

# Przeszukiwanie lokalne

1. Klasyfikacja algorytmów
2. Przeszukiwanie lokalne

# 1. Klasyfikacja algorytmów

## Algorytmy dokładne

- znajdują rozwiązanie optymalne,

# 1. Klasyfikacja algorytmów

## Algorytmy dokładne

- znajdują rozwiązanie optymalne,
- dedykowane do pewnych problemów (specyficznych),

# 1. Klasyfikacja algorytmów

## Algorytmy dokładne

- znajdują rozwiązanie optymalne,
- dedykowane do pewnych problemów (specyficznych),
- oparte o przeszukiwanie wyczerpujące,

# 1. Klasyfikacja algorytmów

## Algorytmy dokładne

- znajdują rozwiązanie optymalne,
- dedykowane do pewnych problemów (specyficznych),
- oparte o przeszukiwanie wyczerpujące,
- oparte o schemat B&B,

# 1. Klasyfikacja algorytmów

## Algorytmy dokładne

- znajdują rozwiązanie optymalne,
- dedykowane do pewnych problemów (specyficznych),
- oparte o przeszukiwanie wyczerpujące,
- oparte o schemat B&B,
- oparte o schemat PD.

Wymagają wygenerowania i sprawdzenia całej przestrzeni rozwiązań dopuszczalnych ( $\mathcal{X}$ ).

Proste w implementacji ze względu na konieczność systematycznego sposobu przeszukiwania  $\mathcal{X}$ .

Przeszukiwanie  $\mathcal{X}$  zależy od reprezentacji rozwiązań.

Wady ...?

# 1. Klasyfikacja algorytmów

## Algorytmy aproksymacyjne

- znajdują rozwiązania przybliżone i ... (dotyczy jakości rozwiązania)

# 1. Klasyfikacja algorytmów

## Algorytmy aproksymacyjne

- znajdują rozwiązania przybliżone i ... (dotyczy jakości rozwiązania)
- ... dają gwarancję znalezienia rozwiązania z określonym przybliżeniem,



# 1. Klasyfikacja algorytmów

## Algorytmy aproksymacyjne

- znajdują rozwiązania przybliżone i ... (dotyczy jakości rozwiązania)
- ... dają gwarancję znalezienia rozwiązania z określonym przybliżeniem,
- dedykowane dla określonych problemów **NP**-trudnych.

Rzadko używane do rozwiązywania problemów rzeczywistych w praktyce.

# 1. Klasyfikacja algorytmów

Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)

# 1. Klasyfikacja algorytmów

Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)
- ... gwarancji jego jakości,

# 1. Klasyfikacja algorytmów

## Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)
- ... gwarancji jego jakości,
- przeszukiwanie lokalne,

# 1. Klasyfikacja algorytmów

## Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)
- ... gwarancji jego jakości,
- przeszukiwanie lokalne,
- przeszukiwanie lokalne "polepszane",

# 1. Klasyfikacja algorytmów

## Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)
- ... gwarancji jego jakości,
- przeszukiwanie lokalne,
- przeszukiwanie lokalne "polepszane",
- algorytmy populacyjne,

# 1. Klasyfikacja algorytmów

## Algorytmy (meta)heurystyczne

- znajdują optima lokalne, ale bez ... (dotyczy jakości rozwiązania)
- ... gwarancji jego jakości,
- przeszukiwanie lokalne,
- przeszukiwanie lokalne "polepszane",
- algorytmy populacyjne,
- algorytmy hybrydowe.

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Co oznacza *lokalne*?



## 2. Przeszukiwanie lokalne

**Pytanie 1.** Co oznacza *lokalne*?

Posługujemy się pojęciem **sąsiedztwa**.

Sąsiedztwo rozwiązania  $x$ ,  $N(x) \subseteq \mathcal{X}$ , to zbiór rozwiązań leżących „blisko”  $x$ .

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Co oznacza *lokalne*?

Posługujemy się pojęciem **sąsiedztwa**.

Sąsiedztwo rozwiązania  $x$ ,  $N(x) \subseteq \mathcal{X}$ , to zbiór rozwiązań leżących „blisko”  $x$ .

**Pytanie 2.** Jak jest definiowane sąsiedztwo w problemach ciągłych i dyskretnych?

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Co oznacza *lokalne*?

Posługujemy się pojęciem **sąsiedztwa**.

Sąsiedztwo rozwiązania  $x$ ,  $N(x) \subseteq \mathcal{X}$ , to zbiór rozwiązań leżących „blisko”  $x$ .

**Pytanie 2.** Jak jest definiowane sąsiedztwo w problemach ciągłych i dyskretnych?

W problemach ciągłych  $N(x)$  definiowane jest przez funkcję odległości (metrykę).

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Co oznacza *lokalne*?

Posługujemy się pojęciem **sąsiedztwa**.

Sąsiedztwo rozwiązania  $x$ ,  $N(x) \subseteq \mathcal{X}$ , to zbiór rozwiązań leżących „blisko”  $x$ .

**Pytanie 2.** Jak jest definiowane sąsiedztwo w problemach ciągłych i dyskretnych?

W problemach ciągłych  $N(x)$  definiowane jest przez funkcję odległości (metrykę).

W problemach dyskretnych  $N(x)$  definiowane jest przez wszystkie możliwe transformacje  $x$ .

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Czym jest transformacja?

## 2. Przeszukiwanie lokalne

**Pytanie 1.** Czym jest transformacja?

Z  $x$  (będąc w  $x$ ) można wykonać jeden ruch  $m$  ze zbioru możliwych ruchów  $M(x)$ .

Ruch  $m$  jest transformacją, która zastosowana do  $x$  daje nowe rozwiązanie  $x'$ .

Zatem

$$n(x) = \left\{ x' : \exists m \in M(x) \quad x' = m(x) \right\}$$

## Cechy dobrego sąsiedztwa

Ograniczenie na rozmiar:

- dla każdego  $x$ ,  $N(x)$  zawiera co najmniej jedno rozwiązanie  $x'$  różne od  $x$ .
- $N(x)$  nie może obejmować całej przestrzeni  $\mathcal{X}$  - nie może być dokładne.

## Cechy dobrego sąsiedztwa

Ograniczenie na rozmiar:

- dla każdego  $x$ ,  $N(x)$  zawiera co najmniej jedno rozwiązanie  $x'$  różne od  $x$ .
- $N(x)$  nie może obejmować całej przestrzeni  $\mathcal{X}$  - nie może być dokładne.

Podobieństwo sąsiadów:

- $x' \in N(x)$  niewiele różni się od  $x$  - ruch elementarny  $m$  z  $x$  do  $x'$  nie może powodować konieczności konstruowania nowego rozwiązania.



## Cechy dobrego sąsiedztwa

Ograniczenie na rozmiar:

- dla każdego  $x$ ,  $N(x)$  zawiera co najmniej jedno rozwiązanie  $x'$  różne od  $x$ .
- $N(x)$  nie może obejmować całej przestrzeni  $\mathcal{X}$  - nie może być dokładne.

Podobieństwo sąsiadów:

- $x' \in N(x)$  niewiele różni się od  $x$  - ruch elementarny  $m$  z  $x$  do  $x'$  nie może powodować konieczności konstruowania nowego rozwiązania.

Równouprawnienie

- niezależnie od początkowego wyboru ( $x^0$ ) osiągalne powinno być każde rozwiązanie w  $\mathcal{X}$ .

# Sąsiedztwo

Pojęcie otoczenia bądź sąsiedztwa punktu w przestrzeni definiuje się w analizie matematycznej i w optymalizacji ciągłej. W przestrzeni  $\mathbb{R}^n$  otoczeniem punktu  $x$  jest hipersfera o promieniu  $\epsilon$  o środku w punkcie  $x$ .

W problemach z dyskretną przestrzenią rozwiązań (np. permutacji  $n$ -elementowych) nie jest to już takie proste.

**Pytanie 1.** Jak zatem definiować sąsiedztwo (otoczenie) w takiej przestrzeni?

# Sąsiedztwo

Pojęcie otoczenia bądź sąsiedztwa punktu w przestrzeni definiuje się w analizie matematycznej i w optymalizacji ciągłej. W przestrzeni  $\mathbb{R}^n$  otoczeniem punktu  $x$  jest hipersfera o promieniu  $\epsilon$  o środku w punkcie  $x$ .

W problemach z dyskretną przestrzenią rozwiązań (np. permutacji  $n$ -elementowych) nie jest to już takie proste.

**Pytanie 1.** Jak zatem definiować sąsiedztwo (otoczenie) w takiej przestrzeni?

Różnie i na wiele sposobów. W literaturze definiowane jest ono na trzy sposoby:

- sąsiedztwo typu *insert*,
- sąsiedztwo typu *swap*,
- sąsiedztwo typu *invert*.

Sąsiedztwem typu *insert* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_I(\pi)$ , gdzie każda permutacja  $\pi' \in N_I(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *insert*.

Ruch typu *insert* zdefiniowany jest przez parę liczb  $(i, j)$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, n$ ,  $i \neq j$  i powoduje usunięcie elementu z pozycji  $i$ -tej i wstawieniu go na pozycję  $j$ -tą.

**Pytanie 2.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

Sąsiedztwem typu *insert* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_I(\pi)$ , gdzie każda permutacja  $\pi' \in N_I(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *insert*.

Ruch typu *insert* zdefiniowany jest przez parę liczb  $(i, j)$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, n$ ,  $i \neq j$  i powoduje usunięcie elementu z pozycji  $i$ -tej i wstawieniu go na pozycję  $j$ -tą.

**Pytanie 2.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

$$\pi' = \langle \pi(1), \dots, \pi(i-1), \pi(j), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j+1), \dots, \pi(n) \rangle$$

**Pytanie 3.** Ile jest/może być permutacji w sąsiedztwie typu *insert*?

**Pytanie 3.** Ile jest/może być permutacji w sąsiedztwie typu *insert*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n - 1)$ .
2. Ruch  $(i, i + 1)$  oraz  $(i + 1, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje o  $(n - 1)$ .

W związku z tym, liczba permutacji w sąsiedztwie typu *insert* wynosi ...

**Pytanie 3.** Ile jest/może być permutacji w sąsiedztwie typu *insert*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n-1)$ .
2. Ruch  $(i, i+1)$  oraz  $(i+1, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje o  $(n-1)$ .

W związku z tym, liczba permutacji w sąsiedztwie typu *insert* wynosi

$$n(n-1) - (n-1) = (n-1)^2 = O(n^2)$$

czyli

$$|N_I(n)| = (n-1)^2 = O(n^2)$$



Sąsiedztwem typu *swap* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_S(\pi)$ , gdzie każda permutacja  $\pi' \in N_S(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *swap*.

Ruch typu *swap* zdefiniowany jest przez parę liczb  $(i, j)$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, n$ ,  $i \neq j$  i powoduje zamianę elementu z pozycji  $i$ -tej z elementem z pozycji  $j$ -tej.

**Pytanie 4.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

Sąsiedztwem typu *swap* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_S(\pi)$ , gdzie każda permutacja  $\pi' \in N_S(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *swap*.

Ruch typu *swap* zdefiniowany jest przez parę liczb  $(i, j)$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, n$ ,  $i \neq j$  i powoduje zamianę elementu z pozycji  $i$ -tej z elementem z pozycji  $j$ -tej.

**Pytanie 4.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(j), \pi(i+1), \dots, \pi(j-1), \pi(i), \pi(j+1), \dots, \pi(n) \rangle$$

**Pytanie 5.** Ile jest/może być permutacji w sąsiedztwie typu *swap*?

**Pytanie 5.** Ile jest/może być permutacji w sąsiedztwie typu *swap*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n - 1)$ .
2. Ruch  $(i, j)$  oraz  $(j, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje dwukrotnie.

W związku z tym, liczba permutacji w sąsiedztwie typu *swap* wynosi ...

**Pytanie 5.** Ile jest/może być permutacji w sąsiedztwie typu *swap*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n-1)$ .
2. Ruch  $(i, j)$  oraz  $(j, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje dwukrotnie.

W związku z tym, liczba permutacji w sąsiedztwie typu *swap* wynosi

$$n(n-1)/2 = O(n^2)$$

czyli

$$|N_S(n)| = n(n-1)/2 = O(n^2)$$

Sąsiedztwem typu *invert* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_V(\pi)$ , gdzie każda permutacja  $\pi' \in N_S(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *invert*.

Ruch typu *invert* zdefiniowany jest przez parę liczb  $(i, j), i = 1, \dots, n, j = 1, \dots, n, i \neq j$  i powoduje odwrócenie w permutacji ciągu od elementu z pozycji  $i$ -tej do elementu z pozycji  $j$ -tej.

**Pytanie 6.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

Sąsiedztwem typu *invert* permutacji  $\pi = \langle \pi(1), \dots, \pi(n) \rangle$  jest zbiór permutacji  $N_V(\pi)$ , gdzie każda permutacja  $\pi' \in N_S(\pi)$  powstaje z permutacji  $\pi$  przez wykonanie ruchu  $m$  typu *invert*.

Ruch typu *invert* zdefiniowany jest przez parę liczb  $(i, j), i = 1, \dots, n, j = 1, \dots, n, i \neq j$  i powoduje odwrócenie w permutacji ciągu od elementu z pozycji  $i$ -tej do elementu z pozycji  $j$ -tej.

**Pytanie 6.** Jak zatem wyglądać będzie permutacja  $\pi'$  po wykonaniu ruchu  $(i, j)$ , jeśli permutacja  $\pi$  ma postać:

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(i), \pi(i+1), \dots, \pi(j-1), \pi(j), \pi(j+1), \dots, \pi(n) \rangle$$

$$\pi = \langle \pi(1), \dots, \pi(i-1), \pi(j), \pi(j-1), \dots, \pi(i+1), \pi(i), \pi(j+1), \dots, \pi(n) \rangle$$

**Pytanie 7.** Ile jest/może być permutacji w sąsiedztwie typu *invert*?



**Pytanie 7.** Ile jest/może być permutacji w sąsiedztwie typu *invert*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n - 1)$ .
2. Ruch  $(i, j)$  oraz  $(j, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje dwukrotnie.

W związku z tym, liczba permutacji w sąsiedztwie typu *invert* wynosi ...

**Pytanie 7.** Ile jest/może być permutacji w sąsiedztwie typu *invert*?

1. Ponieważ dla ruchu  $(i, j)$  wartości  $i$  i  $j$  mogą się zmieniać od 1 do  $n$  i  $i \neq j$ , ruchów  $m$  można wykonać  $n(n - 1)$ .
2. Ruch  $(i, j)$  oraz  $(j, i)$  powodują uzyskanie tej samej permutacji. Stąd liczba ruchów dających różne permutacje zmaleje dwukrotnie.

W związku z tym, liczba permutacji w sąsiedztwie typu *invert* wynosi

$$n(n - 1)/2 = O(n^2)$$

czyli

$$|N_V(n)| = n(n - 1)/2 = O(n^2)$$

## Przykład sąsiedztwa dla TSP

$k$ -zamiana:  $N(x)$  to zbiór rozwiązań powstałych przez usunięcie  $k$  miast i wstawienie ich w inne miejsca - inna kolejność miast, nowa permutacja.

2-zamiana miast 3 i 8

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

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

## Przykład sąsiedztwa dla TSP

Wymiana łuków:  $N(x)$  to zbiór rozwiązań powstałych przez usunięcie  $k$  kolejnych miast i wstawienie ich w odwrotnej kolejności

Wymiana łuków 3 i 8

$$\begin{array}{l} 1 - 2 - \left[ 3 - 4 - 5 - 6 - 7 - 8 \right] - 9 \\ 1 - 2 - \left[ 8 - 7 - 6 - 5 - 4 - 3 \right] - 9 \end{array}$$

## Algorytm przeszukiwania lokalnego

Wybierz rozwiązanie startowe  $x^0$  w  $\mathcal{X}$  (skonstruuj lub losowo)

Powtarzaj dla  $k = 1, 2, \dots$

Wygeneruj nowe rozwiązanie  $x' = N(x^{k-1})$

Jeśli  $x'$  jest lepsze od  $x^{k-1}$ , czyli  $f(x') < f(x^{k-1})$ ,

wybierz  $x'$  jako rozwiązanie bieżące,

$x^k = x'$ ,

w przeciwnym wypadku odrzuć  $x'$  i przyjmij  $x^k = x^{k-1}$ .

...dopóki nie można poprawić rozwiązania bieżącego  $x^{k-1}$  (minimum lokalne).

**Pytanie.** Kiedy

$$x \stackrel{\text{def}}{=} x_{min}$$

czyli  $x_{min}$  jest minimum lokalnym?

**Pytanie.** Kiedy

$$x \stackrel{\text{def}}{=} x_{min}$$

czyli  $x_{min}$  jest minimum lokalnym?

**Definicja**

$x_{min}$  jest **lokalnym minimum**, jeśli:

$$f(x_{min}) \leq f(x)$$

dla wszystkich  $x \in N(x_{min})$

wersja zachłanna *greedy*

wersja stroma *steepest*

wybierz rozwiązanie startowe  $x^0$

powtarzaj  $k = 1, 2, \dots$

1) przeglądaj  $N(x^{k-1})$  w losowej kolejności  
dopóki nie znajdziesz rozwiązania  
 $x' \in N(x^{k-1})$  takiego, że

$$f(x') < f(x^{k-1}).$$

2) jeśli znaleziono  $x'$ , to  $x^k = x'$

1) wybierz  $x'$  jako najlepsze rozwiązanie  
w  $N(x^{k-1})$  tzn. wybierz takie  
 $x'$ , że  $f(x') \leq f(x)$  dla wszystkich

$$x \in N(x^{k-1}).$$

2) jeśli  $f(x') < f(x^{k-1})$  to przyjmij  $x^k = x'$

dopóki nie można poprawić rozwiązania bieżącego  $x^{k-1}$ .



**Pytanie.** Która wersja jest „lepszą”?

**Pytanie.** Która wersja jest „lepsza”?

- Zależy od problemu i sąsiedztwa.
- Wersja stroma osiągnie minimum lokalne szybciej - przy mniejszej liczbie iteracji. **dłaczego?**
- Zachłanna nie przegląda całego sąsiedztwa.
- Każda osiągnie najbliższe (pierwsze) minimum lokalne.

Plusy dodatnie ☺...jak wszystkie metaheurystyki

- do zastosowania w dowolnym problemie kombinatorycznym,
- wystarczy (czy to łatwe?) dostosować definicję sąsiedztwa.

Plusy ujemne ☹...jak wszystkie metaheurystyki

- znajdują (dobrze, że zawsze) minimum lokalne (a kiedy na pewno globalne?),
- nie dają gwarancji jakości rozwiązania - zależna od wyboru  $x^0$ .

**Pytanie.** Jak zmodyfikować  $LS$ , aby uniknąć ww. wad?

1. ...

2. ...

3. ...

4. ...

5. ...

**Pytanie.** Jak zmodyfikować  $LS$ , aby uniknąć ww. wad?

1. Bardziej złożona definicja sąsiedztwa - większy zakres przeszukiwania  $\mathcal{X}$  kosztowne obliczeniowo.
2. Wybrać „dobre”  $x^0$  - tylko jak je skonstruować?
3. Wykonać algorytm wielokrotnie - kosztowne obliczeniowo na jednej maszynie; nie, jeśli zrównoleglimy proces.
4. Akceptowanie (ograniczone) gorszych wartości funkcji celu  $\longrightarrow \dots$ ?
5. Zapamiętywanie i unikanie (chwilowe) sprawdzonych rozwiązań w  $N(x^k)$   $\longrightarrow \dots$ ?

**Pytanie.** Jak zmodyfikować  $LS$ , aby uniknąć ww. wad?

1. Bardziej złożona definicja sąsiedztwa - większy zakres przeszukiwania  $\mathcal{X}$  kosztowne obliczeniowo.
2. Wybrać „dobre”  $x^0$  - tylko jak je skonstruować?
3. Wykonać algorytm wielokrotnie - kosztowne obliczeniowo na jednej maszynie; nie, jeśli zrównoleglić proces.
4. Akceptowanie (ograniczone) gorszych wartości funkcji celu  $\rightarrow$  (SA).
5. Zapamiętywanie i unikanie (chwilowe) sprawdzonych rozwiązań w  $N(x^k) \rightarrow$  (TS).

ad.3.

- Lokalne przeszukiwanie z różnych punktów (*ang. multiply start local search*).
- Lokalne przeszukiwanie ze zmiennym sąsiedztwem (*ang. variable neighborhood locale search*).
- Iteracyjne przeszukiwanie lokalne (*ang. iterated local search*).

# Symulowane wyżarzanie

Metoda wzorowana na zjawiskach zachodzących w procesie ochładzania (zestalania) cieczy i stygnięcia metali. Przy stopniowym, powolnym ochładzaniu cząsteczki ciała oddając energię rozkładają się sposób systematyczny tworząc równomierne struktury.

## Krótką historia algorytmu

1. Nicolas Metropolis. The Beginning of the Monte Carlo Method. Los Alamos Science, No. 15, Page 125
2. N. Metropolis; A.W. Rosenbluth; M.N. Rosenbluth; A.H. Teller and E. Teller (1953). "Equation of State Calculations by Fast Computing Machines". Journal of Chemical Physics. 21 (6): 1087–1092
3. Khachaturyan, A.; Semenovskaya, S.; Vainshtein, B. (1979). "Statistical-Thermodynamic Approach to Determination of Structure Amplitude Phases". Sov.Phys. Crystallography. 24 (5): 519–524
4. Khachaturyan, A.; Semenovskaya, S.; Vainshtein, B. (1981). "The Thermodynamic Approach to the Structure Analysis of Crystals". Acta Crystallographica (A37): 742–754
5. Granville, V.; Krivanek, M.; Rasson, J.-P. (1994). "Simulated annealing: A proof of convergence". IEEE Transactions on Pattern Analysis and Machine Intelligence. 16 (6): 652–656

## Iteracyjne wspinanie się

1. inicjuj  $t$  (liczba iteracji),  $best$
2. (pętla zewnętrzna) wybierz losowo rozwiązanie  $x$ , oceń  $x$
3. (pętla wewnętrzna) wybierz wszystkie rozwiązania z otoczenia, wybierz rozwiązanie  $y$ , oceń  $y$  i zamień jeśli jest lepsze niż  $x$ ; else wyjdź i wykonaj kolejną iterację

## Wady

1. utoyka w lokalnym ekstremum
2. wyszukanie lepszego rozwiązania możliwe tylko po kolejnym uruchomieniu w innym punkcie początkowym

## Słabo



## Stochastyczne wspinanie się

1. inicjuj  $t$ , wybierz losowo  $x$ , oceń  $x$
2. (pętla) wybierz z otoczenia rozwiązanie  $y$ , oceń  $y$ , wybierz  $x$  lub  $y$

Jak wybierane jest rozwiązanie?

## Stochastyczne wspinanie się

1. inicjuj  $t$ , wybierz losowo  $x$ , oceń  $x$
2. (pętla) wybierz z otoczenia rozwiązanie  $y$ , oceń  $y$ , wybierz  $x$  lub  $y$

Wybór  $y$  następuje z prawdopodobieństwem

$$p = \frac{1}{1 + e^{\frac{g(x) - g(y)}{T}}}$$

Jaka jest rola parametru  $T$ ?

Algorytm symulowanego wyżarzania

$t \leftarrow 0$

inicjuj  $T$

wybierz losowo rozwiązanie  $x$

ocień  $x$  - wylicz  $g(x)$

repeat

repeat

wybierz nowe rozwiązanie  $y$  w otoczeniu  $x$

jeżeli  $g(y) < g(x)$  wtedy  $x \leftarrow y$

w przeciwnym przypadku, jeżeli  $p[0,1) < e^{\frac{g(y)-g(x)}{T}}$ , wtedy  $x \leftarrow y$

until warunek zakończenia

$T \leftarrow f(T, t)$

$t \leftarrow t + 1$

until (kryterium zakończenia)

Najważniejsze pytania związane z rozwiązaniem problemu

1. jakie jest rozwiązanie?
2. jakie jest sąsiedztwo rozwiązania?
3. jaki jest koszt rozwiązania?
4. jak ustalić rozwiązanie początkowe?

Najważniejsze pytania związane z samym algorytmem

1. jak ustalić  $T$  (temperature) początkową?
2. jak ustalić współczynnik schładzania  $f(T, t)$ ?
3. jak ustalić warunek zakończenia iteracji?
4. jak ustalić kryterium zatrzymania algorytmu?