Řešení problémů prohledáváním

Martin Šůstek

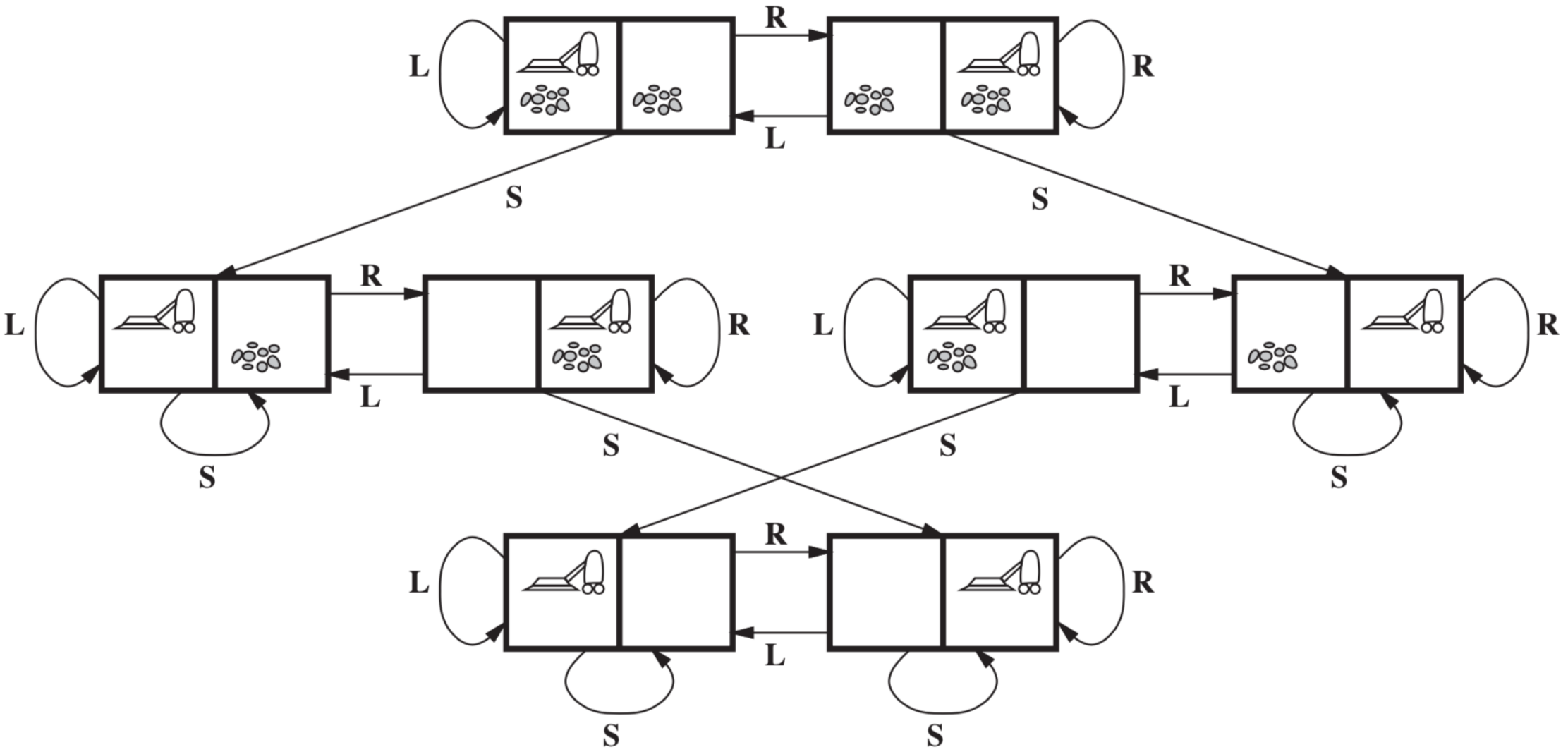
isustek@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

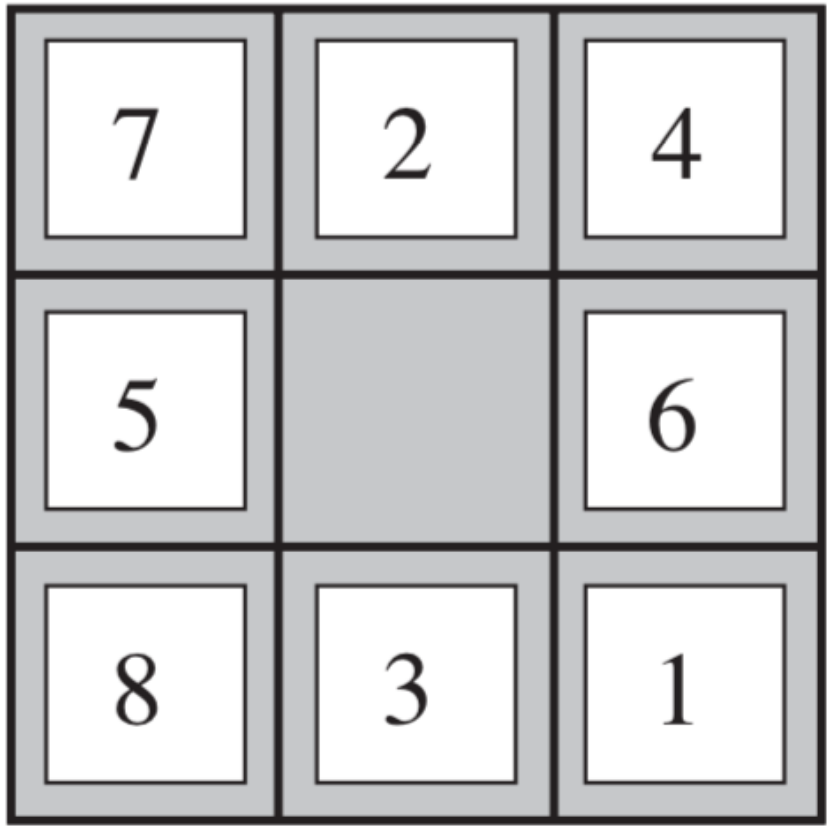
Řešení problémů prohledáváním – Úvod

- Agentní pojetí
  - Cílem (užitkem) řízený agent
  - Známe a jisté prostředí
  - Atomická reprezentace stavů
- Proč? Hledáme řešení úlohy, které chápeme jako **sekvenci akcí**
- Neinformované metody | Informované metody
  - Informovanost je skrze porozumění, jak je určitý stav nadějný (jak daleko od cíle)
- Postup:
  1. Formulace problému (vytvoření validní, ale minimalistické abstrakce)
     - Špatná formulace může zásadně ovlivnit obtížnost úlohy
  2. Spuštění prohledávání → nalezení řešení
  3. Vykonání řešení (nezávisle na vjemech z prostředí)

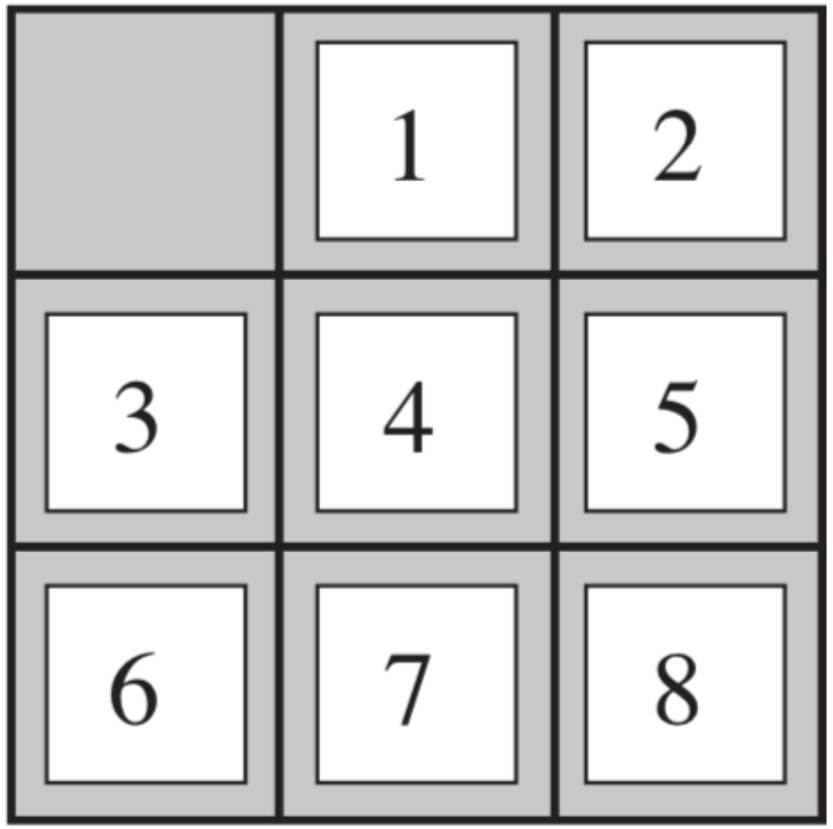
Příklady problémů: Vysavač



Příklady problémů: Loydova osmička

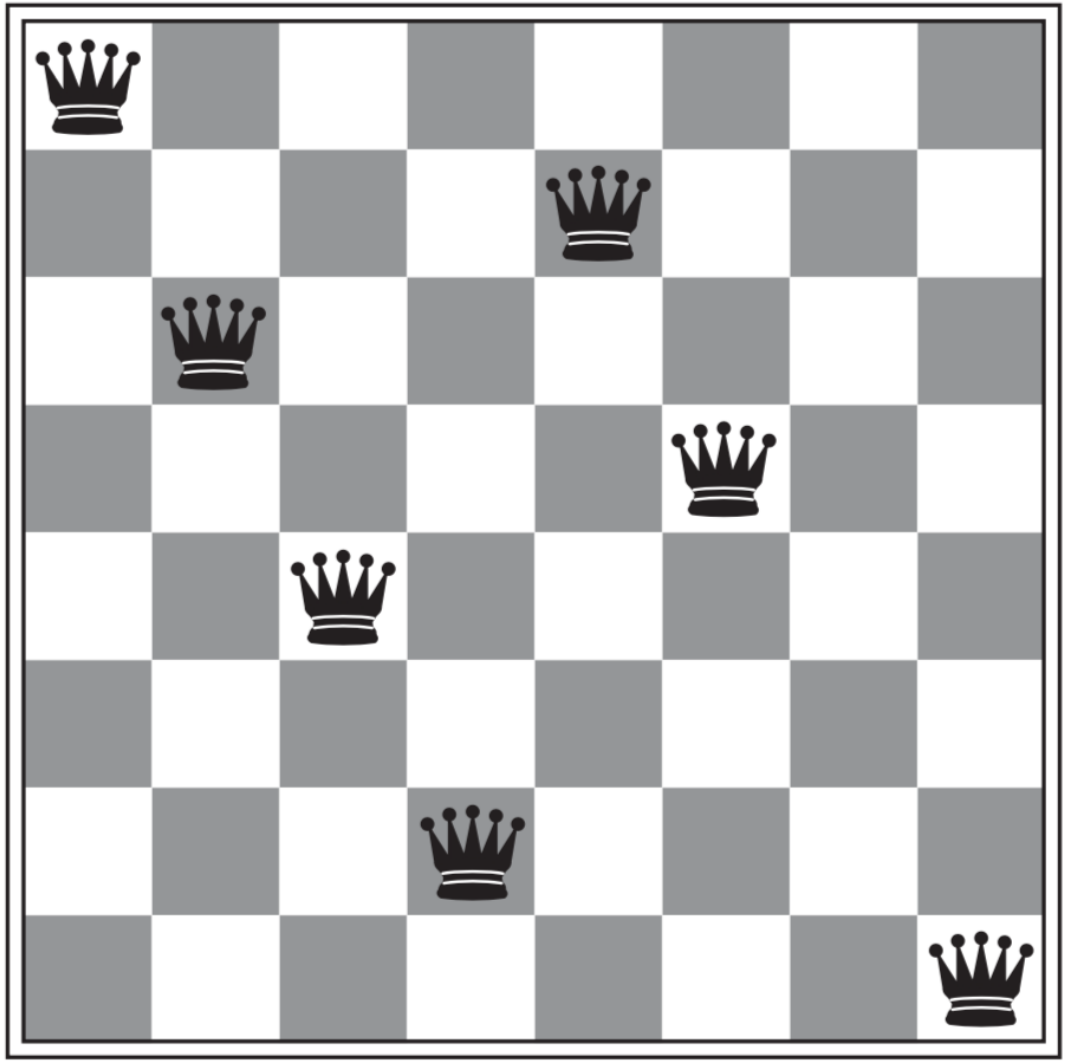


Start State



Goal State

Příklady problémů: 8 dam



Formální definice problému

1. Počáteční stav
2. Akce $ACTIONS(s) = \{a_1, a_2, \ldots, a_n\}$
   - Všechny akce, které lze provést ve stavu s
3. Přechodový model definující $s_{new} = RESULT(s_{old}, a)$,
   - Do jakého stavu s_{new} se dostanu po provedení akce a ve stavu s_{old}
4. Test, je-li stav cílový; $GOAL(s)$
5. Cena cesty (sekvence kroků)
   - Jeden krok je provedení akce a ve stavu s
   - Chápeme jako **sumu cen kroků** $COST(s, a) \rightarrow \mathbb{R}_0^+$
     - **Pouze kladné hodnoty!**

How to solve tasks in AI?

Modeling

Inference

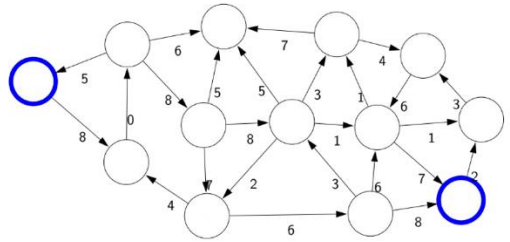
Learning

Zasazení do širšího kontextu: Jak řešit AI úlohy

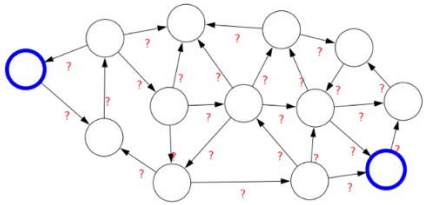
Real world

Modeling

Model



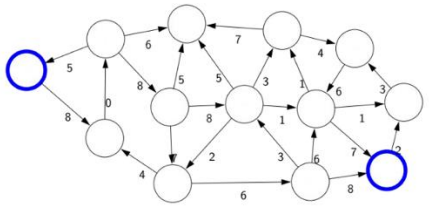
Model without parameters



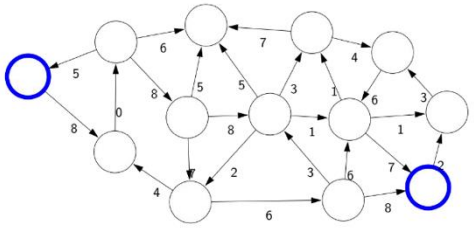
+data

Learning

Model with parameters

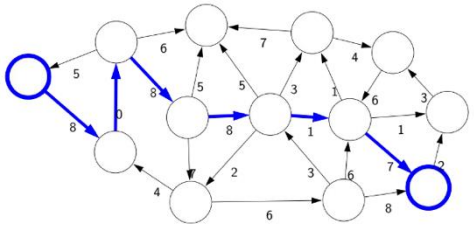


Model



Inference

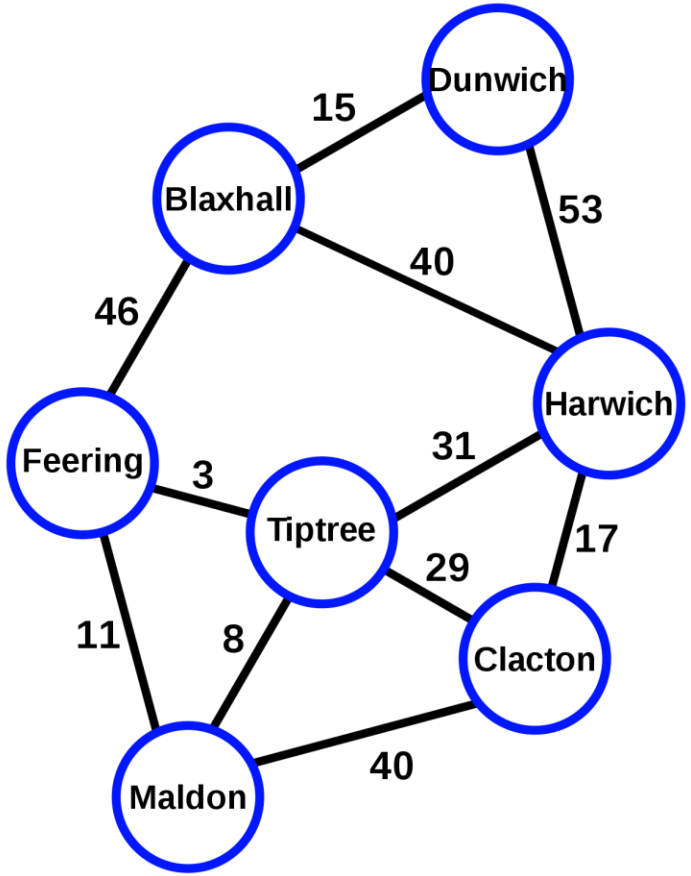
Predictions



Prohledávání s použitím stromu a grafu

- Tree-search | Graph-search
  - Volba abstrakce
- Stav | Uzel
- U graph-search ukládáme prozkoumané **stavy**
  - Využíváme množinu **explored** (seznam closed)
  - Pokud narazíme na již prozkoumaný **stav**, odpovídající **uzel** neuvažujeme
  - Zamezení neustálého opakování stejných kroků

- Akce +11, *3, -15
- Počáteční stav 0
- Cílový stav (například: 550; číslo dělitelné 31)



- **Úplnost**
  - Pokud existuje řešení, algoritmus ho musí nalézt
- **Optimálnost**
  - Nalezené řešení je vždy to s nejlepší cenou cesty
- Časová složitost
- Paměťová složitost

- Reálné problémy:
  - Navíc uvažujeme i cenu prohledávání (výpočet)

Prohledávací strategie a datová struktura

- Uchováváme uzly k prozkoumání (frontier, seznam open)
  - Prohledávací strategie = který uzel vybrat k expandování nejdřív?

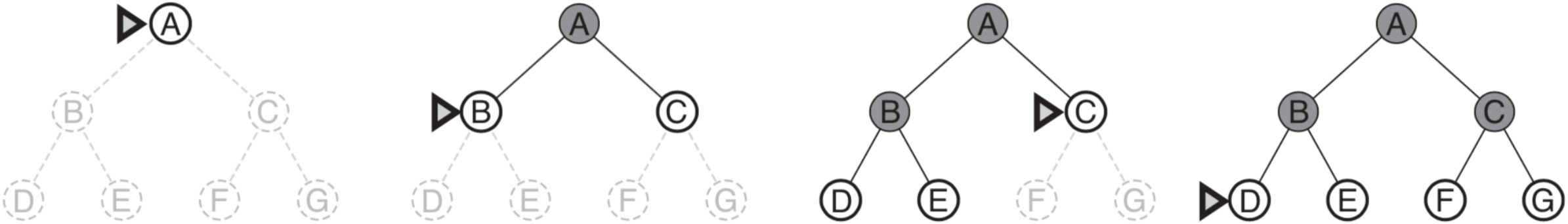
1. Prohledávání do **šířky** – Breadth-first search (**BFS**)
   - FIFO (fronta)
2. Prohledávání do **hloubky** – Depth-first search (**DFS**)
   - LIFO (zásobník)
3. Prohledávání **od nejlepšího** – Best-first search
   - Prioritní fronta

Přehled metod

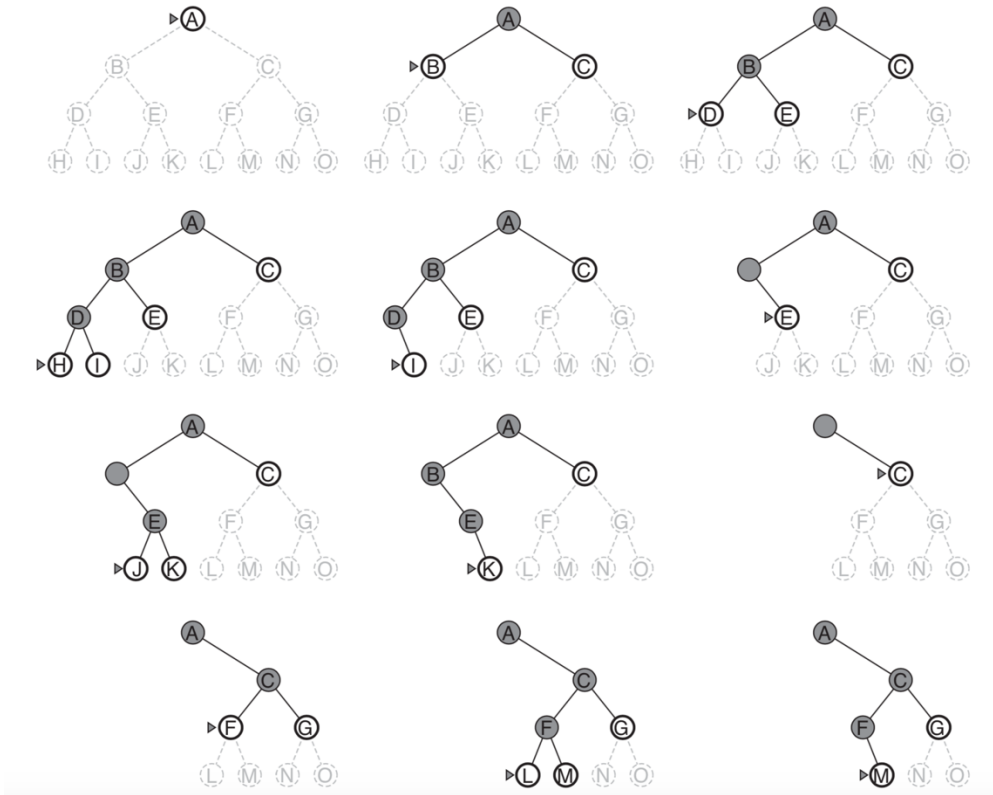
- Neinformované metody
  - BFS (Breadth-first search)
  - DFS (Depth-first search)
  - DLS (Depth-limited search)
  - IDS (Iterative deepening depth-first search)
  - BS (Bidirectional search)
  - UCS (Uniform-cost search)

- Informované metody
  - Greedy best-first search
  - A* search

- Prohledává postupně od nejbližších – vzdálenost je počet uzlů na cestě
- Fronta (FIFO)
- **Optimální** i úplný
- Cílový stav můžeme testovat již při vygenerování
- Extrémní paměťové nároky



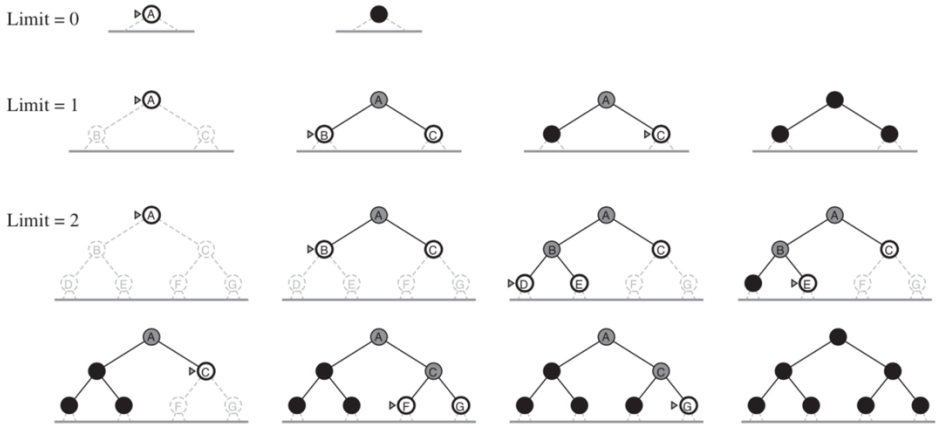
- Řeší problém s pamětí při prohledávání **stromu**
  - V grafu musíme stejně ukládat prozkoumané stavy, takže nám nepomůže
- Zásobník (LIFO)
- **Není optimální**
- **Backtracking search**
  - Varianta DFS, která šetří paměť
  - Neukládají se následníci, ale provedená akce
- Prohledávání stromu je úplné v **konečném** stavovém prostoru, pokud se ověřuje, že nový stav uzlu je na cestě jedinečný

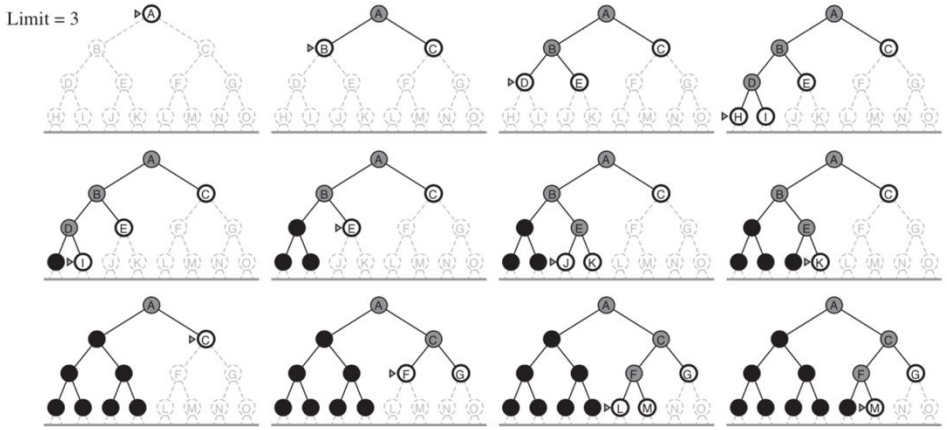


Neinformované metody: Depth-limited search DLS

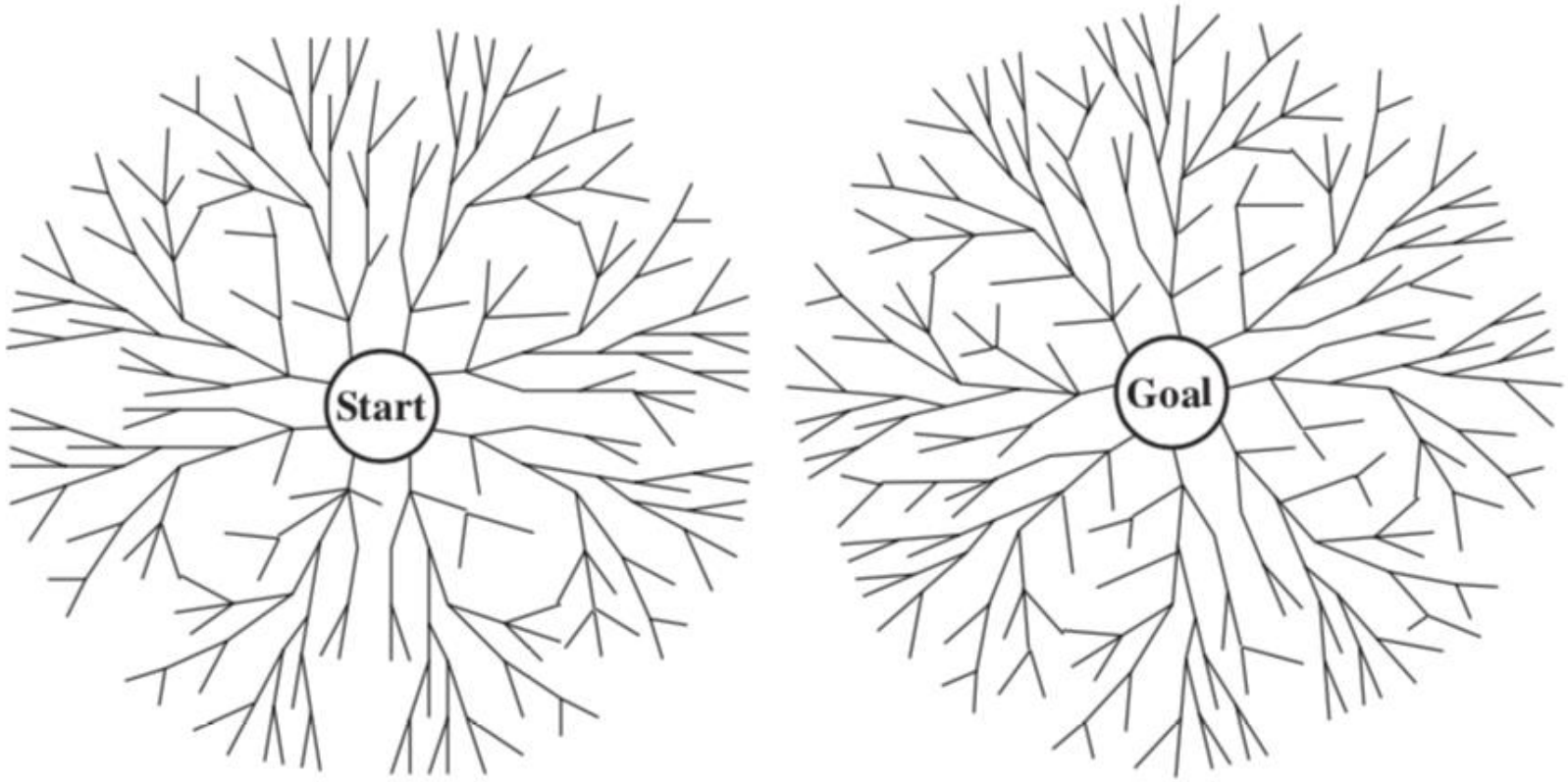
- DFS, kde se omezí hloubka prohledávání
- Úplná metoda, pokud není řešení ve větší hloubce, než je nastavený limit

- Idea: Simuluj BFS pomocí DFS
- Iterativní DLS, začíná se s limitem 0 a v každé iteraci se o 1 zvyšuje
- **Optimální** i **úplná** metoda s malými paměťovými nároky
  - Doporučená metoda při využití neinformovaných metod
  - Pokud víme, že máme dostatek paměti, BFS bude rychlejší

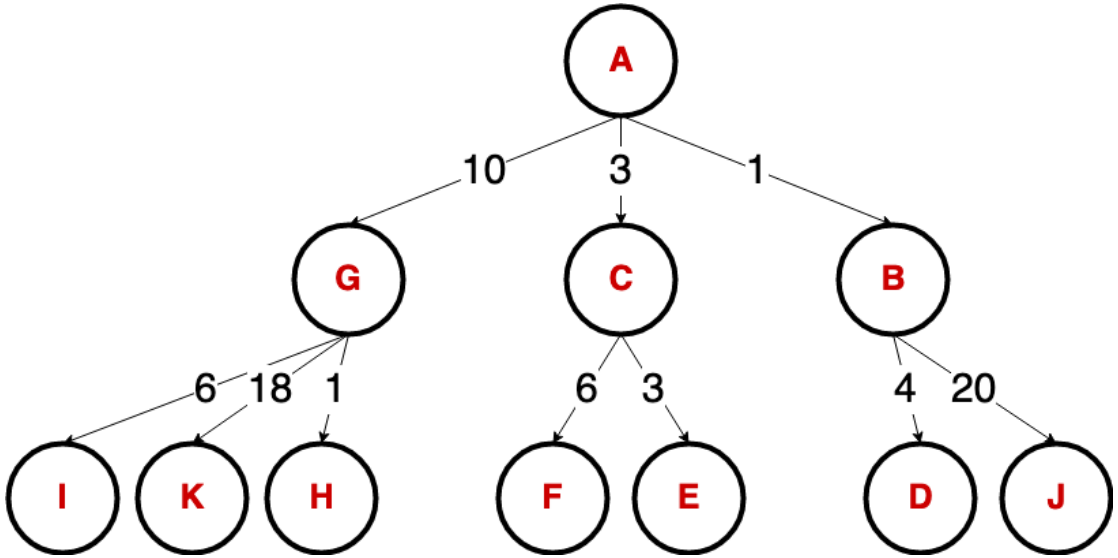




- Obousměrné prohledávání je výhodnější
- Úloha musí mít reverzibilní operátory a jeden cílový stav (nebo je potřeba zvolit)
- Alespoň z jedné strany prohledávání je potřeba uchovávat stavy
  - Velké paměťové nároky
- Už jsme se potkali?
  - Spousta porovnávání



- Co když je každá akce jinak dobrá? → Upravíme BFS!
  - Stačí nevybrat nejbližší uzel (hloubka stromu), ale uzel s **nejmenší cenou cesty**
    - Změna struktury (FIFO → prioritní fronta)
- Cíl se musí testovat až při snaze uzel expandovat, aby byla zajištěna **optimálnost**
  - Může dojít k nahrazení uzlu (stejný stav s nižší cenou) v rámci frontier/open
- Potenciální problém – dálnice z Prahy do Brna

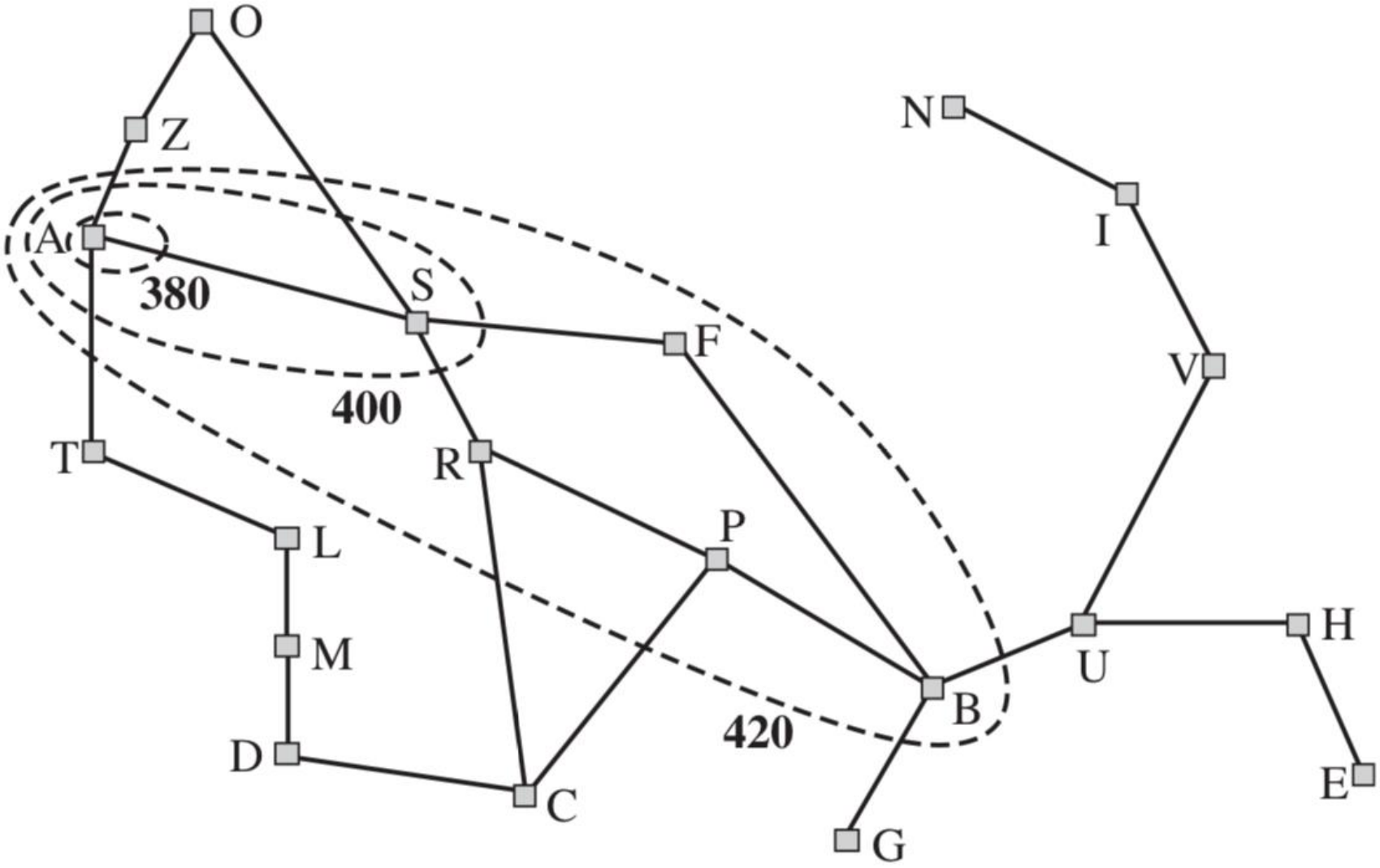


- Implementace metod odpovídá UCS (jediná změna je způsob ohodnocení uzlů)
- UCS využíval pro ohodnocení uzlu n funkci $g(n)$, která odpovídala nejkratší ceně cesty z počátečního uzlu do uzlu n.
- Informované metody se řídí podle funkce $f(n)$, která využívá funkci h(n) a může mít tvar:
- $f(n) = g(n) + h(n)$
- $h(n)$ je heuristická funkce a je to **ODHAD** ceny cesty z uzlu n do nejbližšího cílového uzlu

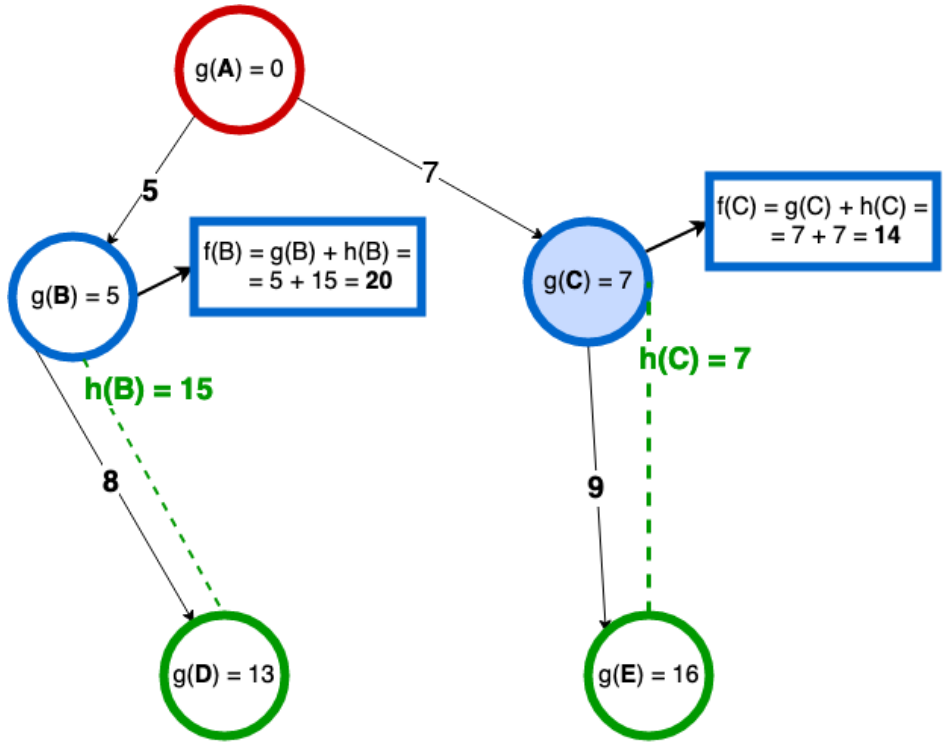
- Metody
  - Greedy best-first search
  - A* search

- Využívá se pouze heuristická funkce, $g(n) = 0$
  - $f(n) = h(n)$
- Není optimální
- Úplnost je obdobná jako pro DFS
- Vhodná metoda, pokud chceme najít nějaké řešení rychle

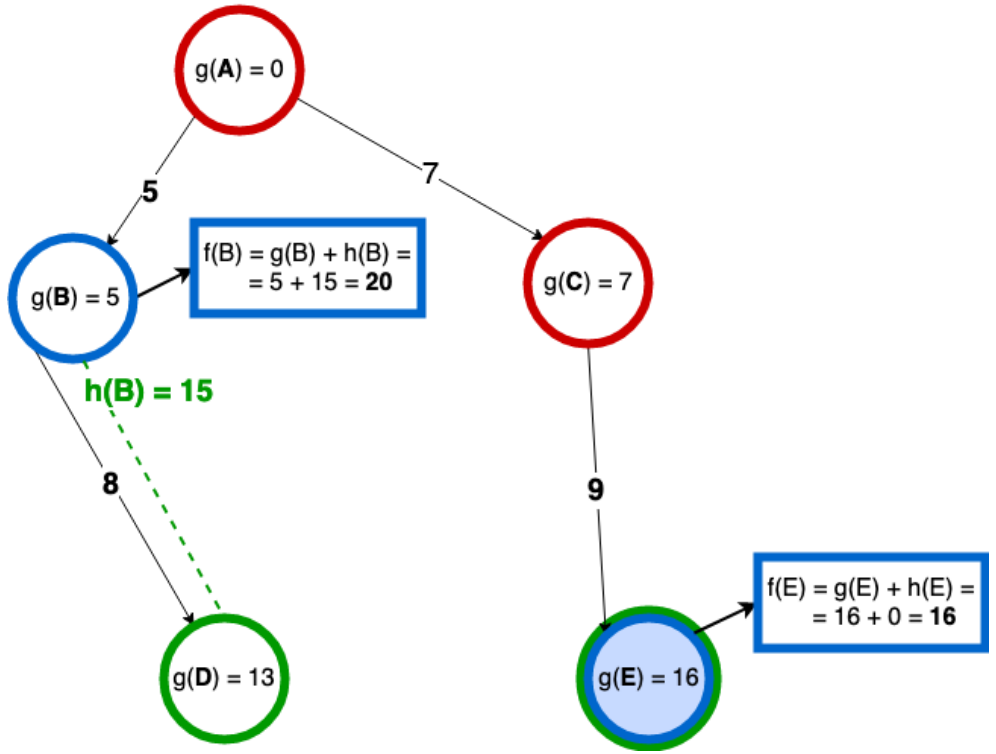
- Využívají se obě složky funkce $f(n)$:
  - $f(n) = g(n) + h(n)$
- **Úplná i optimální**
  - Platí pro tzv. přípustné heuristiky, kdy $h(n)$ není větší než reálná cena do cíle
    - V grafech musí platit i pro každý uzel na cestě do cíle
- Heuristika je v tomto případě chápána jako způsob **prořezávání (prunning)** – vyloučení některých možností bez jejich prozkoumání


O
Z
A
380
S
F
400
T
R
L
P
M
D
C
420
B
G
N
I
V
U
H
E

- Cílové stavy
- Prozkoumané stavy
- Frontier – vybíráme uzel k expanzi (zvolíme nejnižší $f(n)$)



- Jelikož $h(B)$ je větší než cena přechodu z B do D, uzel E se expanduje před uzlem B a je výsledkem celého prohledávání, přestože není optimální



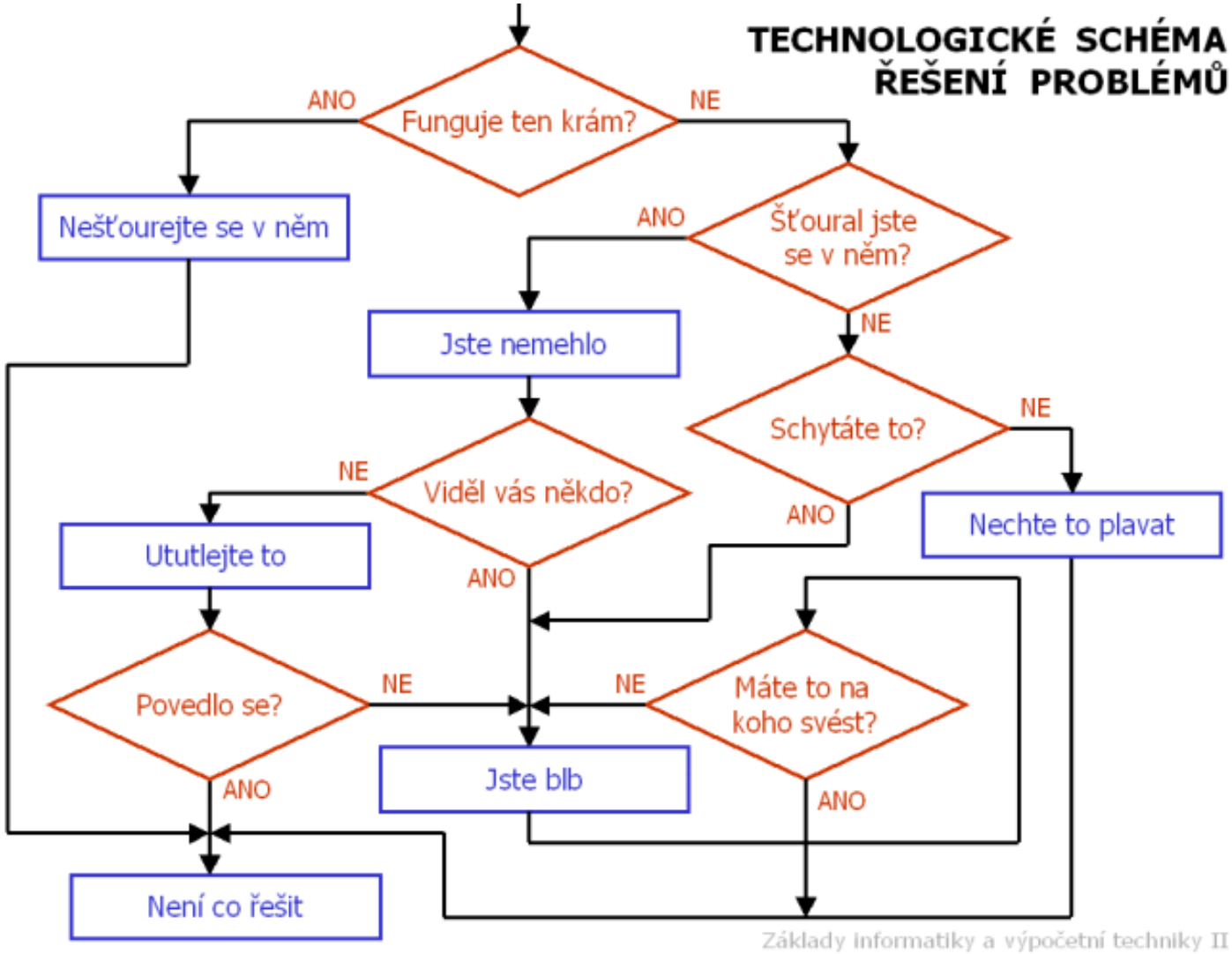
Informované metody: A* – možná vylepšení

- V reálných aplikacích může mít A* taktéž problémy s pamětí → různé úpravy:
  - IDA*
  - RBFS
  - MA*
  - SMA*

- Heuristická funkce $h(n)$ musí být menší než reálná cena cesty do cíle (pokud chceme mít zaručenou optimálnost), ale na druhou stranu, pokud si můžeme vybrat mezi dvěma heuristikami h_1 a h_2, $\forall n: h_1(n) \geq h_2(n)$, pak použijeme h_1, protože má **větší hodnoty**, a tím pádem bude nutné prozkoumat méně *slepých cest*

- Relaxed problem = ignorování některých omezení
  - Musí být lehce spočitatelné
- Co když máme K heuristik, ale neplatí, že jedna je lepší než všechny ostatní:
  - $\exists i \forall k \forall n: h_i(n) \geq h_k(n), 0 \leq i < K, 0 \leq k < K$
  - Vytvoříme novou heuristiku $h_K(n) = \max(h_0, \ldots, h_{K-1})$
- Také se můžeme „učit“ heuristickou funkci pomocí např. neuronových sítí
  - Těžké vynutit, aby byla přípustná → nezaručíme optimálnost

Děkuji za pozornost!



- https://www.komik.cz/dir_obrazky/s/0_schema.gif