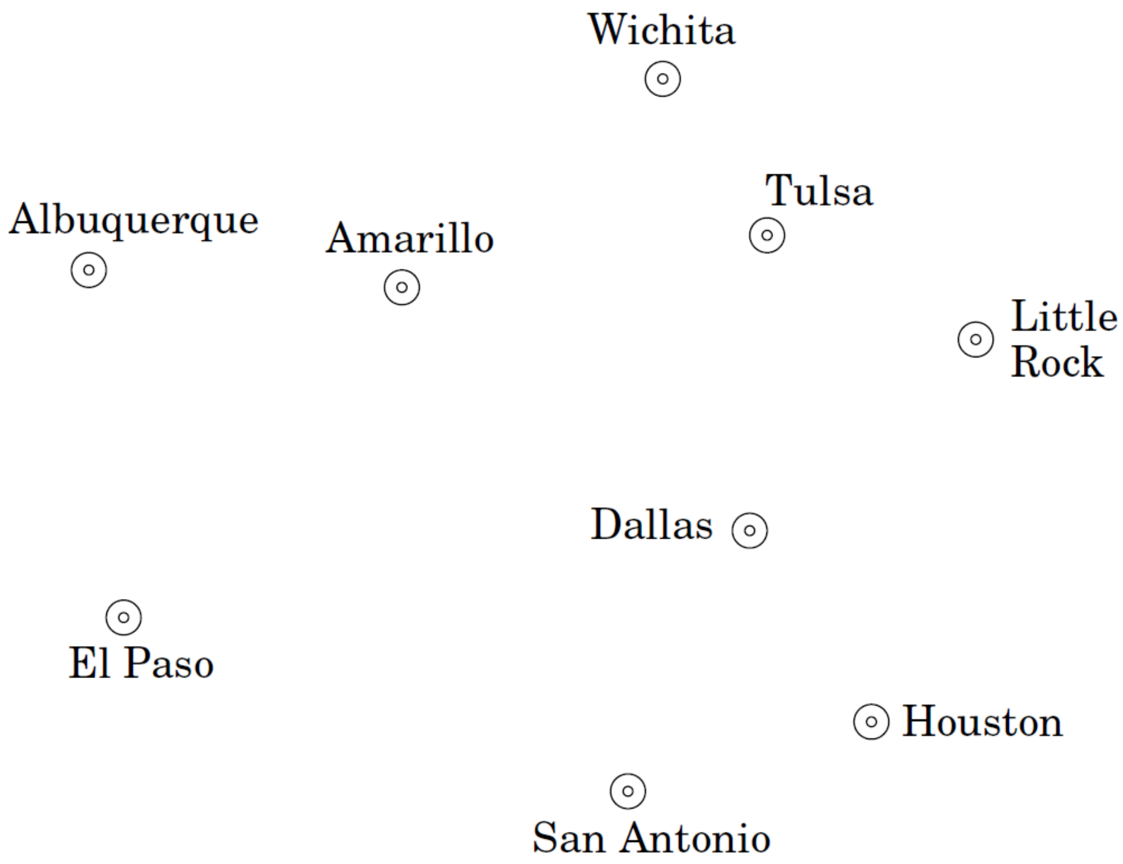
# Является ли система 2-окрестностей корректной?



# Поможет ли здесь система 3-окрестностей?

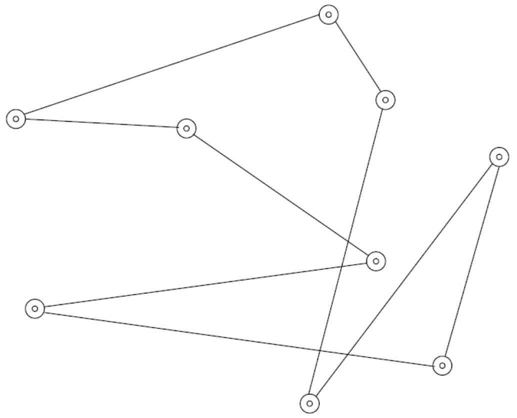


# Пример работы 3-окрестностей

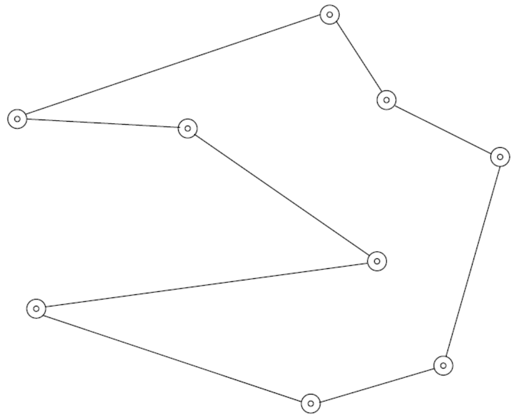


# Пример работы 3-окрестностей

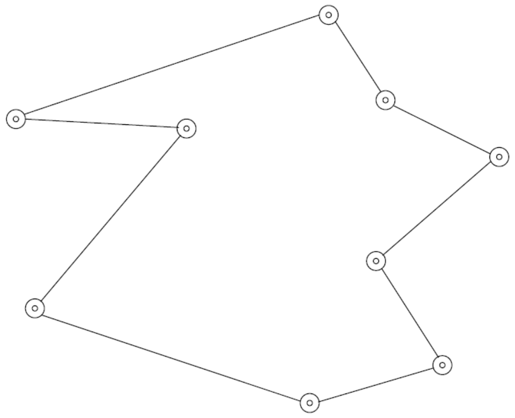
(i)



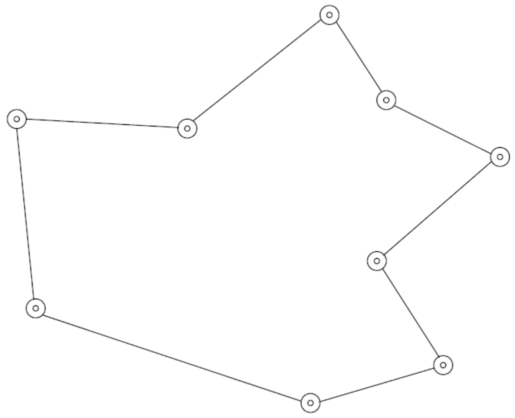
(ii)



(iii)



(iv)



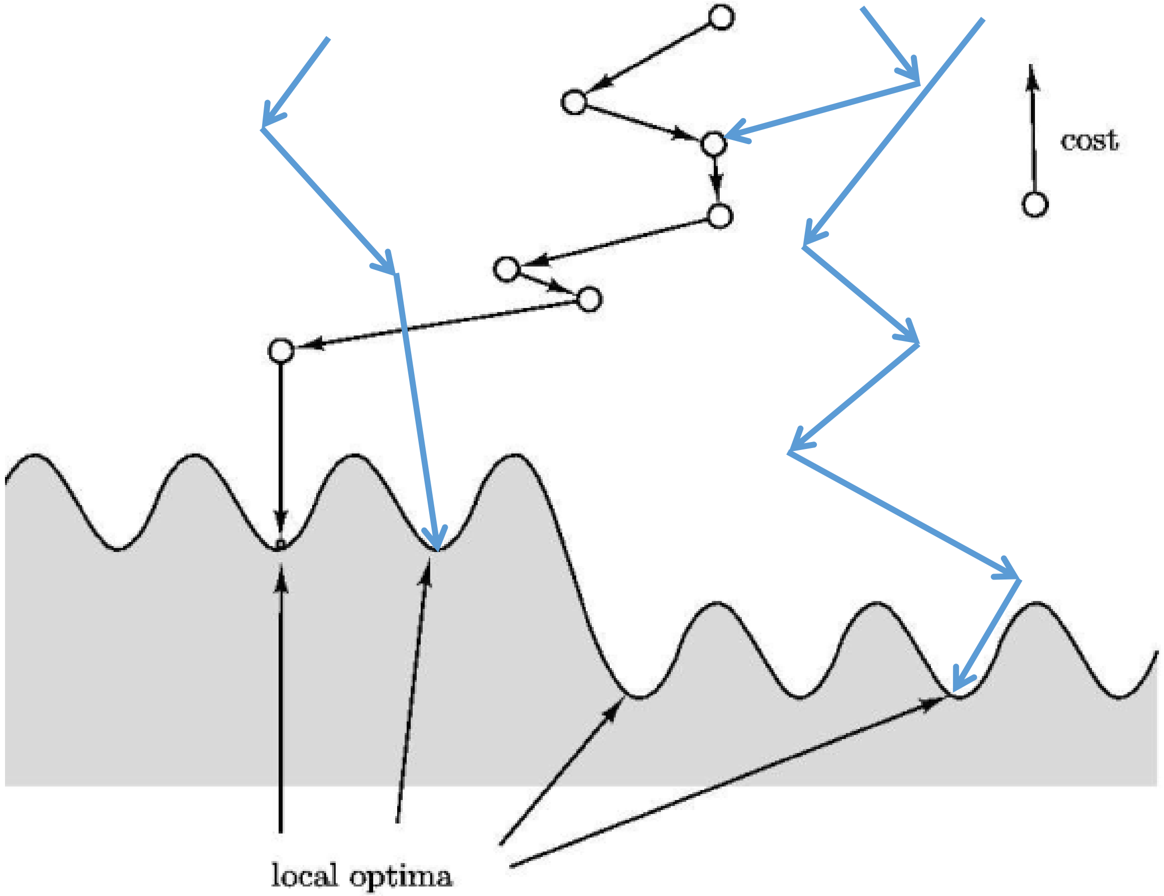
# Локальный поиск: pro et contra

- Очевидное достоинство локального поиска — идейная простота реализации.
- По сути, вся сложность организации хорошего локального поиска в задании хорошей окрестностной функции.
- В сложных задачах не стоит надеяться *только* на стандартный локальный поиск.
  (Замечание: в задаче TSP даже использование $(n-3)$-окрестности не помогает.)

# Борьба с застреванием в локальных оптимумах

- Случайное начальное приближение + множественные запуски
- Переменная глубина: эвристика Кернигана—Лина
- Имитация отжига
- Табу-поиск

# Случайное начальное приближение + множественные запуски



# Случайное начальное приближение + множественные запуски

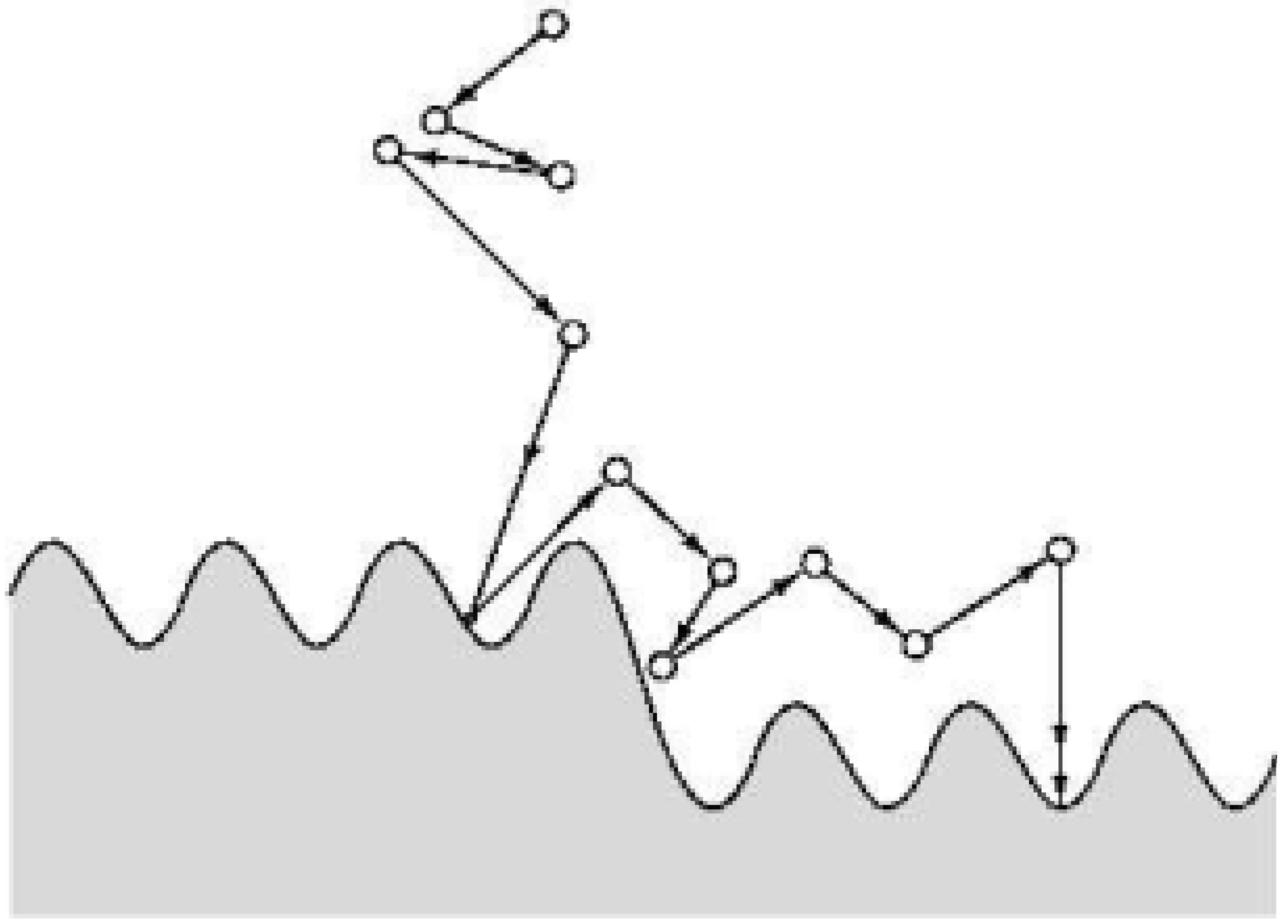
Преимущества:

- Простейшая надстройка над локальным поиском
- Если локальных оптимумов немного, подход работает

Недостатки:

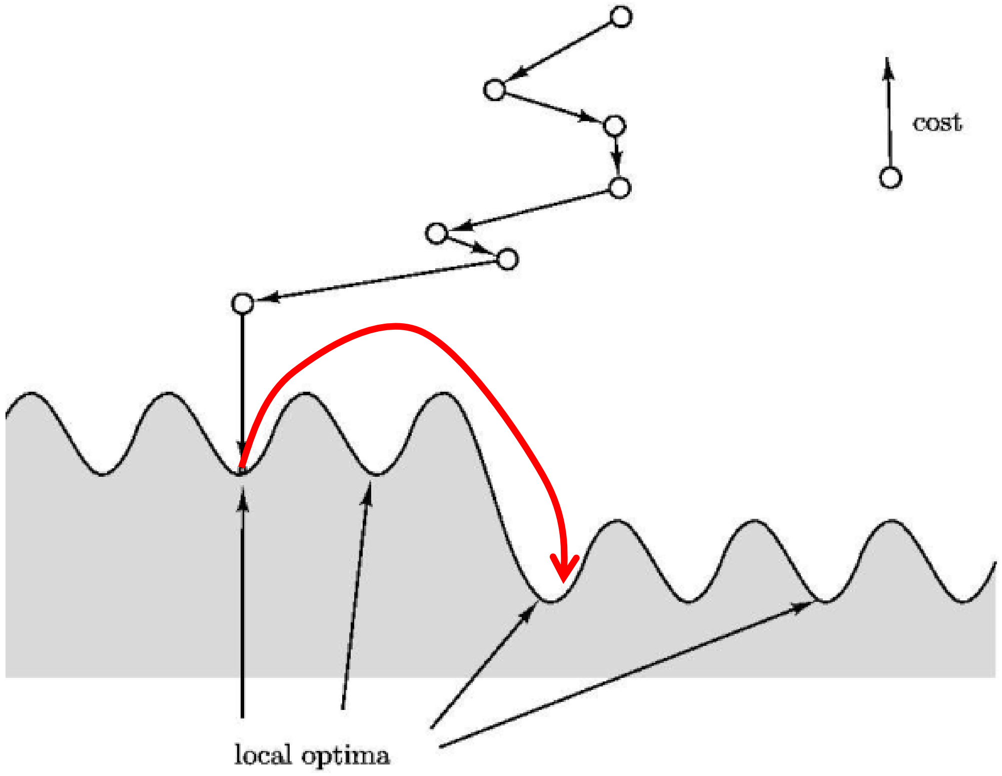
- Локальных оптимумов может быть очень (экспоненциально) много
- Непредсказуемость работы из-за рандомизации

# Невозможная траектория для локального поиска:



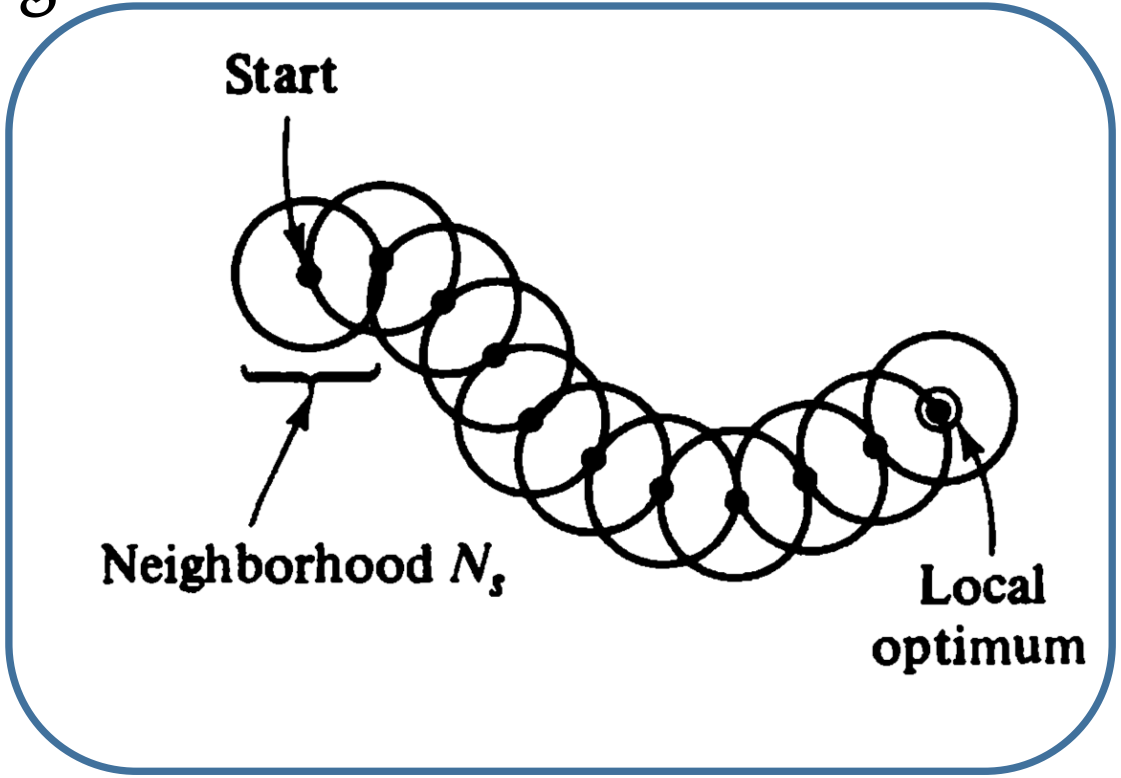
# Локальный поиск переменной глубины (variable depth search)

- Впервые предложен B.W. Kernigan, S. Lin ‘1970 для задачи о разбиении графа (graph partitioning).

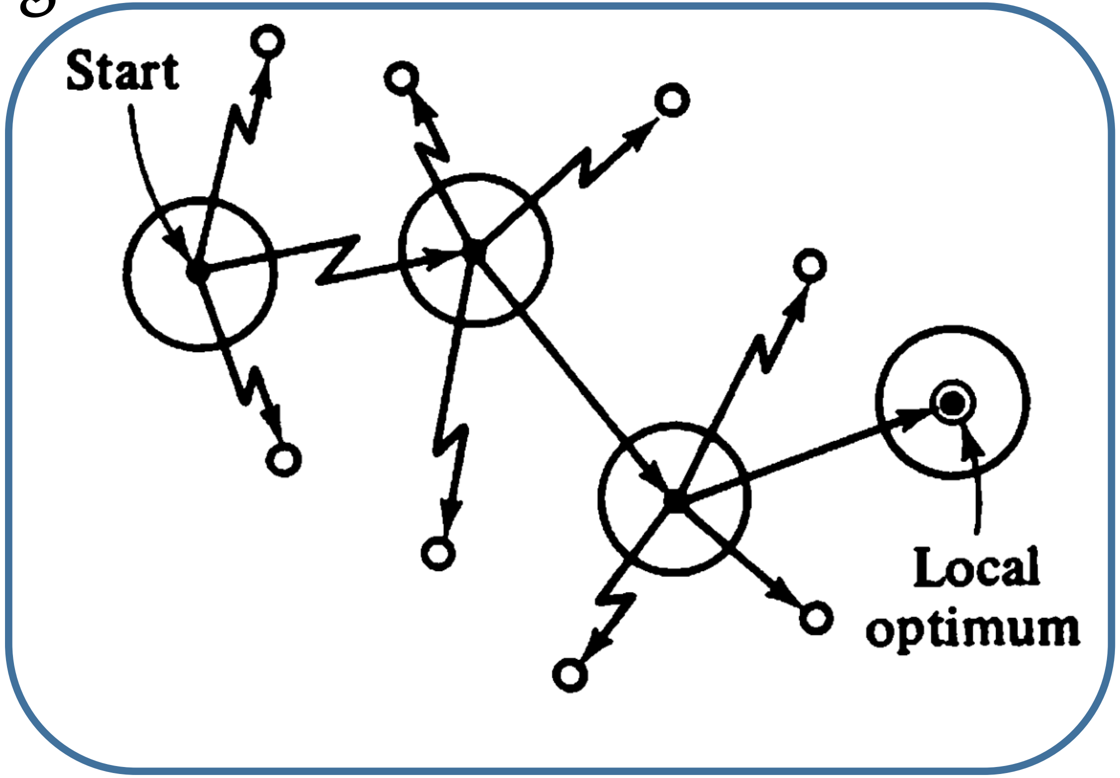


# Керниган—Лин vs. обычный локальный поиск

$\mathcal{S}$




$\mathcal{S}$



# Задача о разбиении графа

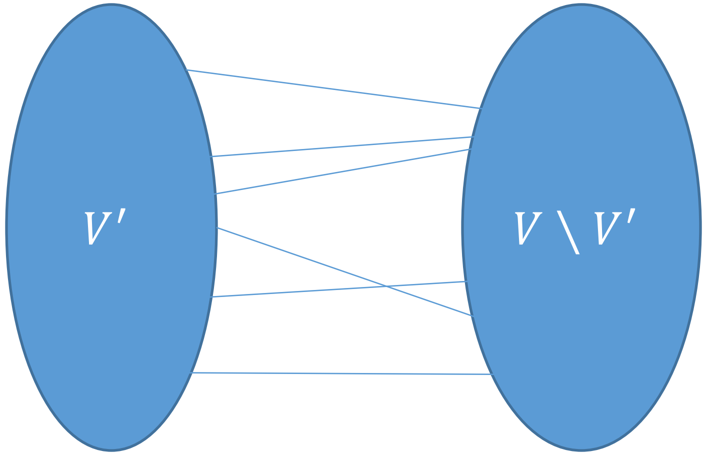
**Дано:**

- Граф $G = (V, E)$ с весами на рёбрах.
- Число $\alpha \in \left(0, \frac{1}{2}\right]$

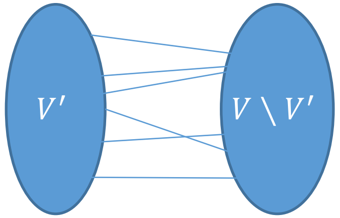
**Найти:**

- $V' \subseteq V$, такое, что $\alpha \leq \frac{|V'|}{|V|} \leq 1 - \alpha$ и

$$\sum_{\substack{u \in V' \\ v \in V \setminus V'}} w(uv) \to \min$$



# Задача о разбиении графа




Подход с помощью локального поиска с очевидной окрестностью:

$$\mathcal{N}_k(V') \coloneqq \left\{ \widetilde{V'} \mid \left|\widetilde{V'}\right| = |V'| \text{ и } \left|\widetilde{V'} \mathbin{\Delta} V'\right| \leq 2k \right\}$$

При фиксированном $k$ имеем

$$|\mathcal{N}_k(V')| = \Theta\left(\binom{n/2}{k}^2\right) = \Theta\left(n^{2k}\right)$$

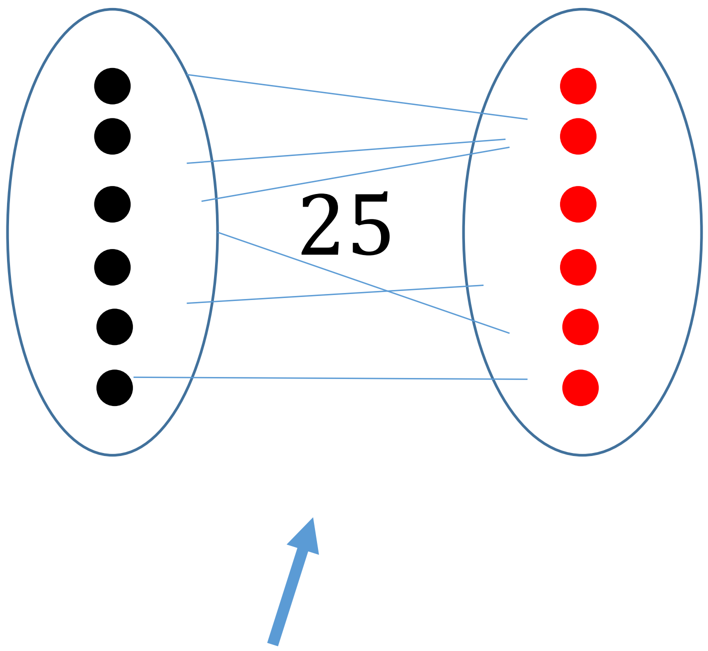
— даже при малых $k \geq 2$ это уже много!

# Задача о разбиении графа: Керниган—Лин

Идея:

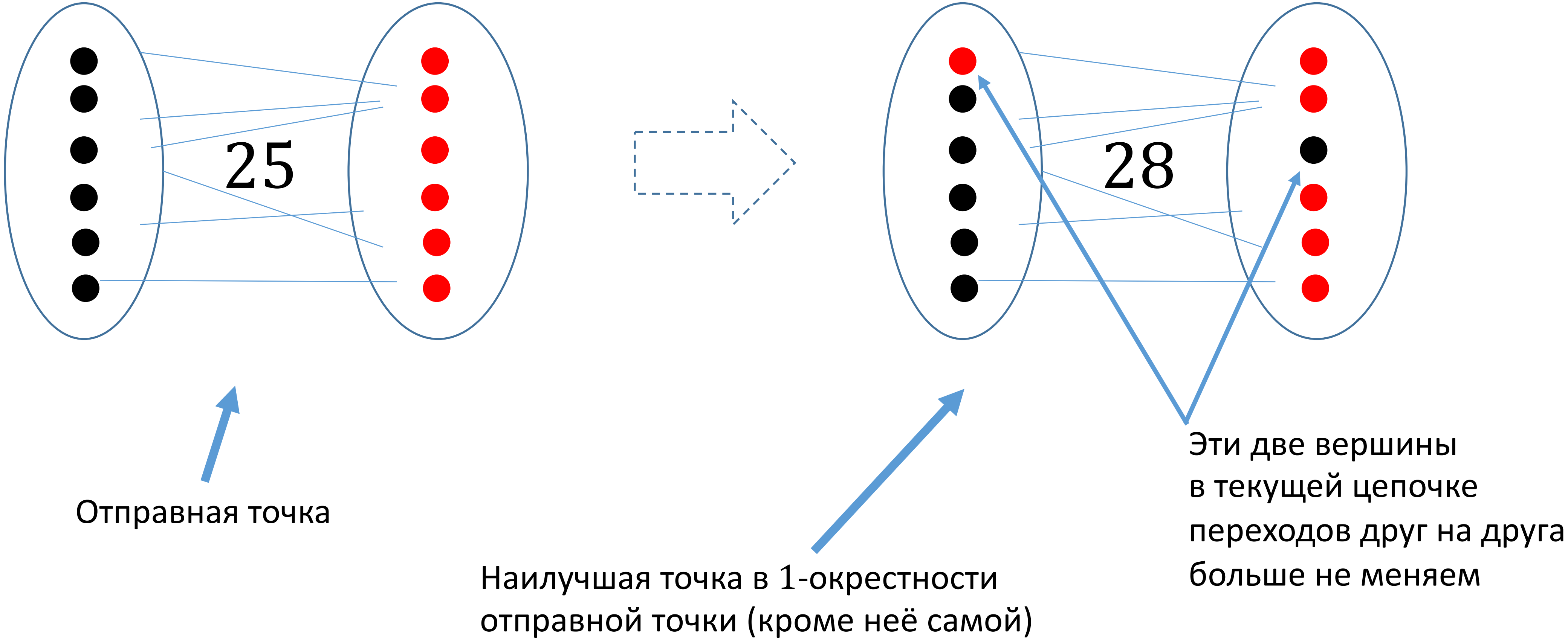
- рассматриваем на элементарном шаге только $1$-окрестность,
- *предварительно разрешаем* переход из текущей точки даже в менее хорошую (но выбираем «меньшее из зол»),
- если последовательность переходов закончилась в точке, которая лучше начальной, то *фиксируем* эту последовательность.

# Задача о разбиении графа: Керниган—Лин

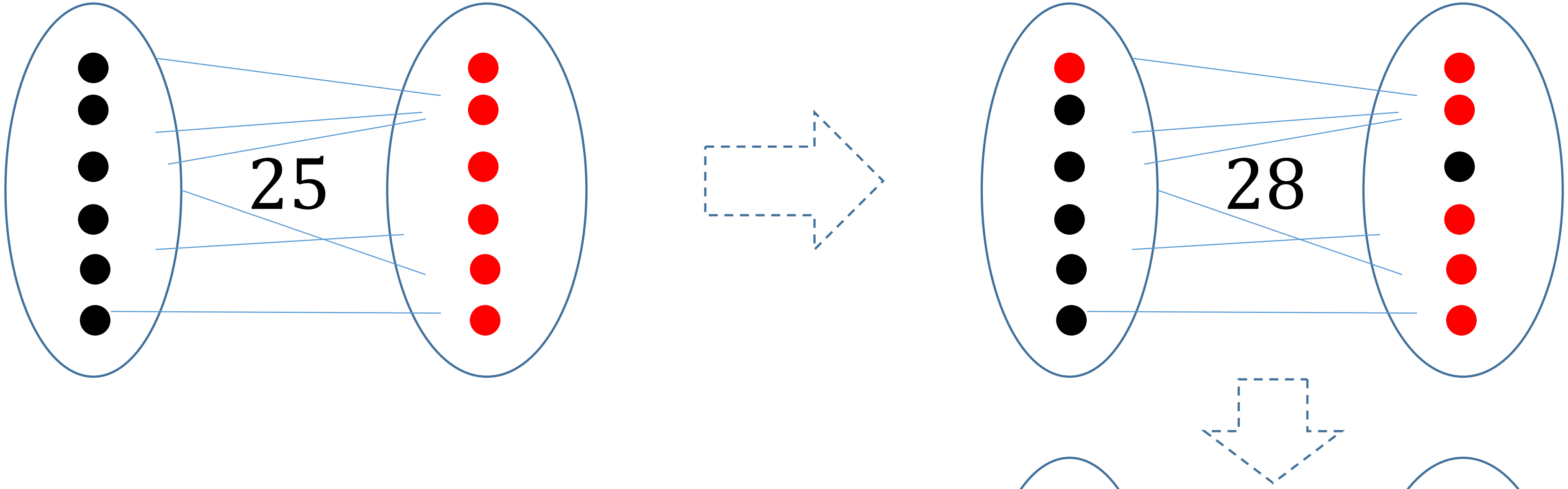



Отправная точка

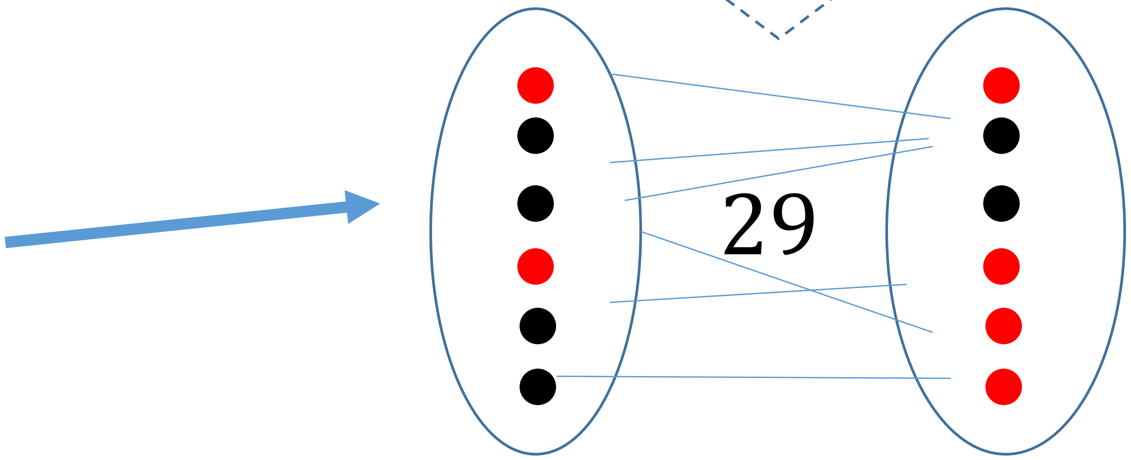
# Задача о разбиении графа: Керниган—Лин



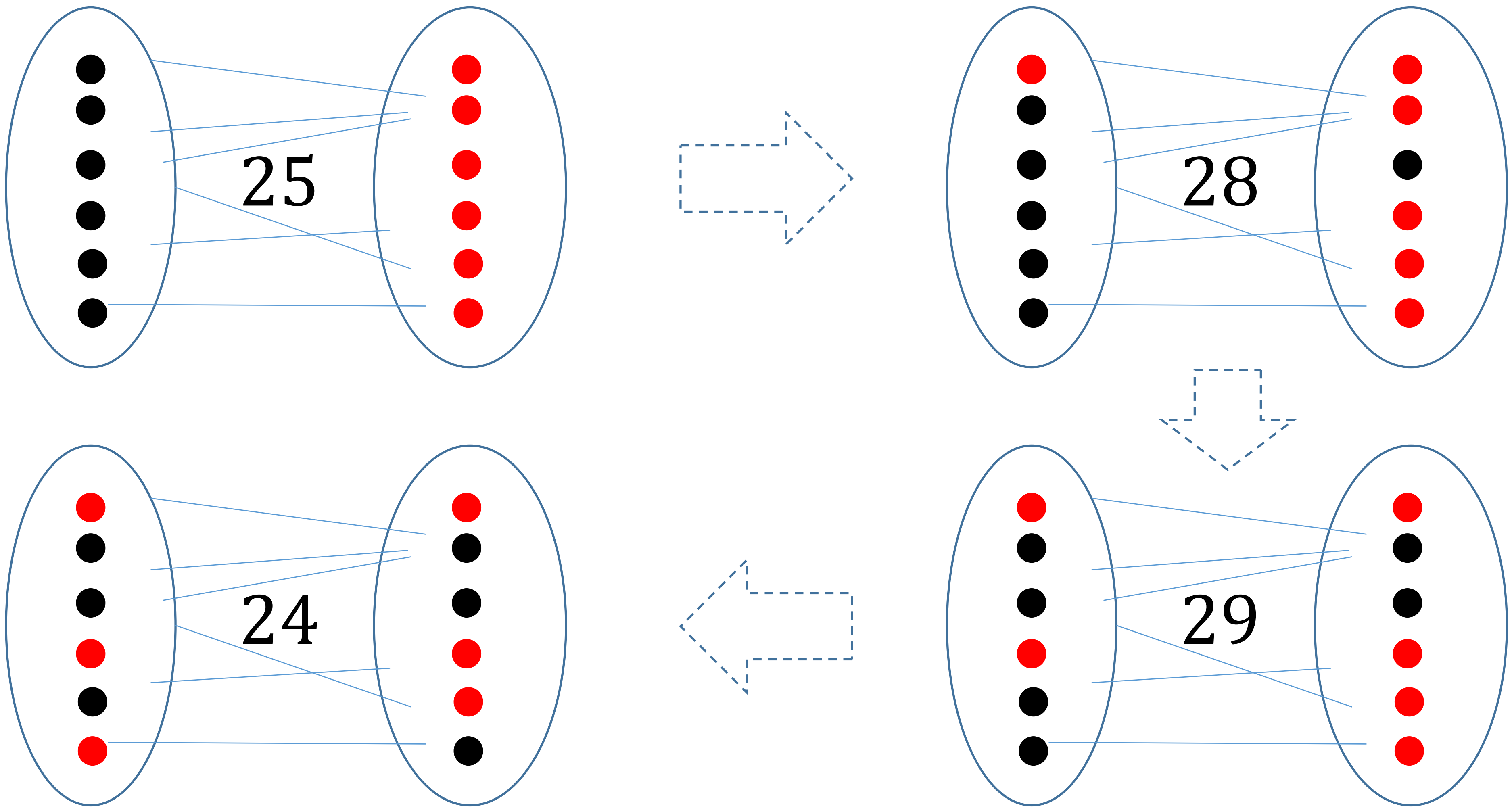
# Задача о разбиении графа: Керниган—Лин



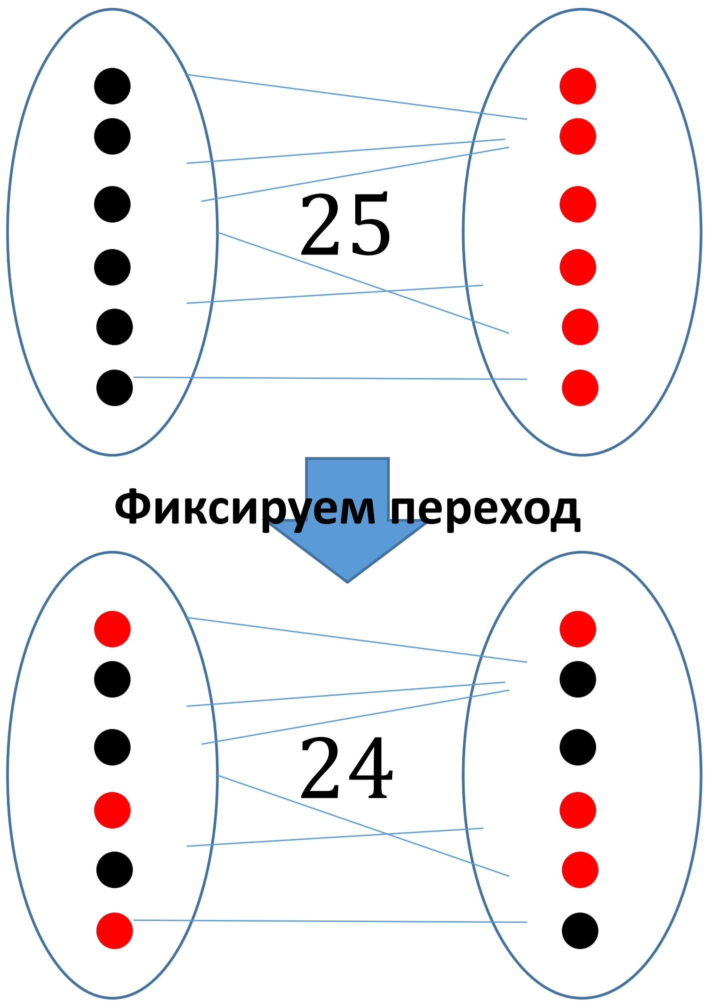

Наилучшая из точек, находящихся в 1-окрестности предыдущей точки, и отличных от всех точек в текущей цепочке переходов



# Задача о разбиении графа: Керниган—Лин



# Задача о разбиении графа: Керниган—Лин




Эта точка, очевидно, находится
в $3$-окрестности отправной точки,
но мы нашли её без просмотра всей $\mathcal{N}_3$,
три раза перейдя по $\mathcal{N}_1$

# Поиск переменной глубины: общий алгоритм

1. $x$ := random($\mathcal{S}$)
2. if $\exists y \in \mathcal{N}(x)$: $f(y) < f(x)$ then
3.     $x$ := $y$, goto 2
4. else
5.     $y_0 \coloneqq x$
6.     for $i \in \{1,2,\ldots,\mathrm{maxDepth}-1\}$ do
7.         $y_{i+1} \coloneqq \underset{y \in \mathcal{N}(y_i) \setminus \{y_0, y_1, \ldots, y_i\}}{\operatorname{argmin}} f(y)$
8.         if $f(y_{i+1}) < f(x)$ then
9.             $x$ := $y$, goto 2
10. return($x$)

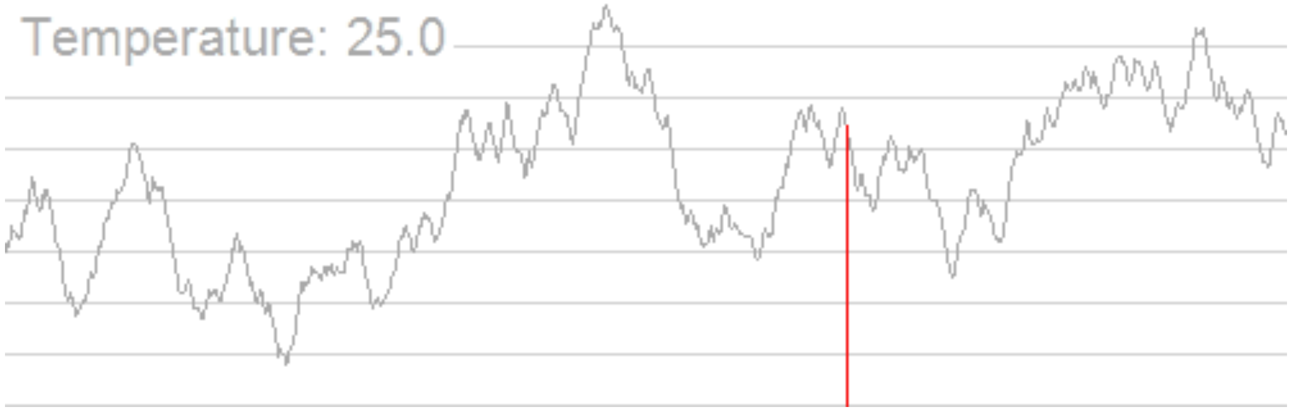
# Имитация отжига

Общая идея:

- Локальный поиск очень прост идейно, но может «застревать» в локальных экстремумах.
- Чтобы выбираться из локальных экстремумов, нужно иногда далеко от них отдаляться.
- Постепенно уменьшаем «рискованность» — дальность прыжка, на которую мы выскакиваем из локального оптимума.

Физическая идея: по мере остывания вещества перемещения атомов в кристаллической решётке всё менее амплитудны.

# Имитация отжига



# Имитация отжига: псевдокод (максимизируем функцию $f$)

```
// В коде ниже P(…) — функция в [0,1], random — генератор случайных чисел из [0,1],
// temperature — убывающая функция, neighbour — рандомизированная функция выбора «соседа».

s ← s0; fs ← f(s)                                   // Начальное состояние
sbest ← s; fbest ← f(s)
k ← 0                                               // Счётчик числа шагов
while k < kmax and fs < ftolerable                  // Пока есть время и простор для улучшения…
  T ← temperature(k/kmax)                           // вычисляем, какая сейчас «температура»,
  snew ← neighbour(s, T)                            // выбираем точку-кандидата на перемещение,
  fnew ← f(snew)                                    // вычисляем, насколько хороша новая точка.
  if fnew > fbest then                              // Если удалось улучшить результат,
    sbest ← snew; fbest ← fnew                      // то сохраняем об этом информацию.
  if P(fs, fnew, T) > random() then                 // Определяем, перемещаться ли в новую тчк.
    s ← snew; fs ← fnew                             // Перемещаемся.
  k ← k + 1
return sbest
```

# Условия остановки

В случае, когда алгоритм рандомизированный, обычно нужны дополнительные условия остановки:

- Алгоритм проделал $N$ итераций.
- Алгоритм проработал $T$ секунд.
- За последние $N$ итераций решение не было улучшено.
- За последние $N$ итераций улучшение целевой функции произошло не больше чем на $\varepsilon$.
- …

# Табу-поиск


- Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research.* Vol. 13, pp. 533-549.

- Hansen, P. 1986. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

# Табу-поиск: предпосылки

В решениях, получаемых локальным поиском, часто бывают «устойчивые признаки». Например:

- В задаче TSP во многие результирующие циклы входит одно и то же ребро.
- В задаче о разбиении графа в итоговых разбиениях некоторая пара вершин всегда оказывается «по разные стороны забора».
- …

Что это наблюдение может означать:

- устойчивый признак устойчив потому, что хорош, и нужно искать решения, которые заведомо его содержат,
- устойчивый признак *недостаточно* хорош, но назойлив и сбивает нас с пути к глобальному оптимуму.

# Табу-поиск: предпосылки

Выход из положения:

- Формировать в ходе поиска список запретов (Tabu List), которые *почти* нельзя нарушать при переходе от одной точки к другой.
- Нарушать запреты можно в случае, когда от этого получается уж очень хороший выигрыш.
- Про некоторые запреты не следует помнить *слишком* долго.

# Tabu Search Strategy

Необходимо выработать стратегию работы с табу-списком:

- Forbidding strategy: когда и чем пополнять табу-список
- Freeing strategy: когда и что удалять из табу-списка
- Short-term strategy: текущее взаимодействие между соблюдением/нарушением табу

# Параметры табу-поиска

- Процедура локального поиска
- Система окрестностей
- Условия преодоления запретов (aspiration conditions/criteria)
- Формы запретов (tabu moves)
- Добавление tabu move
- Максимальный размер табу-списка
- Правила оста

# Табу-поиск: диаграмма