Heuristické metody

- Co očekáváme od praktických heuristik?
- Lokální a globální přístup
- Najít vůbec nějaké řešení a najít lepší – konstruktivní a iterativní fáze
- Jednoduché heuristiky pouze nejlepší a prvé zlepšení
- Problém s lokálními extrémy a co s ním
- Prohledávání s návratem

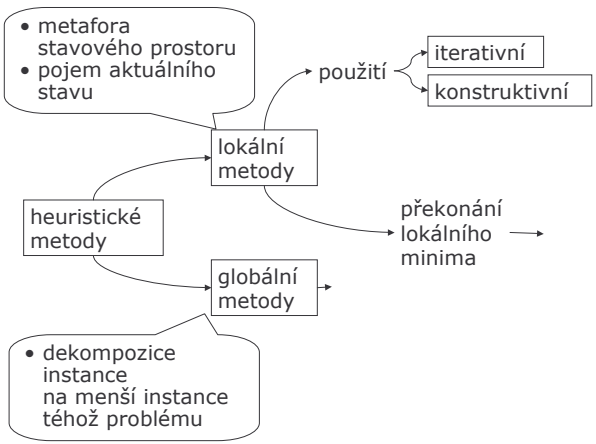
Co očekáváme od heuristických metod

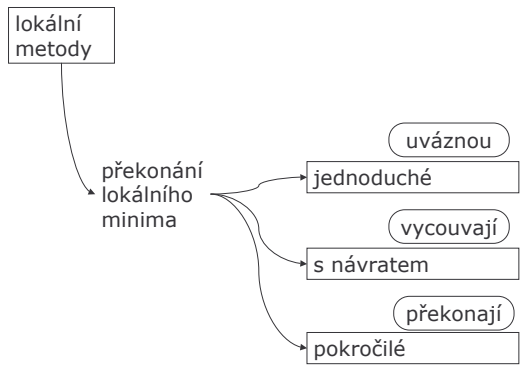
- Použitelnost v praxi:
  - na praktických instancích
  - s přijatelnou přesností
  - s přijatelným výpočetním časem

  na praktických instancích
- Omezení a optimalizace:
  - nejlepší čas pro potřebnou přesnost
  - nejlepší přesnost v časovém limitu

  výpočty v reálném čase
- Přesnost:
  - přibližné řešení
  - přesné řešení

  důkazy správnosti

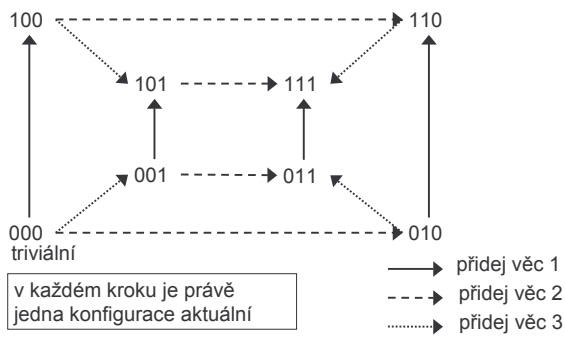




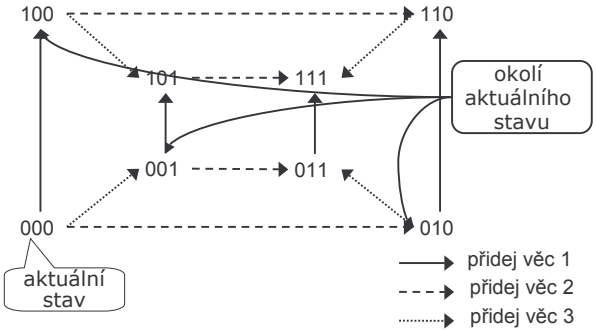
Lokální metody

Stavový prostor a okolí stavu
Systematické a úplné metody
Praktické aplikace: neúplné metody

Stavový prostor instance problému batohu řešené hladovým algoritmem



Lokální metody

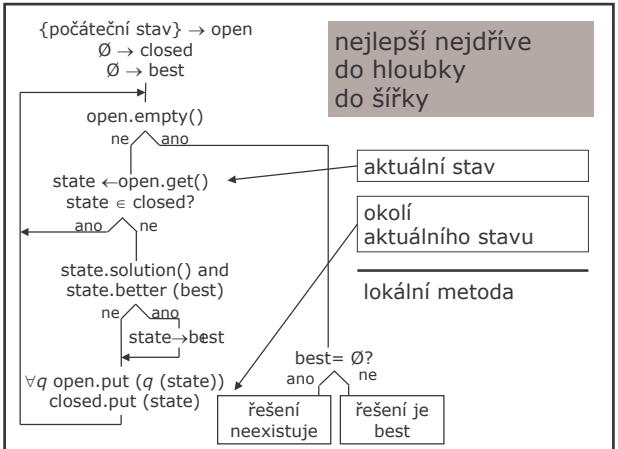


Okolí stavu, sousední stav

- Definice: kam se dostanu jedním krokem
  okolí stavu $s \in S$ je množina stavů, dosažitelných z s aplikací některé operace $q \in Q$.
- Definice: kam se dostanu nejvýše k kroky
  k-okolí stavu $s \in S$ je množina stavů, dosažitelných z s aplikací nejméně jedné a nejvýše k operací $q \in Q$.
- Definice:
  stavy z okolí stavu $s \in S$ se nazývají sousední stavy (sousedé) stavu s.

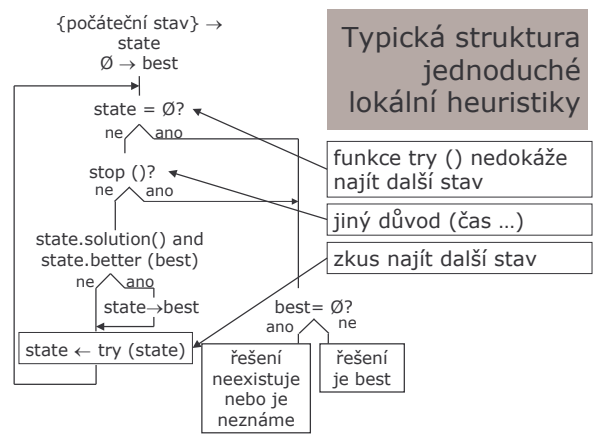
Lokální metoda

- má vždy jeden aktuální stav
- uvažuje sousední stavy z (relativně malého) okolí



Praktické metody

- Úplné a systematické metody uschovají každý stav z okolí pro pozdější zkoumání
  - proto jsou exaktní
  - proto mohou mít nadpolynomiální složitost
- Když vybereme (nějak) z každého okolí jen jeden stav
  - strategie je neúplná, nelze zaručit řešení
  - můžeme najít řešení v přijatelném čase
  - nepotřebujeme strukturu open – má vždy jen jeden prvek
  - strukturu closed si většinou nemůžeme dovolit, musíme se spolehnout na vlastnosti stavového prostoru nebo řízení algoritmu



Příklad

- Dvě jednoduché heuristiky
- Konstruktivní a iterativní fáze
- Dá se to doopravdy použít ...

Problém diskrétního rozmístění

- Dáno:
  - množina n modulů $K=\{k_1, \ldots, k_n\}$
  - množina m pozic $P=\{p_1, \ldots, p_m\}$, $m \geq n$
  - propojení modulů jako hypergraf $G(K, N)$, kde N je množina spojů n
  - cenová funkce $W(R, n)$, která pro každé přiřazení R: $K \rightarrow P$ odhadne cenu realizace spoje n.
- Nalézt:
  - prosté přiřazení R: $K \rightarrow P$ (rozmístění), které minimalizuje součet ohodnocení $W(R, n)$ přes všechny spoje.

Algoritmus max konjunkce min disjunkce

- Nechť $H(G, k_1, k_2)$ je heuristická funkce, která na základě grafu G odhadne stupeň vazby modulů k_1 a k_2.
- Nechť $K^+(R)$ označuje množinu modulů, které jsou právě rozmístěny (mají dvojici v R), $K^-(R)$ množinu právě nerozmístěných modulů.
- Obdobně definujeme množinu obsazených pozic $P^+(R)$ a volných pozic $P^-(R)$.
- $R \leftarrow (k_i, p_j)$, buď ze zadání nebo pomocnou heuristikou.

Algoritmus max konjunkce min disjunkce

1. Vyber modul $k \in K^-(R)$ takový, že $\sum_{l \in K^+(R)} H(G,k,l) = \max.$ (max. přidrátovaný k rozmístěným modulům)
2. Je-li jich více, vyber z nich modul k takový, že $\sum_{l \in K^-(R)} H(G,k,l) = \min.$ (min. přidrátovaný k nerozmístěným modulům)
3. Najdi pozici $p \in P^-(R)$ takovou, že $\sum_{n} W(R \cup (k,p), n) = \min.$ (pozice pro nejlevnější dráty)

   kde suma je přes všechny spoje incidentní s k.
4. $R \leftarrow R \cup (k,p)$
5. Opakuj 1., dokud $K^-(R)$ není prázdná.

Okolí

- Krok: dáme nerozmístěný prvek na volnou pozici.
- Vezmeme $K^-(R)$, najdeme nejlepší prvek podle heuristické funkce.
- Vezmeme $P^-(R)$, najdeme nejlepší prvek podle optimalizačního kritéria.
- „Šidíme" tak hledání nejlepšího prvku $(k,p) \in K^-(R) \times P^-(R)$ (nepotřebovali bychom heuristickou funkci!).

Stavový prostor je acyklický. Algoritmus se zastaví po n krocích.

Konstruktivní heuristika

- Začali jsme ze (skoro) triviální konfigurace
- Dopracovali jsme se k řešení (konfiguraci, která vyhovuje omezením – zde prosté zobrazení R)
- → konstruktivní heuristika

Iterativní zlepšování

1. Najít pozice p_1 a p_2 takové, že vzájemným prohozením jejich „obsahu" (modul nebo nic) se nejvíce zlepší hodnota optimalizačního kritéria.
2. Není taková $\Rightarrow$ stop.
3. Provést záměnu, opakovat 1.

Stavový prostor je acyklický (nemohu provést záměnu, která by nezlepšila optimalizační kritérium – nemohu se vrátit)

Okolí

- Vezmu $P \times P$, najdeme nejlepší prvek podle optimalizačního kritéria
- Můžeme to „šidit":
  - najdeme pozici p_1, která „toho má nejvíc zapotřebí" (heuristická funkce)
  - najdeme pozici p_2, kde je zlepšení největší
- Jiná možnost: hledáme v $P \times P$, ale spokojíme se s první záměnou, která zlepší optimalizační kritérium

Iterativní heuristika

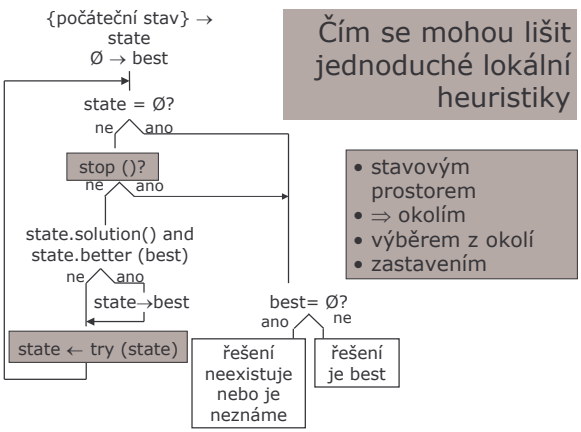
- Začali jsme z řešení
- Dopracovali jsme se k lepšímu řešení
- $\rightarrow$ iterativní heuristika

Dvojfázové heuristiky

- Prvá fáze
  - konstruktivní
  - náhodná, více náhodných řešení (viz GSAT minule)
- Druhá fáze iterativní

Jednoduché heuristiky

Pouze nejlepší
Prvé zlepšení
Okolí heuristik Kernighan-Lin

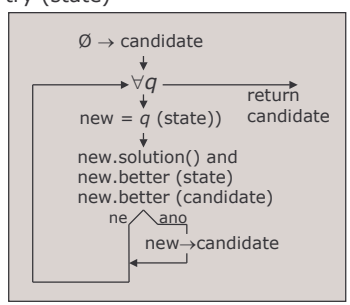


Čím se mohou lišit jednoduché lokální heuristiky

- stavovým prostorem
- $\Rightarrow$ okolím
- výběrem z okolí
- zastavením

Pouze nejlepší
(best only)

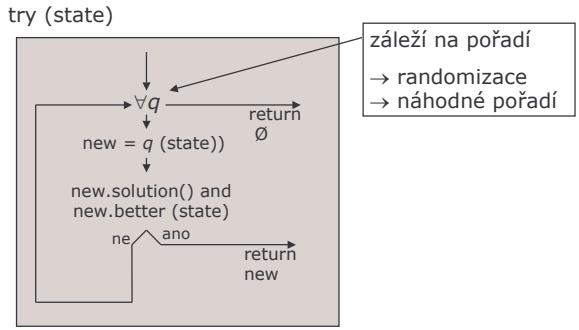
try (state)



Metoda pouze nejlepší

- celá heuristika se zastaví, jestliže v nějakém stavu neexistuje zlepšující tah
- pokud better() používá jiné hodnocení než optimalizační kritérium, solution() může chybět
- jiné názvy: metoda nejrychlejšího sestupu/vzestupu, horolezecký algoritmus
- na pořadí prohledávání okolí nezáleží

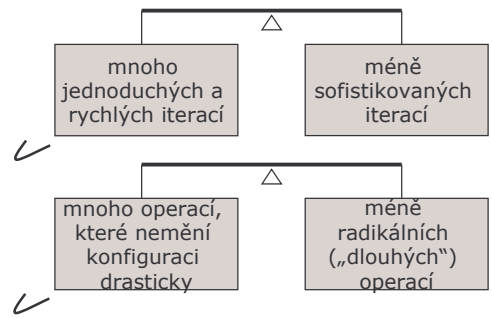
Prvé zlepšení
(first improvement)



Metoda první zlepšení

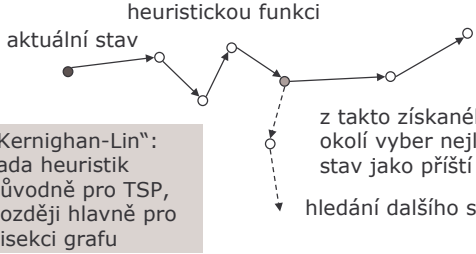
- celá heuristika se zastaví, jestliže v nějakém stavu neexistuje zlepšující tah
- pokud better() používá jiné hodnocení než optimalizační kritérium, solution() může chybět
- na pořadí prohledávání okolí záleží

Návrh heuristiky



Okolí heuristik Kernighan-Lin

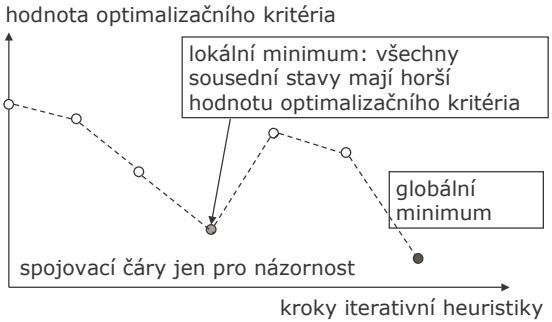
aplikuj opakovaně danou transformaci až do stop podmínky bez ohledu na optimalizační kritérium nebo heuristickou funkci



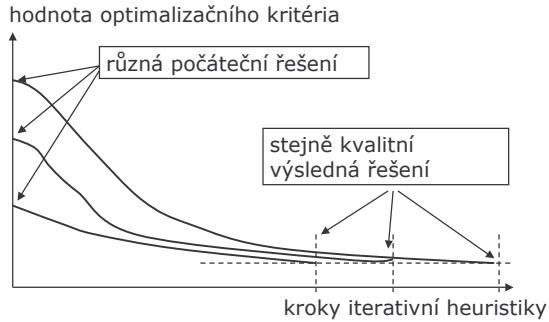
„Kernighan-Lin": řada heuristik původně pro TSP, později hlavně pro bisekci grafu

Problém lokálních extrémů

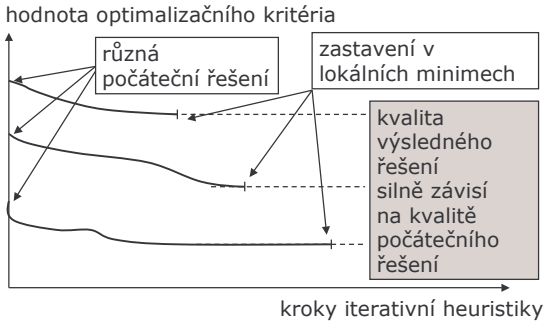
Lokální minimum



Správný průběh iterativní heuristiky



Uváznutí v lokálních minimech



Problém lokálních extrémů

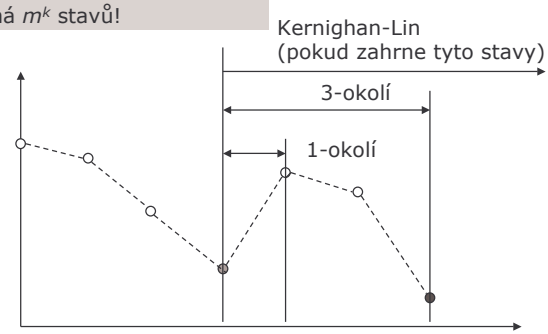
- Heuristiky, které neřeší uváznutí v lokálním extrému (pouze nejlepší, první zlepšení a jim podobné) se zastaví v lokálním minimu
- Projev: závislost výsledného řešení na počátečním, tzv. nedostatečná iterační síla
- Centrální problém lokálních heuristik

Řešení

- Zvětšení okolí
- Více náhodných počátečních řešení (GSAT)
  - odstranění následků, ne příčiny
  - závisí na možnosti generovat náhodná řešení
- Návraty
  - odvolání rozhodnutí, které vedly do slepé uličky
- Únik
  - kroky, které připustí zhoršení aktuálního stavu
  - nutnost dokonalejšího řízení algoritmu (bloudění)
  - nutnost řízení ochoty k horšímu řešení

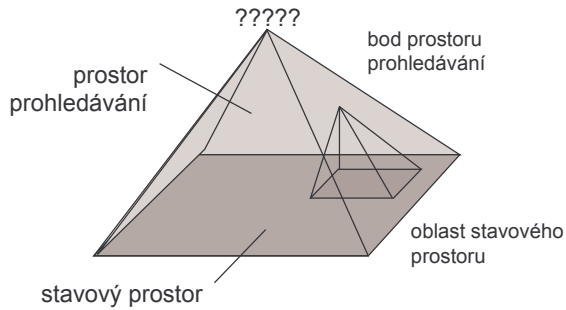
Zvětšení okolí

k-okolí o *m* transformacích má m^k stavů!



Prohledávání s návratem

Prostor prohledávání

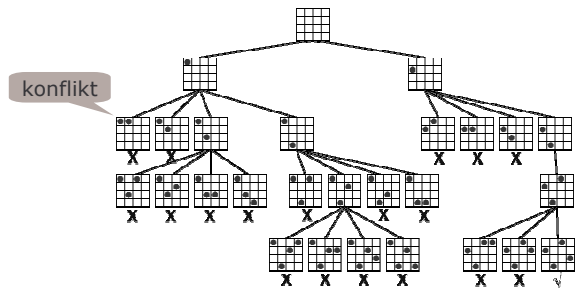


Základní prohledávání s návratem (backtracking)

- V algoritmech, kde se postupně ohodnocují proměnné, nejčastěji konstruktivní úlohy
- Heuristická funkce, udávající pořadí, v jakém mají proměnné být ohodnocovány
- Rozpoznání slepé uličky
  - pro úplné řešení (list prohledávacího stromu)
  - pro částečně ohodnocenou konfiguraci
- Omezení prohledávacího prostoru
- Není zaručena polynomiální složitost
- Tyto techniky se používají i v kombinaci s jinými

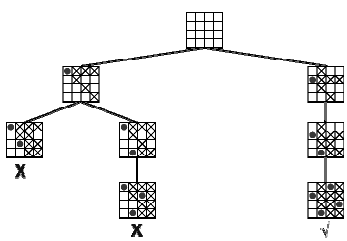
Problém dam na šachovnici

Rozestavit n dam na šachovnici $n \times n$ tak, aby se vzájemně neohrožovaly



Dopředná kontrola (forward checking)

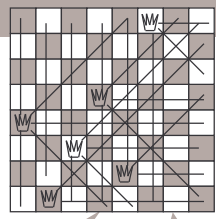
Udržovat informaci, která dovolí (snadno, rychle) se vyhýbat konfliktům



Zpětné značení (backmarking)

Zpětné skoky
(backjumping)

pořadí
umisťování
dam



je třeba vrátit se až sem, jinak budeme narážet na stejný problém

který darebák jako poslední ucpal políčko v tomto sloupci?