Lokální prohledávání

Martin Šůstek

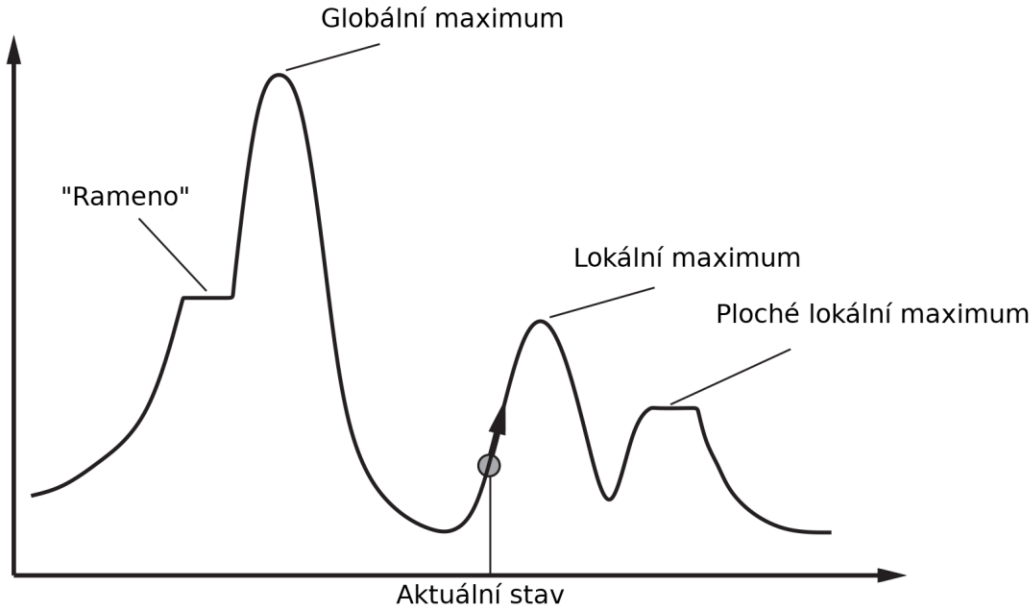
isustek@fit.vutbr.cz

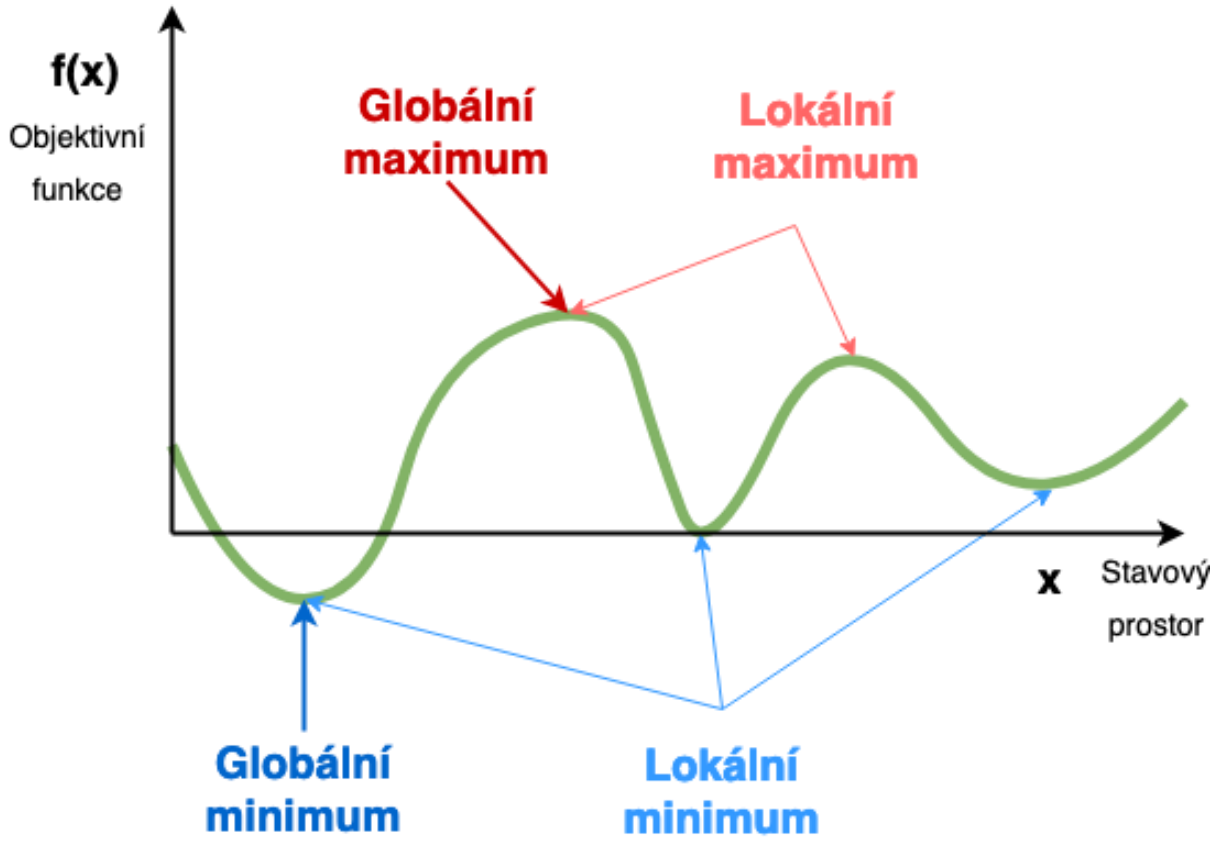
Lokální prohledávání jako alternativa k úplnému prohledávání

- Řešení je **stav**/uzel a ne sekvence akcí
- Neuvažujeme cenu cesty, ale kvalitu stavu
- Neprohledáváme systematicky celý prostor
  - Jeden nebo několik **aktuálních** stavů/uzlů, které vyhodnocujeme a modifikujeme
    - Modifikace skrze **sousednost** stavů (akce, náhodná úprava, kombinace více stavů, ...)
- Malé nároky na paměť
  - Aplikovatelné na velké (i spojité) prostory → systematické prohledávání neaplikov.
- Hledání stavu s ideální hodnotu vyhodnocovací metriky
  - Pokud nalezneme stav s touto hodnotou, ukončujeme prohledávání
- Optimalizační úloha
  - Hledání nejlepšího možného řešení s omezenými prostředky (nelze zkusit vše)
  - Typicky neznáme ideální hodnotu vyhodnocovací metriky

Průběh funkce a tvar stavového prostoru

- Spojitý | Diskrétní





Přehled algoritmů pro lokální prohledávání

- Horolezecký algoritmus (Hill-climbing search)
- Simulované žíhání (Simulated annealing)
- Lokální paprskové prohledávání (Local beam search)
- Genetický algoritmus (Genetic algorithm)

Hill-climbing

- Také označovaný jako greedy local search (steepest-ascent)
- Udržuje jeden stav
  - Pokud je soused, který je lepší → nahrazení; jinak konec algoritmu (lokální extrém)
- Extrémně rychlý
- Lokální extrémy a plateau (část, kde je funkce konstantní) jsou problematické
  - Ridge + diskretizace?

- Úpravy:
  - Stochastic hill-climbing – zvol libovolného lepšího souseda
    - First-choice hill-climbing: kterého lepšího? prvního nalezeného
  - Random-restart hill-climbing – několik běhů s různou inicializací
    - Zvyšujeme pravděpodobnost, že některý lokální extrém bude globální

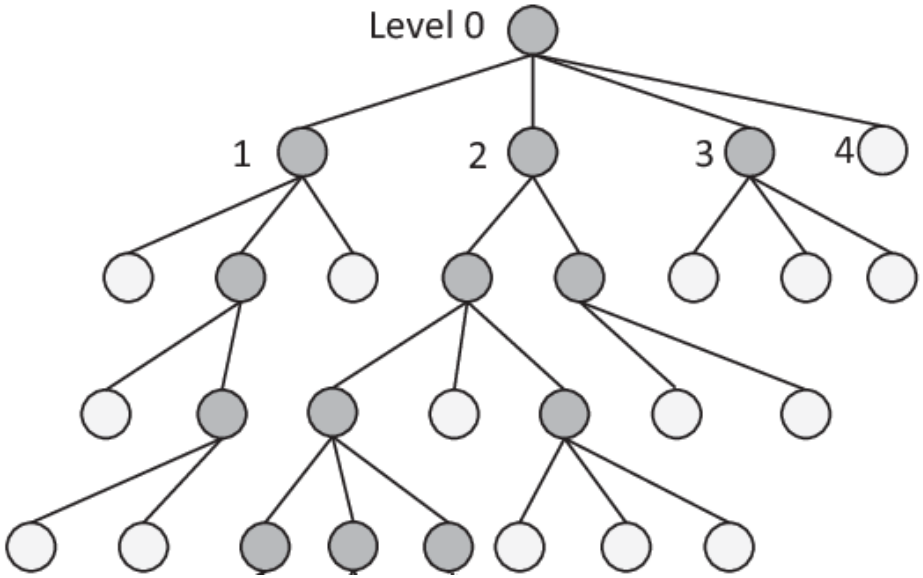
Simulované žíhání

- Hill-climbing + náhodná procházka (náhodný soused)
- Náhodný vyběr souseda, výpočet $\Delta E = E(new) - E(current)$
  - Nový je lepší ($\Delta E \geq 0$) → nahradíme
  - Nový je horší ($\Delta E < 0$) → nahradíme s pravděpodobností $e^{\Delta E/T}$
- Postupné snižování teploty T v každé iteraci (snižujeme prozkoumávání)
- Exploration → exploitation

- Vychází z beam search – algoritmus pro hledání cesty

Beam search

- BFS, kde v každé úrovni ponecháme k nejlepších uzlů
- Využívaný u dekódování pomocí neuronových sítí
  - Strojový překlad
  - Automatické rozpoznávání řeči



Local beam search

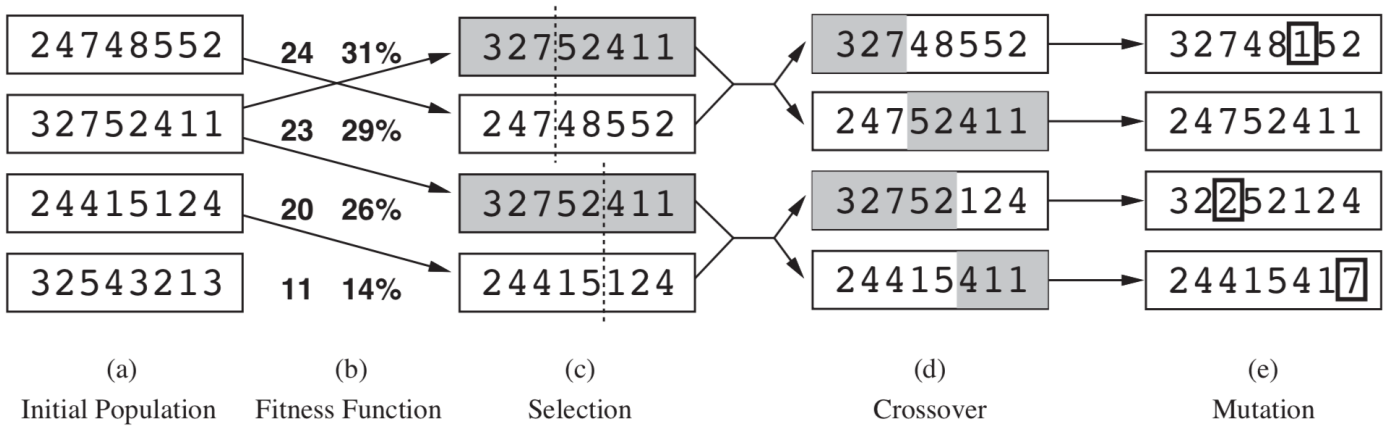
- Local beam search je paralelní beam search inicializovaný několika stavy
- Velmi rychle se ztrácí diverzita
  - Stochastic beam search
    - Následníci mohou být horší stavy, ale lepší ohodnocení by mělo zvyšovat pravděpodobnost výběru souseda
    - Podobná myšlenka jako hill-climbing → simulované žíhání

Genetický algoritmus: Motivace

- Jedinci, kteří jsou *lepší*, mají větší šanci na přežití (rozmnožení) – přirozený výběr
- Pokud se na aktuální uzly díváme jako na populaci jedinců, na sousední uzly jako na potomky, pak **stochastic beam search** připomíná **nepohlavní** rozmnožování
- Genetický algoritmus může být viděň jako **pohlavní** rozmnožování
  - Potřebujeme umět kombinovat 2 stavy, kteří nejsou sousedé

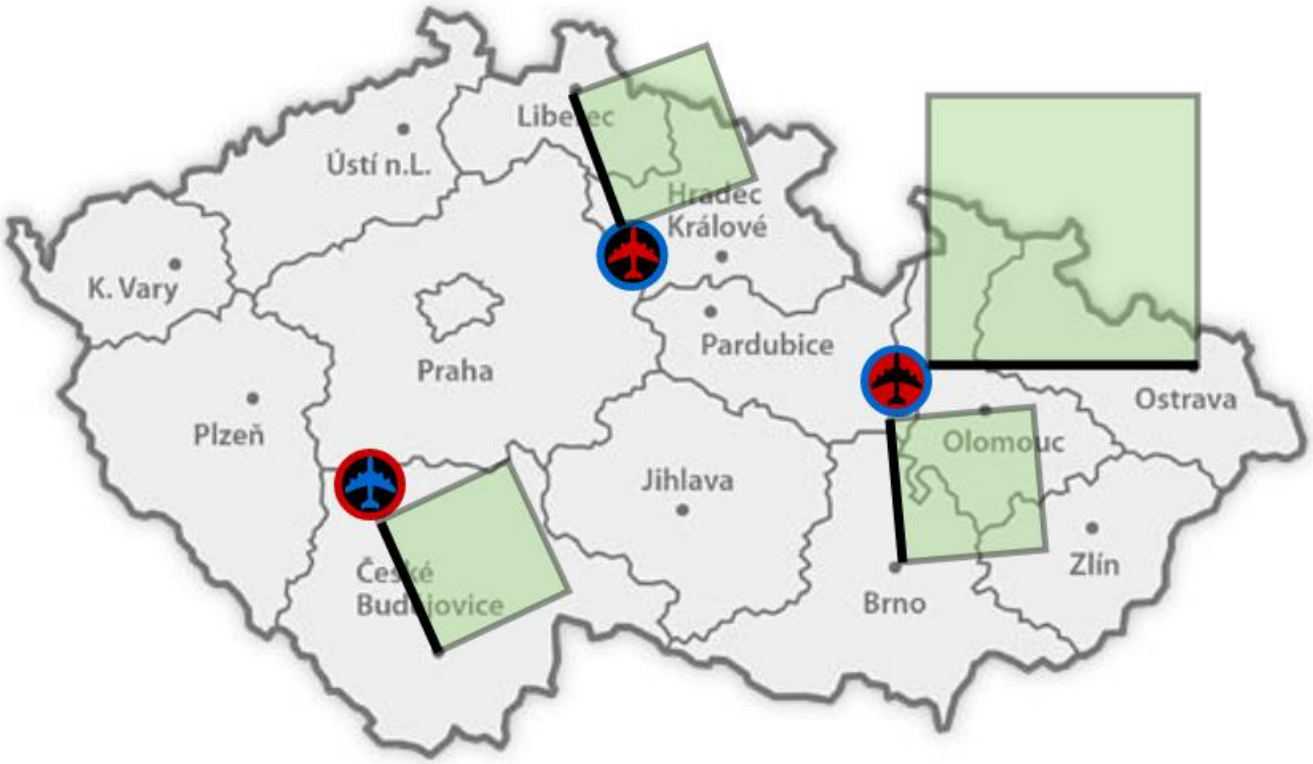
- Uvažujeme **faktorizovaný** stav
- **Fitness funkce** – metrika pro ohodnocení jedinců
- **Selekce/výběr** – kdo se stane rodičem
- **Křížení** (crossover) – process vytvoření potomka z rodičů
- **Mutace** – náhodná změna části potomka (některého atributu)

- Mnoho variant
  - Křížení
  - Mutace
  - Nová populace



Lokální prohledávání ve spojitém prostoru: Příklad

- Hledání shluků (clusters)
- Zvol pozice 3 letišť tak, aby byl součet druhých mocnin euklidovských vzdáleností mezi městi a nejbližším letištěm minimální
  - Minimalizujeme $\sum_{ci} dist2NA(ci)^2$
    - Město ci
    - Vzdálenost $dist2NA(ci)$
    - Vůči nejbližšímu letišti
  - Uvažujeme pevně zvolené přiřazení měst do shluků



1. Diskretizace → aplikování odprezentovaných metod
2. Přímo řešení ve spojitém prostoru
   - Letiště s pozicí (x, y) v mapě → 3 letiště = 6 proměnných $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ → 6-D prostor P, kde $P = (x_1, x_2, x_3, y_1, y_2, y_3)$
   - Pevně zvolené přiřazení měst k letištím (shlukům):
     - $f(P) = \sum_{cl} \sum_{ci \in Cl_{cl}} (x_{cl} - x'_{ci})^2 + (y_{cl} - y'_{ci})^2$
       - Cl je množina shluků, cl konkrétní shluk a ci město
   - Obtížnější pokud bychom neměli zvolené přiřazení měst k letištím

- $\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial y_3}\right)$
- $f(P) = \sum_{cl} \sum_{ci \in Cl_{cl}} (x_{cl} - x'_{ci})^2 + (y_{cl} - y'_{ci})^2$
- $\frac{\partial f}{\partial x_1} = 2 \sum_{ci \in Cl_1} (x_1 - x'_{ci})$

  1. Vyřešit analyticky → $x_1 = \sum_{ci \in Cl_1} \frac{x'_{ci}}{|Cl_1|}$
  2. Iterativně pomocí gradientního vzestupu/sestupu (**gradient ascent/descent**):
     - **Gradient ascent:** $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla \mathrm{f}(\mathbf{x})$
     - **Gradient descent:** $\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla \mathrm{f}(\mathbf{x})$,
     - Koeficient učení α (jak velké kroky děláme)
- Optimalizaci ve spojitých prostorech → časti o strojovém učení!

Lokální prohledávání ve spojitém prostoru: Závěr

- Optimalizace může podléhat nějakým omezením (**constrained optimization**)
  - Umístění letiště nemůže být mimo ČR, v zastavěných plochách, v jezeře, …
  - Pokud jsou omezení lineární, pak lze řešit pomocí **lineárního programování**
- Je možné použít i další metody, které pracují s populací jedinců
  - Např. **optimalizace hejnem částic** (particle swarm optimization PSO)

Děkuji za pozornost!

- https://www.cartoonstock.com/cartoon?searchID=EC345429