Metody optymalizacji dyskretnej

Metody przybliżone

Metody optymalizacji dyskretnej

Większość problemów optymalizacji dyskretnej pochodzących z praktyki (szeregowanie, harmonogramowanie, transport, plany zajęć, itp.) jest NP-trudnych. ☹

Implikuje to wykładniczą złożoność obliczeniową algorytmu rozwiązywania. ☹

Moc obliczeniowa procesorów wzrasta liniowo. ☺

Koszt obliczeń wzrasta wykładniczo z rozmiarem problemu. ☹

Wniosek: nie ma nadziei na rozwiązywanie optymalne rzeczywistych przykładów problemów w akceptowalnym w praktyce czasie. ☹

☹ - 4, ☺ -1

Metody optymalizacji dyskretnej

Metoda dokładna wyznacza rozwiązanie globalnie optymalne tzn. $x^n \in X$ takie, że

$$K^* \stackrel{\text{def}}{=} K(x^n) = \min K(x)$$

Metody optymalizacji dyskretnej

Do grupy metod dokładnych, w zależności od przynależności problemu do klasy złożoności obliczeniowej, należą:

1. efektywne algorytmy dedykowane – uważane za tanie obliczeniowo metody specjalizowane dla problemów należących do P-klasy lub NP-trudnych problemów liczbowych
2. …

Metody optymalizacji dyskretnej

2. metody oparte o schemat podziału i ograniczeń (B&B),
3. metody oparte o schemat programowania dynamicznego (PD),
4. metody oparte na programowaniu liniowym całkowitoliczbowym (PLC),
5. metody oparte na programowaniu liniowym binarnym (PLB),
6. metody subgradientowe.

… są kosztownymi obliczeniowo metodami polecanymi dla rozwiązywania problemów silnie NP-trudnych.

Jakie jednak uzyskiwać przydatne rozwiązania?

Metody przybliżone

Metoda przybliżona A wyznacza pewne rozwiązanie $x^A \in X$ takie, że

$$K(x^A) jest\ bliskie\ K(x^*)$$

Jak powinno być bliskie $K(x^A)$ do $K(x^*)$, aby było przydatne?

Błąd przybliżenia (1)

Zbiór danych liczbowych problemu specyfikuje *przykład konkretny Z* tego problemu – instancję problemu.

Oznaczmy przez $X(Z)$ zbiór wszystkich rozwiązań problemu dla tego przykładu zaś przez $K(x; Z)$ wartość kryterium K dla rozwiązania x w przykładzie Z.

Rozwiązanie $x^* \in X(Z)$ takie, że $K(x^*; Z) = min_{x \in X(Z)} K(x; Z)$ jest nazywane rozwiązaniem optymalnym dla tego przykładu.

Niech $x^A \in X(Z)$ oznacza rozwiązanie przybliżone generowane przez algorytm A dla przykładu Z.

Za błąd przybliżenia możemy przyjąć… co?

Błąd przybliżenia (2)

$$B^A(Z) = |K(x^A; Z) - K(x^*; Z)| \qquad S^A(Z) = \frac{K(x^A; Z)}{K(x^*; Z)}$$

$$T^A(Z) = \frac{K(x^A; Z) - K(x^*; Z)}{K(x^*; Z)} \qquad U^A(Z) = \frac{K(x^A; Z) - K(x^*; Z)}{K(x^A; Z)}$$

Definicja błędu może być dowolna, ale…

(a) musi być sensowna ($K(x^*) = 0$),

(b) musi być adekwatna do własności algorytmu,

(c) powinna pozwalać wykonać analizę błędu dla różnych Z.

Błąd przybliżenia (3)

Jak określić, czy dany algorytm przybliżony jest dobry (czy daje dobre rozwiązanie)?

Należy wykonać analizę błędu przybliżenia.

Jak?

Najlepiej tak, aby uzyskać możliwie dużo informacji ☺

Błąd przybliżenia (4)

Analiza zachowania się błędu przybliżenia

- Eksperymentalna – łatwa, subiektywna, zależna od wyboru przykładów
- Analiza najgorszego przypadku
- Analiza probabilistyczna – (obie) – trudniejsze niż AE, niezależne od przykładów.

W sumie AE, ANP i AP wzajemnie się uzupełniają i w połączeniu z analizą złożoności obliczeniowej stanowią kompletną charakterystykę algorytmu.

Analiza eksperymentalna (1)

- Wykonywana *a posteriori (po fakcie, z następstwa)* na reprezentatywnej próbce przykładów (czyli jakiej?)
- NP-trudność problemu implikuje kłopoty z liczeniem rzeczywistych wartości błędów ze względu na zbyt duży koszt obliczeń $K(x^*)$.
- W związku z tym redefiniujemy pojęcie błędu używając wartości referencyjnej K^{Ref} w miejsce $K(x^*)$.
- Jako K^{Ref} można przyjąć dolne ograniczenie optymalnej wartości funkcji celu lub wartość $K(x^{Ref})$, gdzie x^{Ref} jest najlepszym znanym rozwiązaniem, rozwiązaniem przybliżonym lub rozwiązaniem losowym.

1 (p51)

Analiza eksperymentalna (2)

Analiza eksperymentalna polega na wygenerowaniu pewnego zbioru S instancji badanego problemu π oraz rozwiązania wszystkich tych instancji badanym algorytmem A.

Ważne jest, aby zbiór S był zbiorem reprezentatywnym (czyli jakim?) zbioru $X(Z)$.

Najlepiej, aby $S = X(Z)$, jednak $|X(Z)| = \aleph_0$, zatem nie ma na to szans.

Jak wyznaczyć odległość $K(x^A)$ od $K(x^*)$.

Jaka miarą można się posłużyć?

Analiza eksperymentalna (3)

Dla $K(x^A; Z)$ i $K(x^*; Z)$ miarą odległości rozwiązania dostarczanego przez algorytm A od wartości optymalnej jest

$$\eta(Z) = \frac{K(x^A; Z) - K(x^*; Z)}{K(x^*; Z)}$$

dla problemów minimalizacji oraz

$$\eta(Z) = \frac{K(x^*; Z) - K(x^A; Z)}{K(x^*; Z)}$$

dla problemów maksymalizacji.

Analiza eksperymentalna (4)

Redefiniujemy pojęcie błędu używając wartości referencyjnej K^{Ref} w miejsce $K(x^*)$.

Jako K^{Ref} przyjmujemy np. dolne ograniczenie optymalnej wartości funkcji celu.

Ponieważ:

$LB \leq K(x^*; Z)$ – problem minimalizacji

$UB \geq K(x^*; Z)$ – problem maksymalizacji

zatem…

Analiza eksperymentalna (5)

$$\eta(Z) = \frac{K(x^A; Z) - LB}{LB}$$

dla problemów minimalizacji oraz

$$\eta(Z) = \frac{UB - K(x^A; Z)}{UB}$$

dla problemów maksymalizacji.

Analiza eksperymentalna (6)

Mając wyznaczone wartości η dla każdego Z możemy ocenić jakość algorytmu określając:

$$\bar{\eta} = \frac{1}{|S|}\sum_{Z \in S} \eta(Z)$$

średnią odległość od optimum,

$$\eta^{MAX} = \max\big(\eta(Z)\big), Z \in S$$

maksymalną odległość od optimum.

Analiza eksperymentalna (7)

Im $\bar{\eta}$ i η^{MAX} są bliższe zeru, tym lepszy jest algorytm.

Jeżeli $\bar{\eta} = \varepsilon$ to oznacza, że algorytm daje średnio błąd względny o wartości ε.

Należy jednak pamiętać, że S nie oznacza całego zbioru instancji. Nie można zatem powiedzieć, że η^{MAX} oznacza wartości maksymalną błędu !!!

Analiza eksperymentalna (8)

Dla problemu plecakowego algorytm A dał następujące wyniki:

Z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
K	30	34	33	48	24	41	27	35	36	29	19	21	35	36	34	42
UB	33	36	38	49	24	42	32	36	36	36	22	28	36	41	37	42
eta	0,09	0,06	0,13	0,02	0,00	0,02	0,16	0,03	0,00	0,19	0,14	0,25	0,03	0,12	0,08	0,00
eta	9,09%	5,56%	13,16%	2,04%	0,00%	2,38%	15,63%	2,78%	0,00%	19,44%	13,64%	25,00%	2,78%	12,20%	8,11%	0,00%
	/eta	0,08	8,24%													
	etaMAX	0,25	25,00%													

Co możemy powiedzieć o takim algorytmie?

Analiza najgorszego przypadku (1)

Ocena aprioryczna zachowania się błędu na *całej* populacji przykładów konkretnych Z (instancji).

Najczęściej stosowana do błędu

$$S^A(Z) = \frac{K(x^A; Z)}{K(x^*; Z)}$$

W stosunku do powyższego definiuje się:

współczynnik najgorszego przypadku

$$\eta^A = min\{y: K(x^A; Z)/K(x^*; Z) \leq y, \forall Z\}$$

asymptotyczny współczynnik najgorszego przypadku

$$\eta_{\infty}^{A} = min\{y: K(x^A; Z)/K(x^*; Z) \leq y, \forall Z \in \{W: K(x^*; W) \geq L\}\}$$

Analiza najgorszego przypadku (2)

Analiza najgorszego przypadku dostarcza ocen skrajnie pesymistycznych.

Wystąpienie ekstremalnej wartości błędu, choć możliwe teoretycznie, może w praktyce się nie wydarzyć (lub zdarzyć bardzo rzadko), ze względu na wysoką specyficzność danych instancji.

Stąd, oceny pesymistyczne mogą odbiegać znacznie od ocen eksperymentalnych i przeciętnych.

Gwarancję ograniczenia wartości błędu posiadają również *schematy aproksymacyjne*.

1 (p61)

Analiza probabilistyczna (1)

Podobnie jak ANP jest to analiza aprioryczna.

Zakłada ona, że każdy przykład konkretny *Z* został otrzymany jako realizacja pewnej liczby *n* niezależnych zmiennych losowych o znanym (zwykle równomiernym) rozkładzie prawdopodobieństwa.

Dla podkreślenia tego założenia używamy zapisu Z_n zamiast Z.

Zarówno $K(x^A; Z)$ i $K(x^*; Z)$są także zmiennymi losowymi.

Przez $M^A(Z_n)$ (również zmienna losowa) oznaczymy wartość błędu algorytmu A dla przykładu Z_n.

np. $M^A(Z_n) = K(x^A; Z_n) - K(x^*; Z_n)$

Analiza probabilistyczna (2)

Analiza probabilistyczna dostarcza podstawowych informacji o zachowaniu się zmiennej losowej $M^A(Z_n)$, np. jej rozkładzie prawdopodobieństwa, momentach: średniej $E(M^A(Z_n))$ i wariancji $Var(M^A(Z_n))$, etc.

Jeżeli $E(M^A(Z_n)) = 0$ to oznacza, że w przypadku średnim algorytm A dostarcza rozwiązań optymalnych.

Jednakże najbardziej interesujące charakterystyki dotyczą typu zbieżności $M^A(Z_n)$ do wartości stałej m wraz ze wzrostem n oraz szybkości tej zbieżności.

Analiza probabilistyczna (3)

Wyróżniamy trzy typy zbieżności:

(1) prawie na pewno (najlepszy z całej trójki)

$$P\{\lim_{n\to\infty} M^A(Z_n) = m\} = 1$$

(2) według prawdopodobieństwa

$$\lim_{n\to\infty} P\{|M^A(Z_n) - m| > \varepsilon\} = 0, \forall \varepsilon > 0$$

(3) według średniej

$$\lim_{n\to\infty} |E(M^A(Z_n)) - m| = 0$$

Analiza probabilistyczna (4)

Występują następujące wynikania dla powyższych typów zbieżności:

$$(1) \Rightarrow (2) \text{ i } (3) \Rightarrow (2)$$

ale

$$(2) \Rightarrow (1) \text{ i } (2) \Rightarrow (3)$$

zachodzą tylko wtedy, gdy

$$\sum_{n=1}^{\infty} P\{\,|M^A(Z_n) - m| > \varepsilon\} < \infty$$

Analiza probabilistyczna (5)

Analiza probabilistyczna dostarcza ocen uśrednionych po *całej* populacji instancji.

Stąd, wyniki obserwowane w eksperymentach mogą odbiegać od ocen teoretycznych.

W praktyce, większość algorytmów wykazuje w analizie eksperymentalnej zbieżność błędów względnych do zera przy wzroście rozmiaru problemu.

Ponieważ analiza probabilistyczna dostarcza znaczących wyników zachowania się algorytmu, jest zwykle dość skomplikowana. Z tego powodu do tej pory niewiele algorytmów doczekało się starannego opracowania tego tematu.

Schematy aproksymacyjne (1)

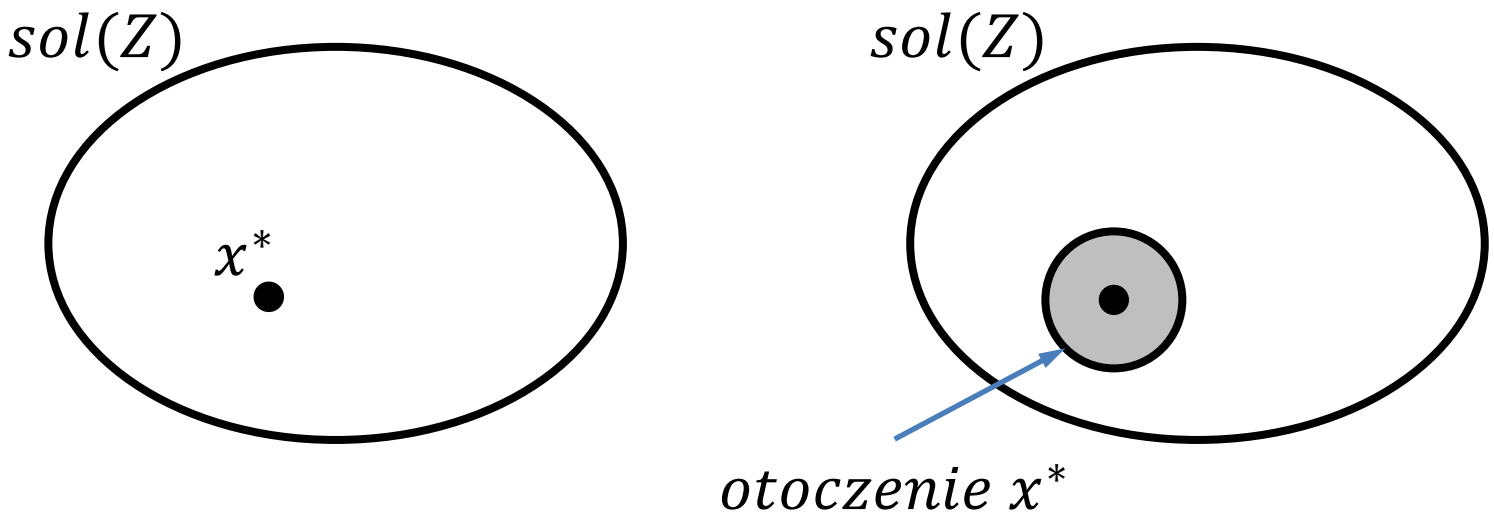
Jeżeli mamy do czynienia z problemem optymalizacyjnym $\mathcal{NP}$–trudnym, to znalezienie rozwiązania optymalnego może być czasochłonne.

Idea aproksymacji polega na osłabieniu wymagań dotyczących rozwiązania.

W konsekwencji oznacza to, że szukamy rozwiązania *bliskiego* optymalnemu.

Jak bliskiego? Czy gorszego? Jeśli tak, to o ile gorszego?

Schematy aproksymacyjne (2)



np. $F(x) \leq 2F(x^*)$

musi jednak być akceptowalne

Schematy aproksymacyjne (3)

Wielomianowy algorytm A nazywamy *algorytmem k-aproksymacyjnym* ($k \geq 1$), dla problemu minimalizacji (maksymalizacji) Π jeżeli dla każdych danych wejściowych Z zwraca on rozwiązanie $x \in sol(Z)$ takie, że:

$$F(x) \leq kF(x^*) \qquad \left(F(x) \leq \frac{1}{k}F(x^*)\right),$$

gdzie x^* jest rozwiązaniem optymalnym dla danych Z

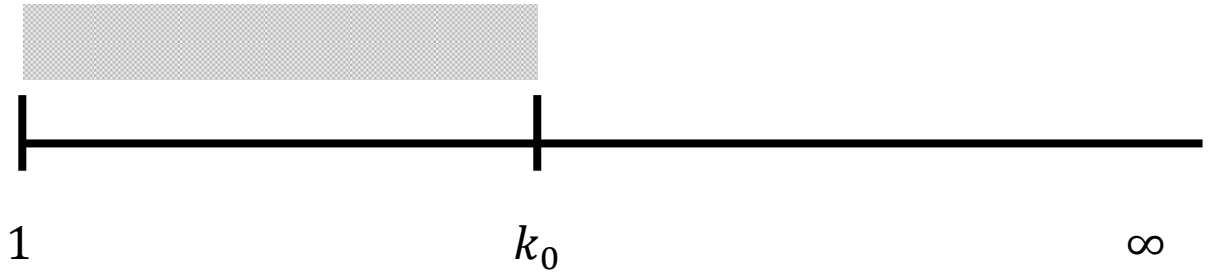
Schematy aproksymacyjne (4)

Ważne:

1. im mniejsze k tym lepszy algorytm k-aproksymacyjny. Algorytm 1-aproksymacyjny jest algorytmem wielomianowym dokładnym dla problemu Π. Dla problemów $\mathcal{NP}$–trudnych algorytm taki nie istnieje, jeżeli $\mathcal{P} \neq \mathcal{NP}$.
2. Algorytm k-aproksymacyjny jest efektywny – *szybko* zwraca rozwiązanie.
3. Algorytm k-aproksymacyjny daje gwarancję, że skonstruowane rozwiązanie będzie co najwyżej k razy gorsze od optymalnego.

Schematy aproksymacyjne (5)

Kres dolny (infimum) k_0 po wszystkich wartościach k, dla których istnieje algorytm k-aproksymacyjny dla problemu Π nazywamy *progiem aproksymacji* dla Π.



Poniżej progu k_0 aproksymacja staje się $\mathcal{NP}$–trudna, czyli obliczeniowo równie trudna, co znalezienie rozwiązania optymalnego.

Schematy aproksymacyjne (6)

Dla $\mathcal{NP}$–trudnego algorytmu optymalizacyjnego Π możliwe są trzy przypadki:

1. $k_0 \in (1, \infty)$.
2. $k_0 = \infty$ (problem nieproksymowany)
3. $k_0 = 1$ (problem dowolnie aproksymowany)

Schematy aproksymacyjne (7)

przykłady dla $k_0 \in (1, \infty)$

minimalne pokrycie wierzchołkowe (V-COVER)

Dany jest graf $G = (V, E)$. Pokryciem wierzchołkowym grafu G nazywamy podzbiór wierzchołków $W \subseteq V$ taki, że każda krawędź $e \in E$ przylega do co najmniej jednego wierzchołka należącego do W. Należy wyznaczyć najmniejsze pokrycie wierzchołkowe.

Schematy aproksymacyjne (8)

we: spójny graf $G = (V, E)$.

wy: podzbiór wierzchołków $W \subseteq V$.

1: $W \leftarrow \emptyset$

2: while G zawiera przynajmniej jedną krawędź do

3: wybierz dowolną krawędź (u, v) w G

4: $W \leftarrow W \cup \{u, v\}$

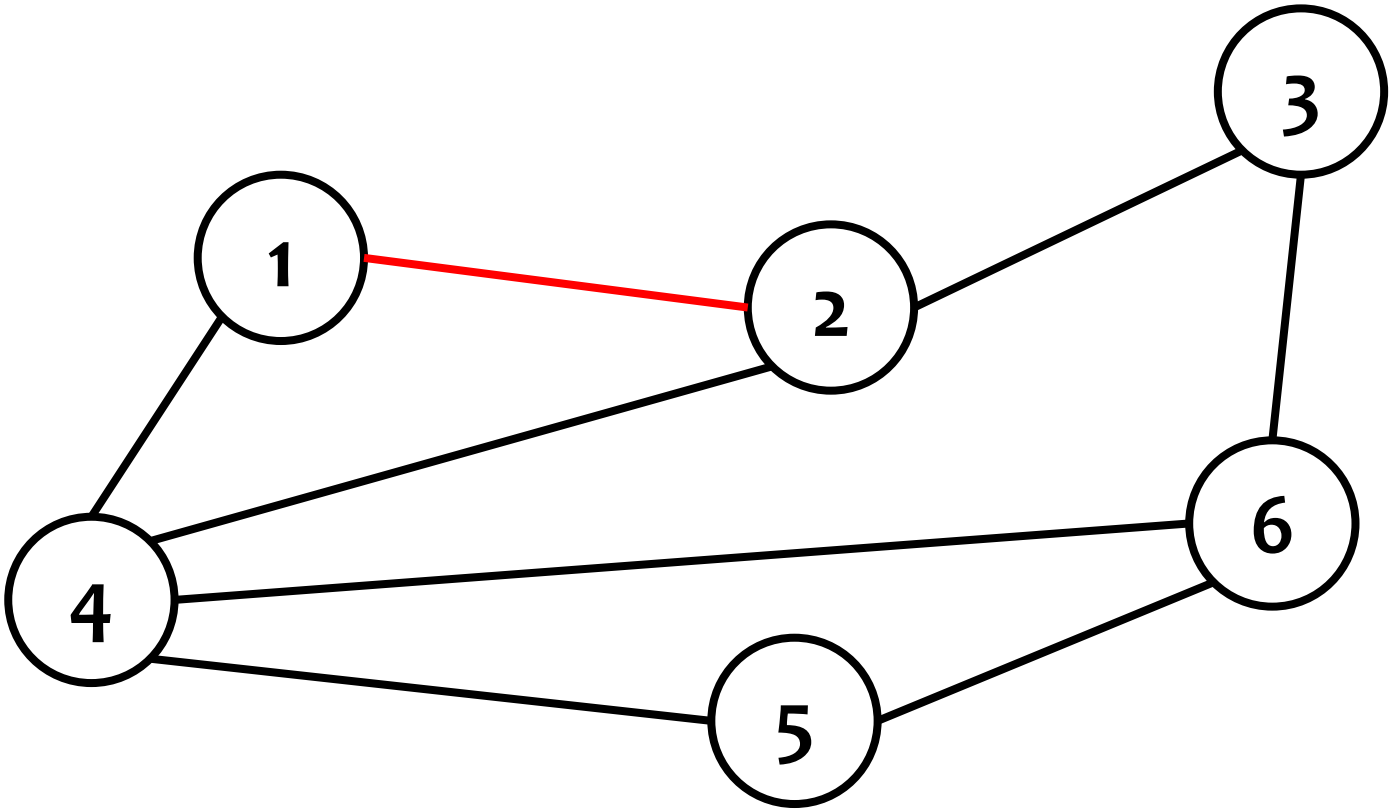
5: usuń wszystkie krawędzie pokryte przez u i v z G

6: end while

7: return W

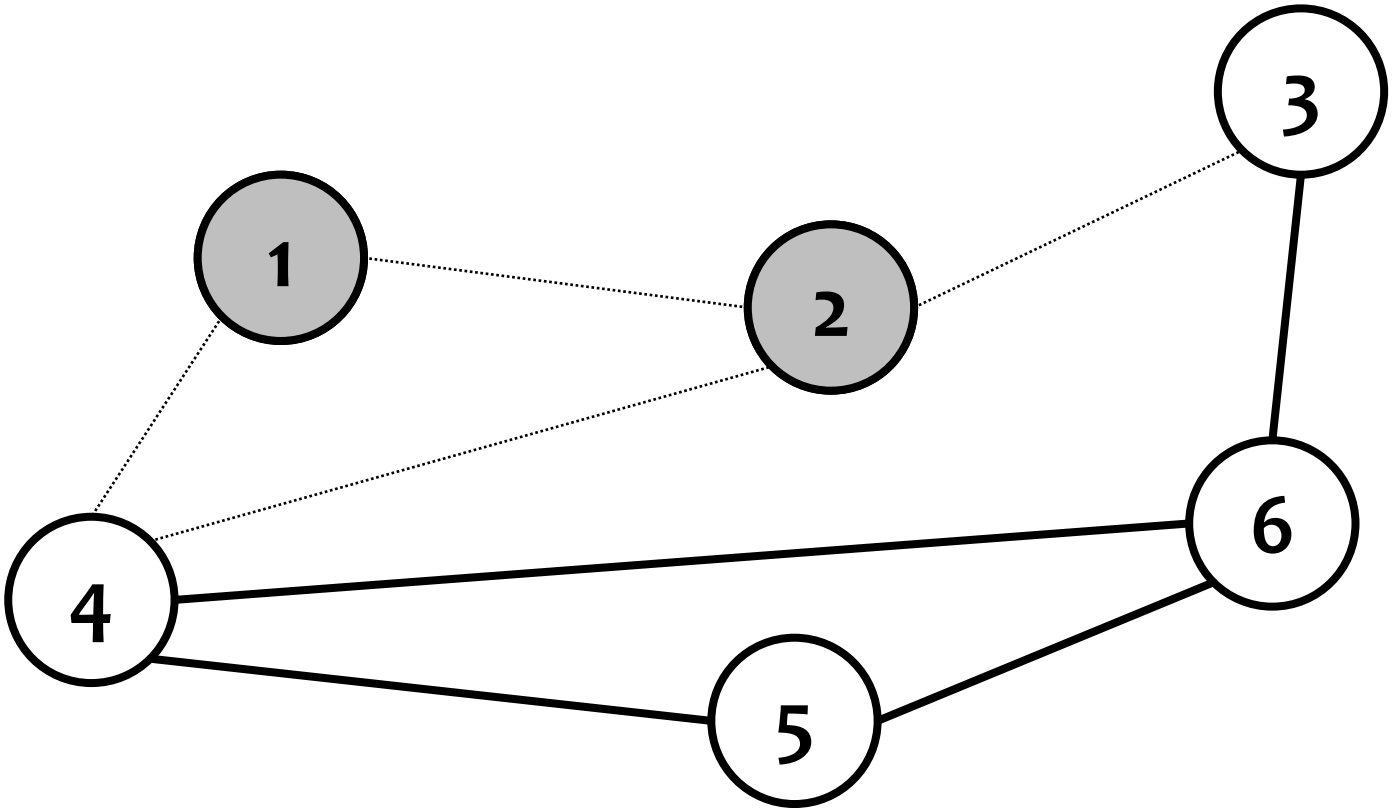
Schematy aproksymacyjne (9)

wybieramy krawędź (1,2)



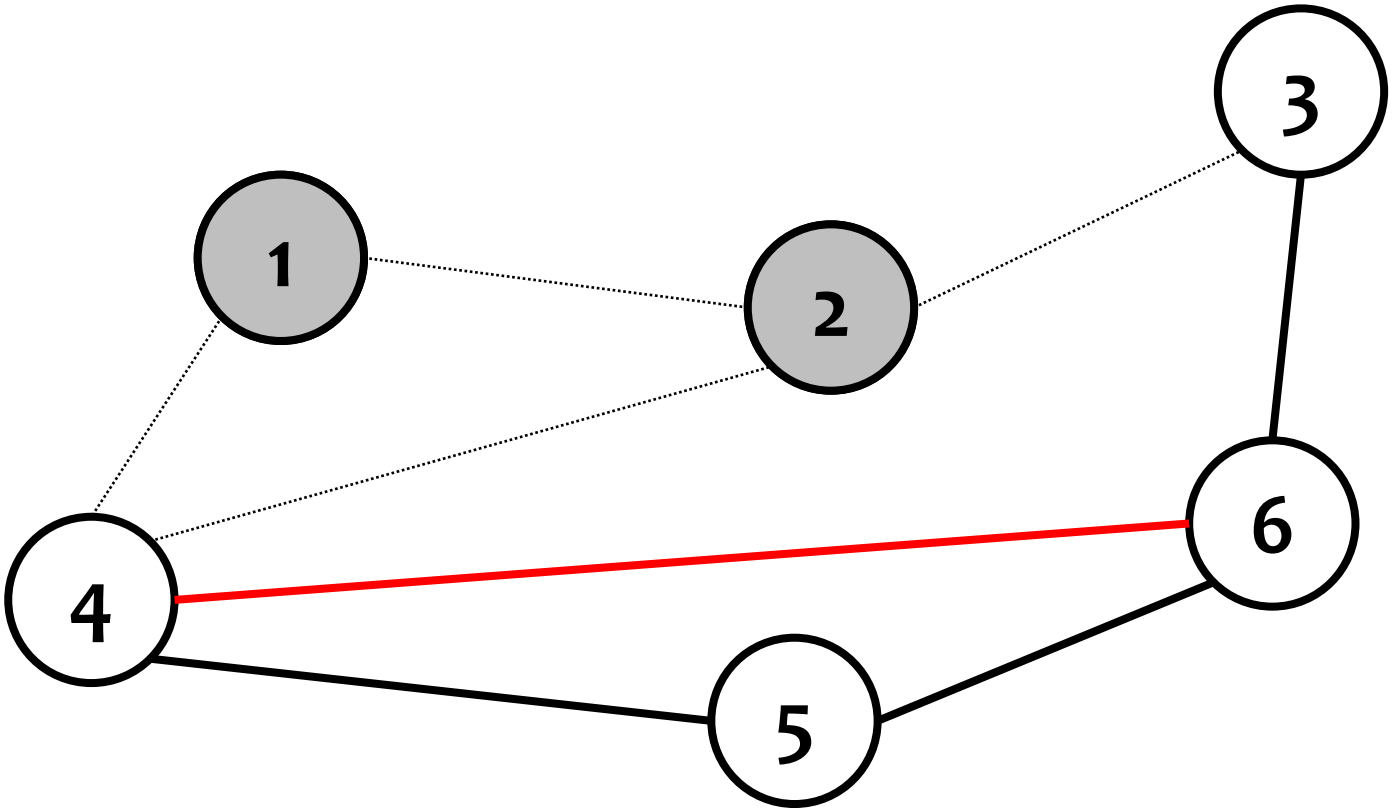
Schematy aproksymacyjne (10)

dodajemy wierzchołki 1 i 2 do pokrycia i usuwamy wszystkie krawędzie pokryte przez 1 i 2



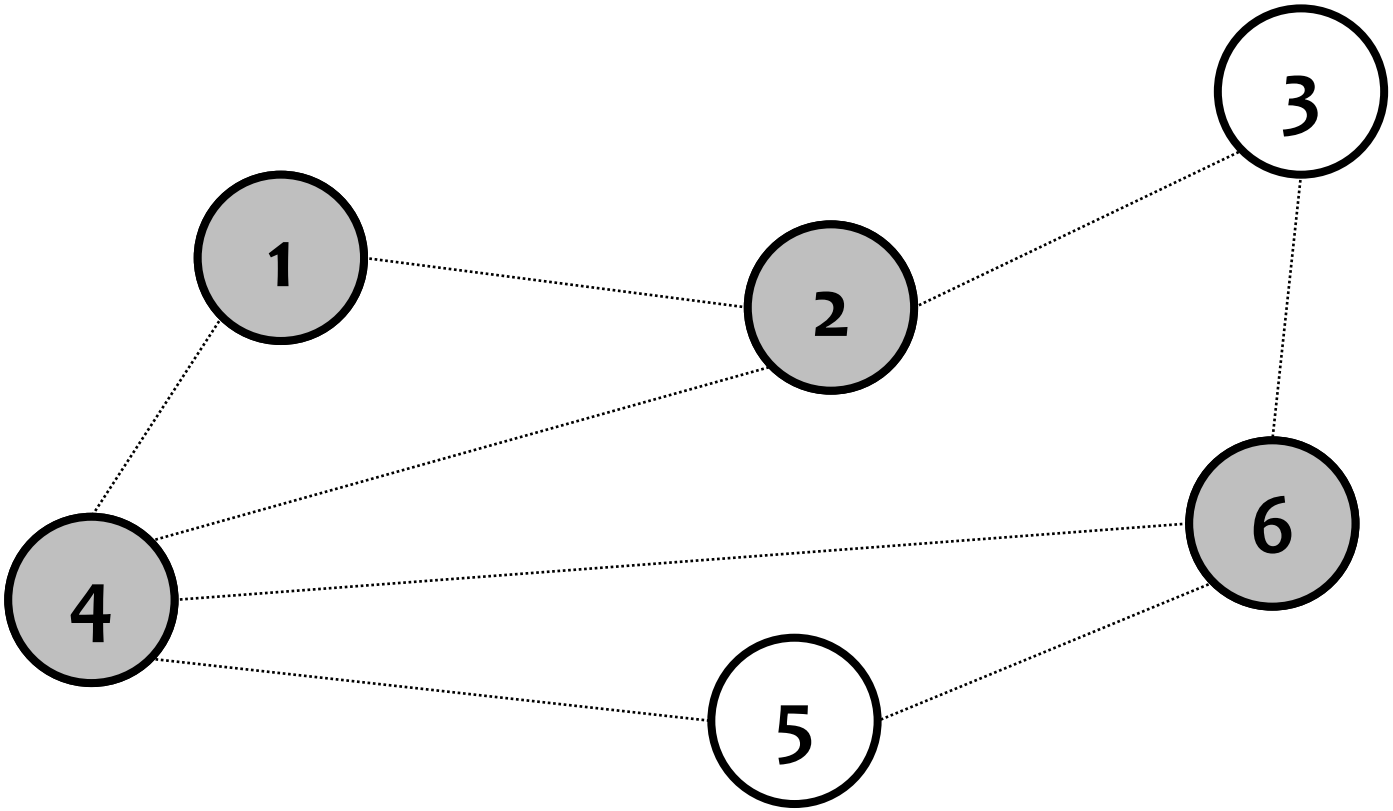
Schematy aproksymacyjne (11)

wybieramy krawędź (4,6)

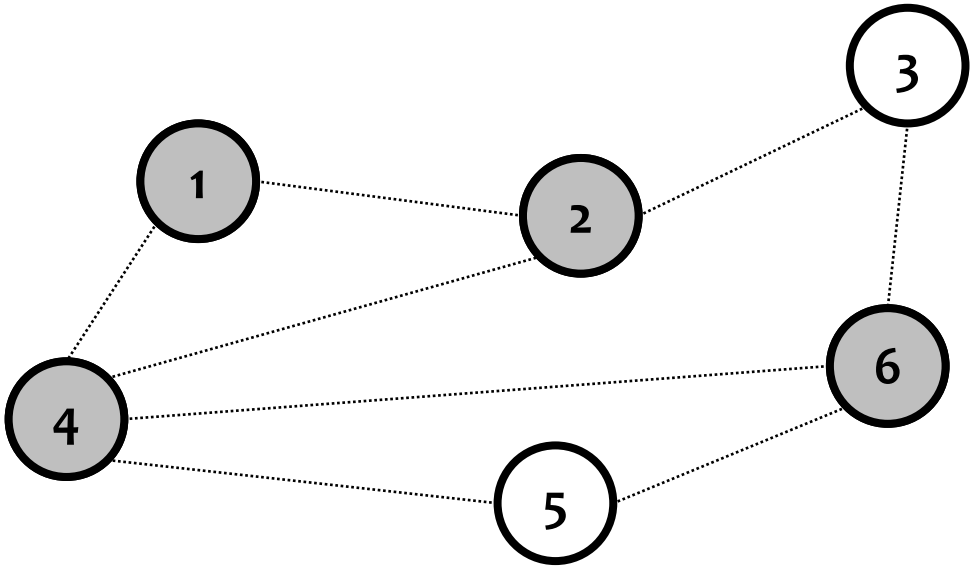


Schematy aproksymacyjne (12)

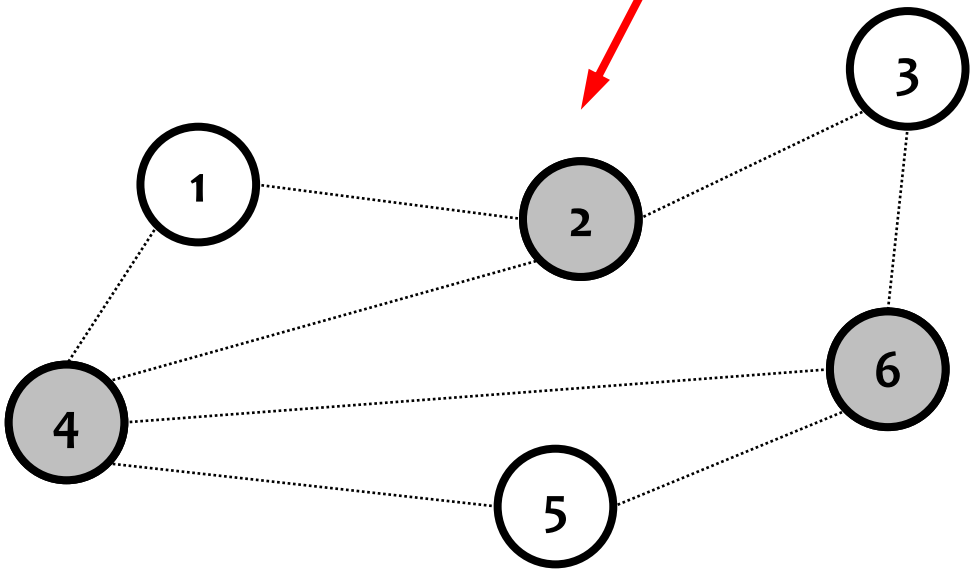
dodajemy wierzchołki 4 i 6 do pokrycia i usuwamy wszystkie krawędzie pokryte przez 4 i 6



Schematy aproksymacyjne (13)



pokrycie minimalne



Schematy aproksymacyjne (14)

Algorytm V-COVER(APRX)

1. jest wielomianowy; złożoność $O(|E|)$.
2. jeżeli krawędź (u, v) została wybrana w **3:**, to u lub v musi znaleźć się w każdym pokryciu wierzchołkowym grafu G (w szczególności w pokryciu W^*). W przeciwnym wypadku krawędź (u, v) nie byłaby pokryta. Oznacza to, że…

Schematy aproksymacyjne (15)

Algorytm V-COVER(APRX)

1. jest wielomianowy; złożoność $O(|E|)$.
2. jeżeli krawędź (u, v) została wybrana w 3:, to u lub v musi znaleźć się w każdym pokryciu wierzchołkowym grafu G (w szczególności w pokryciu W^*). W przeciwnym wypadku krawędź (u, v) nie byłaby pokryta. Oznacza to, że…

$$\boldsymbol{|W| \leq 2|W^*|},$$

czyli…

Schematy aproksymacyjne (16)

Algorytm V-COVER(APRX)

1. jest wielomianowy; złożoność $O(|E|)$.
2. jeżeli krawędź (u, v) została wybrana w 3:, to u lub v musi znaleźć się w każdym pokryciu wierzchołkowym grafu G (w szczególności w pokryciu W^*). W przeciwnym wypadku krawędź (u, v) nie byłaby pokryta. Oznacza to, że…

$$|W| \leq 2|W^*|,$$

czyli **algorytm jest 2-aproksymacyjny**

Schematy aproksymacyjne (17)

A jaki jest próg aproksymacji dla V-COVER?

Schematy aproksymacyjne (18)

Dla problemu V-COVER próg aproksymacji należy do przedziału $[1.1666, 2]$.

Jego dokładna wartość nie jest znana.

Schematy aproksymacyjne (19)

metryczny problem komiwojażera (T-TSP)

Dany jest zbiór miast $V = \{1, \dots, n\}$. *Przez* c_{ij}

oznaczymy odległość między miastami $i, j \in V$. **Założenie:** odległości te spełniają nierówność trójkąta:

$$c_{ij} + c_{jk} \geq c_{ik} \qquad (1)$$

dla wszystkich $i, j, k \in V$.

Należy wyznaczyć trasę o najmniejszej odległości.

Schematy aproksymacyjne (20)

rozpatrzmy następujący algorytm aproksymacyjny dla problemu T-TSP.

k1: Skonstruuj MST T dla zbioru miast V.

Koszt $F(T) = \sum_{(i,j)\in T} c_{ij}$ jest nie większy niż koszt optymalnej trasy Π^*,

$$F(T) \leq F(\pi^*).$$

a co gdyby $F(T) > F(\Pi^*)$?

Schematy aproksymacyjne (21)

k2: Zastąp każdą krawędź w T dwoma krawędziami. Wyznacz cykl *Eulera* L w otrzymanym grafie. Koszt cyklu L jest dwukrotnie większy niż koszt drzewa T, czyli:

$$F(L) = 2F(T).$$

Schematy aproksymacyjne (22)

k3: Przekształć cykl *Eulera* L w trasę komiwojażera π usuwając kolejne wystąpienia każdego wierzchołka z L.

Korzystając z (1) otrzymujemy, że:

$$F(\pi) \leq F(L).$$

Schematy aproksymacyjne (23)

wynika to z faktu, że usuwając miasto j z cyklu Eulera postaci $(\ldots i, j, k \ \ldots)$ dodajemy krawędź (i, k) i usuwamy krawędzie (i, j) oraz (j, k). Z (1) wynika, że operacja ta nie powiększa kosztu konstruowanej trasy. Otrzymujemy zatem:

$$F(\pi) \leq F(L) = 2F(T) \leq 2F(\pi^*).$$

Ilu-aproksymacyjny jest to algorytm? Czy jest wielomianowy?

Schematy aproksymacyjne (24)

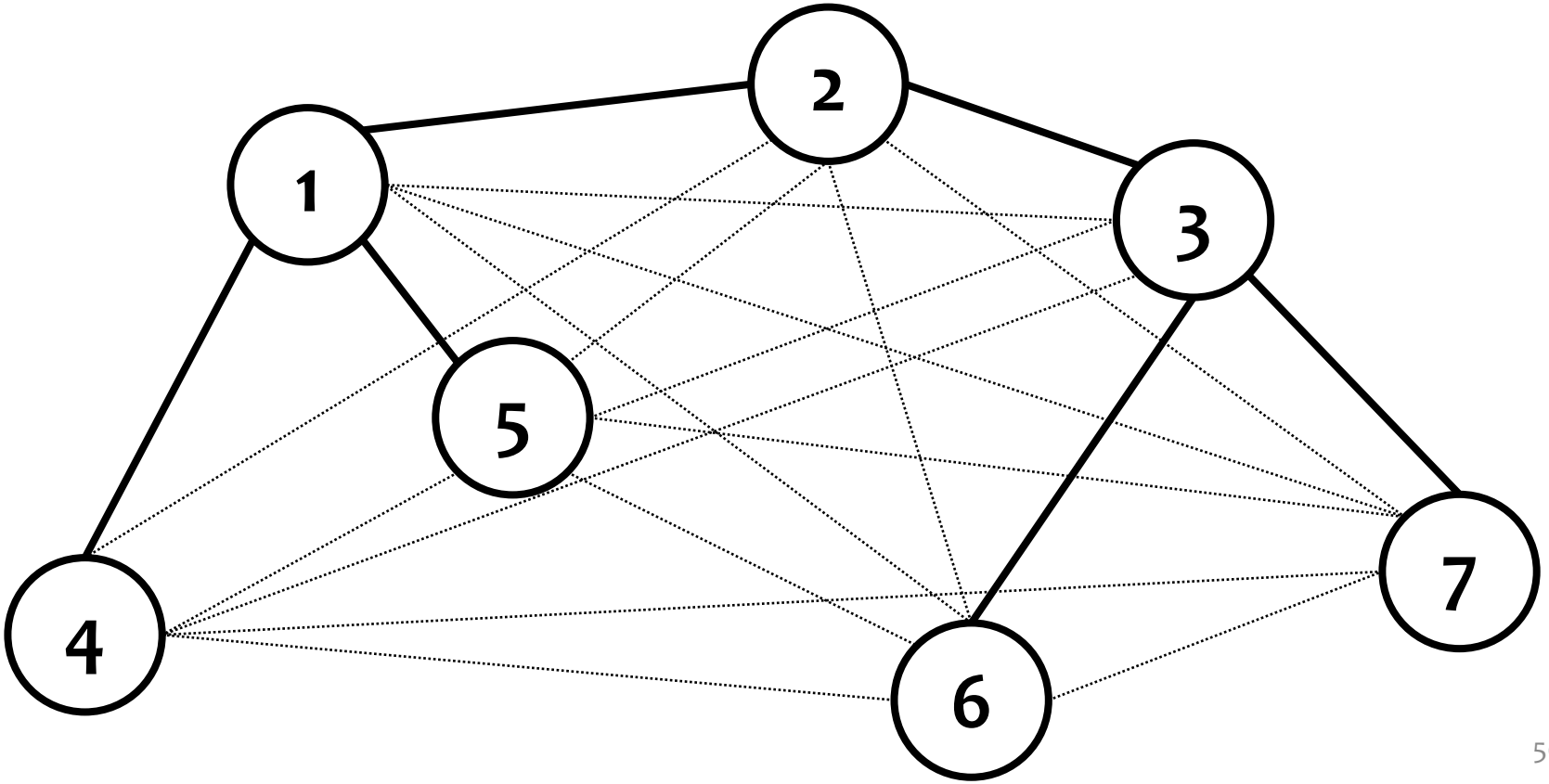
Powyższy algorytm jest **2-aproksymacyjny**.

Jest wielomianowy, ponieważ zarówno TSP jak i cykl Eulera można wyznaczyć w czasie wielomianowym.

A jak działa?

Schematy aproksymacyjne (25)

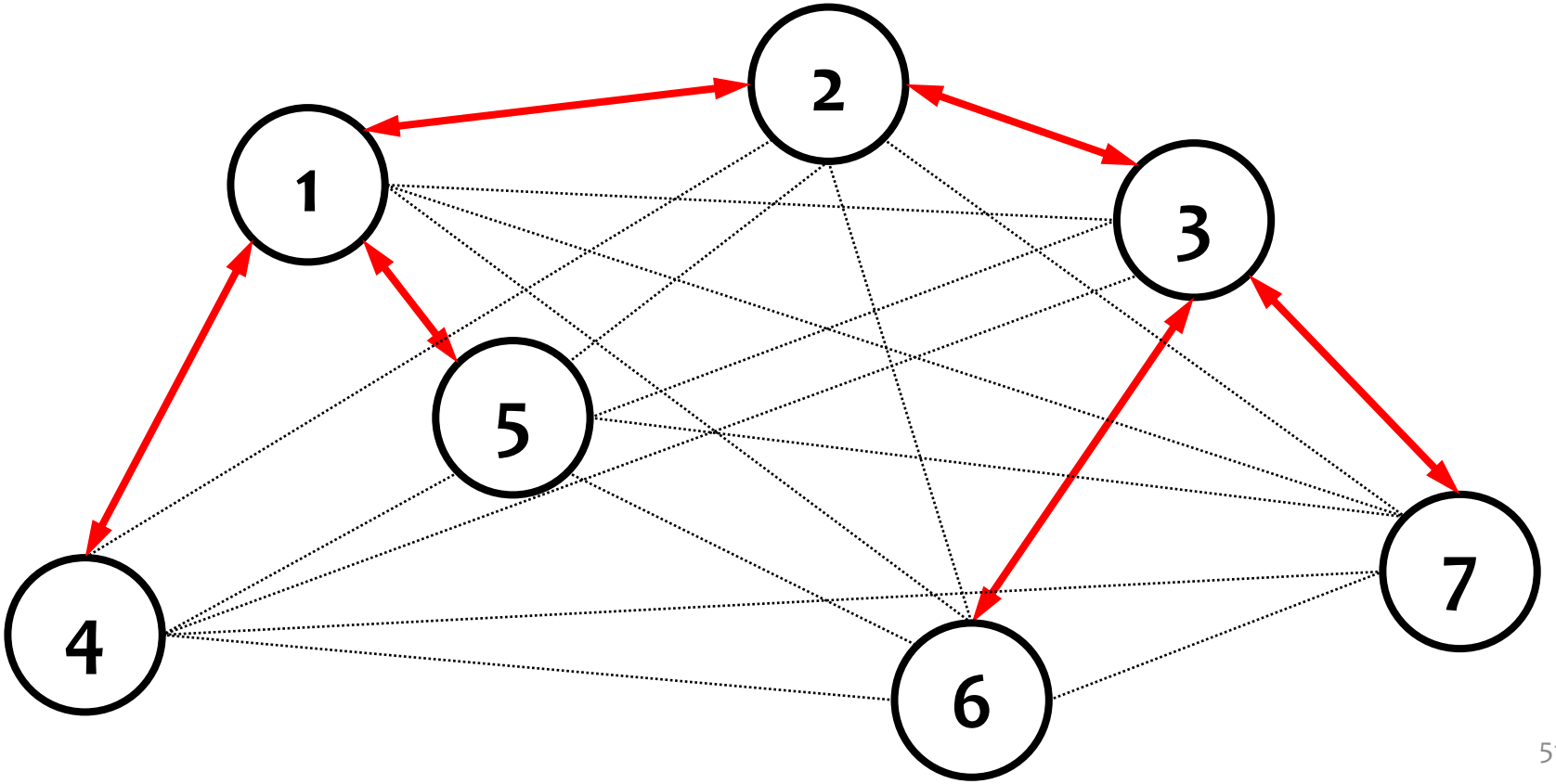
wyznaczamy TSP T



Schematy aproksymacyjne (26)

podwajamy krawędzie w T i konstruujemy cykl Eulera

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



Schematy aproksymacyjne (27)

Usuwamy z L kolejne wystąpienia każdego wierzchołka…

s1: $L = (1, 4, 1, 5, 1, 2, 3, 6, 3, 7, 3, 2, 1)$

Schematy aproksymacyjne (28)

Usuwamy z L kolejne wystąpienia każdego wierzchołka…

s1: $L = (1, 4, 1, 5, 1, 2, 3, 6, 3, 7, 3, 2, 1)$

s2: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, 3, 7, 3, 2, \mathrm{X})$

Schematy aproksymacyjne (29)

Usuwamy z L kolejne wystąpienia każdego wierzchołka…

s1: $L = (1, 4, 1, 5, 1, 2, 3, 6, 3, 7, 3, 2, 1)$

s2: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, 3, 7, 3, 2, \mathrm{X})$

s3: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, 3, 7, 3, \mathrm{X}, \mathrm{X})$

Schematy aproksymacyjne (30)

Usuwamy z L kolejne wystąpienia każdego wierzchołka…

s1: $L = (1, 4, 1, 5, 1, 2, 3, 6, 3, 7, 3, 2, 1)$

s2: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, 3, 7, 3, 2, \mathrm{X})$

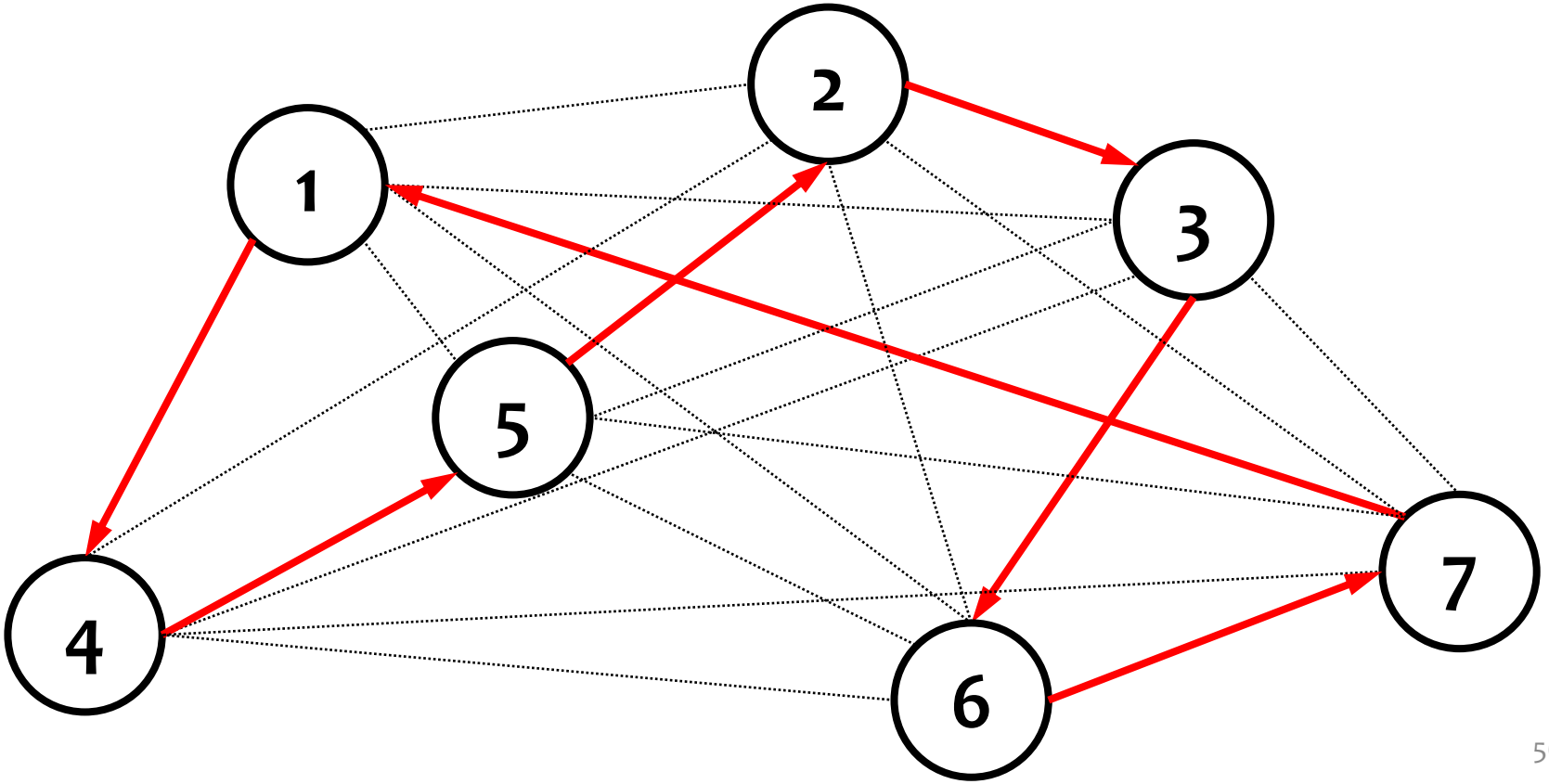
s3: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, 3, 7, 3, \mathrm{X}, \mathrm{X})$

s3: $L = (1, 4, \mathrm{X}, 5, \mathrm{X}, 2, 3, 6, \mathrm{X}, 7, \mathrm{X}, \mathrm{X}, \mathrm{X})$

Schematy aproksymacyjne (31)

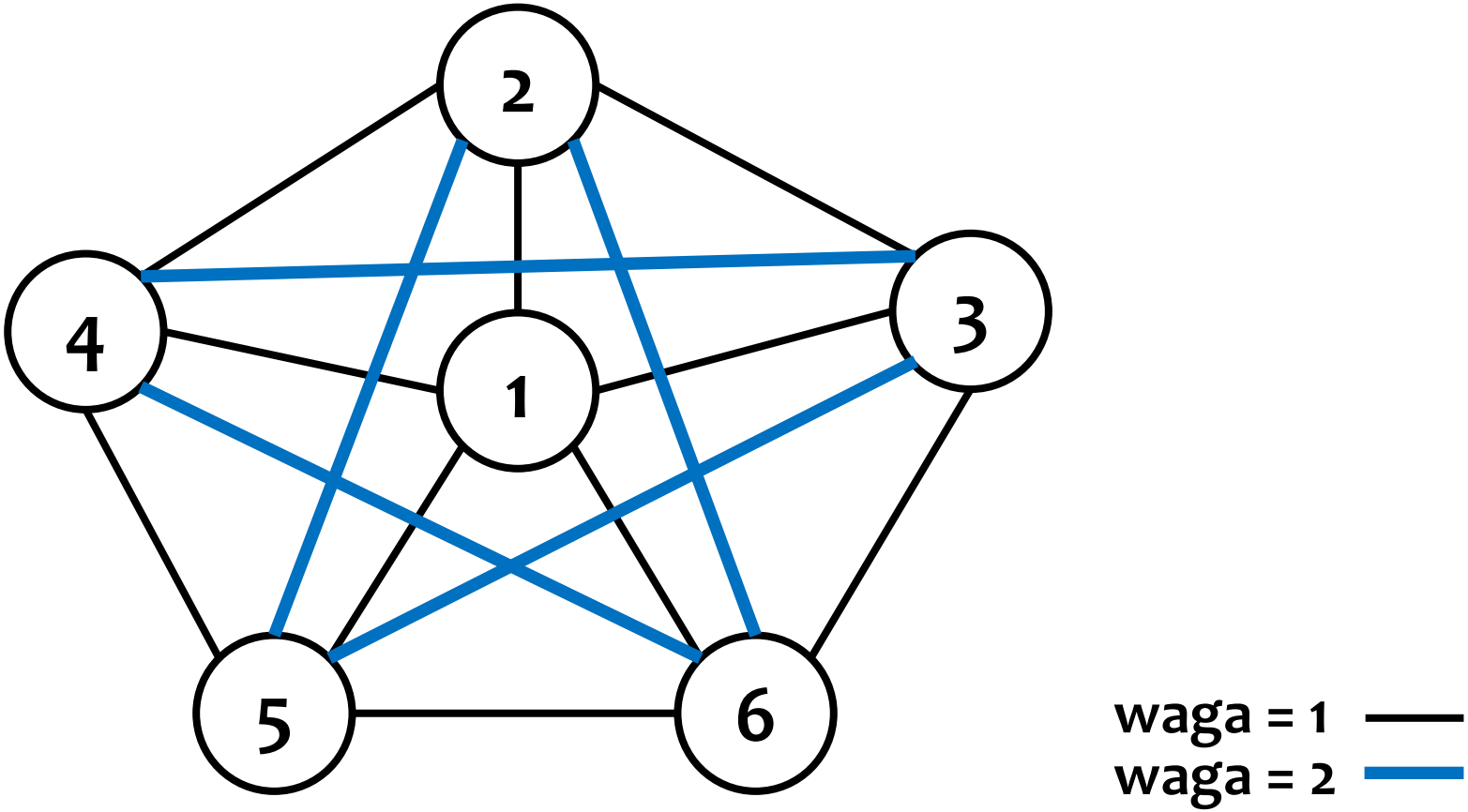
otrzymując trasę komiwojażera

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



Schematy aproksymacyjne (32)

A co z takim przypadkiem?



Schematy aproksymacyjne (33)

Istnieje lepszy algorytm aproksymacyjny dla problemu T-TSP. Jest to algorytm $^{3}/_{2}$**-aproksymacyjny** [Christofides 1976].

Próg aproksymacji dla tego problemu należy do przedziału $[\frac{2805}{2804}, 3/2]$.

Nicos Christofides, *Worst-case analysis of a new heuristic for the travelling salesman problem*, Report 388, Graduate School of Industrial Administration, CMU, 1976

Schematy aproksymacyjne (34)

Problem pakowania (PACK)

Dany jest zbiór przedmiotów $J = \{1, \dots, n\}$ o rozmiarach $a_i > 0, i \in J$.

$$J = \{1, 2, 3, 4, 5\}, \qquad \mathrm{A} = \{2, 1, 3, 4, 1\}$$

Należy wyznaczyć zapakowanie przedmiotów do jak najmniejszej liczby skrzyń $\mathrm{C} = 5$.

Schematy aproksymacyjne (35)

Algorytm jest wielomianowy, **2-aproksymacyjny**.

Gdyby jednak rozpatrywać przedmioty w kolejności niemalejących rozmiarów, prowadziłoby przeważnie to lepszych rezultatów [Johnson et. al. 1974].

D.S. Johnson et. al., Worst-Case performance bounds for simple one-dimensonal packing algorithms, SIAM Journal of Computing (1974), 3, 299-326

Schematy aproksymacyjne (36)

maksymalna spełnialność (MAX-SAT)

Dana jest formuła logiczna F :

$C_1 \wedge C_2 \wedge \ldots \wedge C_m$, gdzie C_i jest alternatywą pewnej liczby zmiennych logicznych lub ich negacji. Niech x będzie pewnym wartościowaniem logicznym zmiennych występujących w F. Przez $C_i(x)$ oznaczymy wartość logiczną klauzuli C_i dla wartościowania x.

Należy wyznaczyć wartościowanie x, dla którego wartość $\sum_{i=1}^{m} w_i C_i(x)$ jest maksymalny.

Schematy aproksymacyjne (37)

rozważmy formułę

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_1} \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4)$$

wagi klauzul wynoszą: $1, 1, 2, 3$.

Jak zmaksymalizować wagę spełnionej formuły F?

Schematy aproksymacyjne (38)

Można udowodnić, że powyższy algorytm MAX-SAT jest **2-aproksymacyjny**. Ale, można skonstruować lepszy algorytm $4/3$-**aproksymacyjny** [V. Vazirani, Algorytmy Aproksymacyjne].

Schematy aproksymacyjne (39)

Problem szeregowania $(P\|prec\|C_{max})$

Dany jest zbiór $J = \{1, \dots, n\}$ zadań z czasami wykonania $p_i, i \in J$, które mają być wykonywane na m maszynach.

Zadane są ograniczenia kolejnościowe w zbiorze prac, tj. jeżeli $i \to j$, to zadanie i musi być wykonane przed zadaniem j.

Należy znaleźć dopuszczalny harmonogram wykonania prac na maszynach o minimalnym czasie wykonania.

Schematy aproksymacyjne (40)

Nieaproksymowalność $(\boldsymbol{k_0 = \infty})$

Czy dla każdego $\mathcal{NP}$–trudnego problemu można skonstruować algorytm k-aproksymacyjny dla pewnej wartości $k > 1$?

Jeżeli $\mathcal{P} \neq \mathcal{NP}$, to nie można.

W takim przypadku nie istnieją algorytmy k -aproksymacyjne dla:

1. ogólny problem TSP
2. programowanie liniowe całkowitoliczbowe PLC
3. programowanie liniowe 0-1 PLC

Schematy aproksymacyjne (41)

Wielomianowe schematy aproksymacyjne $(\boldsymbol{k_0 = 1})$

Jeżeli pewien $\mathcal{NP}$–trudny problem optymalizacyjny posiada algorytm k-aproksymacyjny, to czy problem ten można aproksymować z dowolną dokładnością?

Czy liczba k może być dowolnie bliska 1?

Jeżeli tak, to mówimy że problem posiada wielomianowy schemat aproksymacji.

Schematy aproksymacyjne (42)

Rodzinę algorytmów $A_\varepsilon, 0 < \varepsilon < 1$, nazywamy wielomianowym schematem aproksymacji dla problemu Π jeżeli:

1. Algorytm A_ε jest $(1+\varepsilon)$-aproksymacyjnym algorytmem dla problemu Π.
2. Algorytm A_ε jest wielomianowy dla ustalonego $\varepsilon \in (0,1)$.

Schematy aproksymacyjne (43)

Wielomianowy schemat aproksymacji A_ε nazywamy w pełni wielomianowym schematem aproksymacji jeżeli złożoność algorytmu A_ε jest wielomianem zależnym od rozmiaru problemu i od wartości $1/\varepsilon$.

Schematy aproksymacyjne (44)

Algorytm A jest ***wielomianowym schematem aproksymacyjnym*** jeśli dla każdego ustalonego ε posiada on wielomianową złożoność obliczeniową.

Jeżeli dodatkowo ta złożoność jest wielomianem od $1/\varepsilon$, to A jest nazywany ***w pełni wielomianowym schematem aproksymacyjnym***.

W praktyce stopień wielomianu złożoności obliczeniowej rośnie bardzo szybko przy ε dążącym do zera.

Powoduje to że schematy aproksymacyjne są ciągle algorytmami mało konkurencyjnymi w porównaniu do metod korzystnych eksperymentalnie.

Schematy aproksymacyjne (45)

Wielomianowe schematy aproksymacyjne (***P**olymonial **T**ime **A**pproximation **S**cheme*) – $\boldsymbol{PTAS}$: czas działania algorytmu A_ε jest ograniczony od góry przez wielomian od rozmiaru instancji $N(I)$oraz funkcję wykładniczą od wartości błędu $1/\varepsilon$ czyli

$$TIME(A_\varepsilon) = O\left(p(N(I)) \cdot e\left(\frac{1}{\varepsilon}\right)\right),$$

gdzie p jest wielomianem oraz e jest funkcją wykładniczą.

np. złożoność A_ε może wynosić $O\left(n^2 \cdot 2^{\frac{1}{\varepsilon}}\right)$

Schematy aproksymacyjne (46)

W pełni wielomianowe schematy aproksymacyjne (***F**ully **P**olymonial **T**ime **A**pproximation **S**cheme*) – $\boldsymbol{FPTAS}$: czas działania algorytmu A_ε jest ograniczony od góry przez wielomian od rozmiaru instancji $N(I)$oraz wielomian od odwrotności błędu $1/\varepsilon$ czyli

$$TIME(A_\varepsilon) = O\left(p\left(N(I), \frac{1}{\varepsilon}\right)\right),$$

gdzie p jest wielomianem.

np. złożoność A_ε może wynosić $O\left(\frac{n^3}{\varepsilon^2}\right)$

Schematy aproksymacyjne (47)

Klasy aproksymacji

Klasa APX składa się z wszystkich problemów optymalizacyjnych, które mają algorytm k –aproksymacyjny dla pewnego ustalonego $k \in (1, \infty)$.

Klasa $PTAS$ składa się z problemów, które mają wielomianowy schemat aproksymacji.

Klasa $FPTAS$ składa się z problemów, które mają w pełni wielomianowy schemat aproksymacji.

$$FPTAS \subset PTAS \subset APX$$

Schematy aproksymacyjne (48)

Podsumowanie

1. TSP i KNAPSACK są $\mathcal{NP}$–trudne. Jednak TSP jest zdecydowanie trudniejszy. W praktyce można dokładnie rozwiązać KNAPSACK dla tysięcy elementów, jednak TSP dla kilkuset miast może być bardzo trudny.
2. Najtrudniejszymi problemami optymalizacyjnymi są te, które nie należą do klasy APX.

Schematy aproksymacyjne (49)

3. Bardzo wiele problemów $\mathcal{NP}$–trudnych należy do klasy APX. Oznacz to, że można efektywnie konstruować rozwiązania co najwyżej k razy gorsze od optimum. Wartość k rzadko jest większa od 3.

4. Najłatwiejszymi problemami $\mathcal{NP}$–trudnymi są problemy należące do klasy $FPTAS$. Niestety problemów takich nie ma wiele.

Schematy aproksymacyjne (50)

Konstrukcja schematów aproksymacyjnych jest oparta na algorytmach programowania dynamicznego. Stosuje się następujące tehniki konstrukcji:

Zaokrąglenie (*Rounding*)

Dekompozycja funkcji kryterialnej (*Objective function decomposition*)

Zmniejszanie przestrzeni stanów (*Interval partitioning*)

2 (p17)

Poszukiwanie lokalne (1)

Bazą metody jest analiza wybranych lub wszystkich rozwiązań leżących w pewnym bliskim otoczeniu $N(X) \subseteq X$ wybranego rozwiązania x.

Analiza dostarcza rozwiązań, które stają się kolejnymi źródłami lokalnych otoczeń, zastępując rozwiązania bieżące i umożliwiając tym samym powtarzanie procesu poszukiwań.

Odpowiednie metody *LS* (*Local Search*) łączą wiele zalet: (a) znaczną szybkość pracy, (b) prostotę implementacji oraz (c) mały błąd do rozwiązania optymalnego.

Uważa się metody te za najbardziej obiecujące dla szczególnie trudnych problemów optymalizacji dyskretnej.

Poszukiwanie lokalne (2)

Termin lokalne poszukiwanie odnosi się do całej grupy metod, w tym także tych najbardziej zaawansowanych "odpornych" na lokalne ekstrema. Zwykle implementacja algorytmów tego typu jest prosta, chociaż niektóre z nich są całkiem nietrywialne.

Wymienić można na przykład:

- **przeszukiwanie zstępujące**,
- **przeszukiwanie losowe**,
- przeszukiwanie snopowe,
- przeszukiwanie progowe,
- **przeszukiwania ewolucyjne**,

Poszukiwanie lokalne (3)

i dalej:

- ewolucja różnicowa,
- podejście immunologiczne,
- symulowane wyżarzanie,
- **Tabu Search**,
- poszukiwanie z zakazami,
- **poszukiwanie mrówkowe**,
- sieci neuronowe,
- poszukiwanie rojem cząstek,
- etc..

3 (p1)