Alternativní přístupy k plánování

Otakar Trunda

Seminář z umělé inteligence

Plánování

- Vstup:
  - *Satisficing task*: počáteční stav, cílové stavy, přípustné akce
  - *Optimization task*: počáteční stav, cílové stavy, přípustné akce, ceny akcí
- Výstup:
  - *Satisficing task*: posloupnost akcí, která vede ke splnění cíle
  - *Optimization task*: posloupnost akcí, která vede ke splnění cíle a minimalizuje cenu plánu (součet cen akcí)
- Klasické plánování:
  - Deterministický, plně pozorovatelný, statický svět
  - Akce jsou okamžité
  - Techniky jsou doménově nezávislé

Plánování

- Vstup:
  - *Satisficing task*: počáteční stav, cílové stavy, přípustné akce
  - ***Optimization task*: počáteční stav, cílové stavy, přípustné akce, ceny akcí**
- Výstup:
  - *Satisficing task*: posloupnost akcí, která vede ke splnění cíle
  - ***Optimization task*: posloupnost akcí, která vede ke splnění cíle a minimalizuje cenu plánu (součet cen akcí)**
- Klasické plánování:
  - Deterministický, plně pozorovatelný, statický svět
  - Akce jsou okamžité
  - Techniky jsou doménově nezávislé

Plánovací problém formálně

- Stavy zadané jako **hodnoty stavových proměnných:**
  - Množina stavových proměnných $V = \{v_1, v_2, \ldots, v_n\}$
  - Každá $v_i \in V$ má obor hodnot R_i
  - Množina stavů $S = \{s \mid s\colon V \mapsto \bigcup_i R_i \wedge s(v_i) \in R_i\}$
- Přechody mezi stavy zadané jako **povolené změny hodnot stavových proměnných:**
  - Množina operátorů O, pro každý operátor $op \in O$:
    - $prec(op) : V \to \bigcup_i R_i \wedge prec(op)(v_i) \in R_i$
    - $eff(op) : V \to \bigcup_i R_i \wedge eff(op)(v_i) \in R_i$

SAS+ formát

- Počáteční stav:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

SAS+ formát

- Akce:
  - (A3 = 7, B1 = 4) => (A3 = 1)
  - (D3 = 2, D1 = 6) => (C5 = 1, B2 = 1)
  - () => (C1 = 1)
  - …
- Cíl:
  - D6 = 1

SAS+ formát

- Současný stav:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

- Akce: ($B_4 = 1$, $C_2 = 7$) => $C_5 = 4$

SAS+ formát

- Současný stav:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

- Akce: ($B_4 = 1$, $C_2 = 7$) => $C_5 = 4$

SAS+ formát

- Nový stav:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	4	1
D	25	2	1	3	5	0

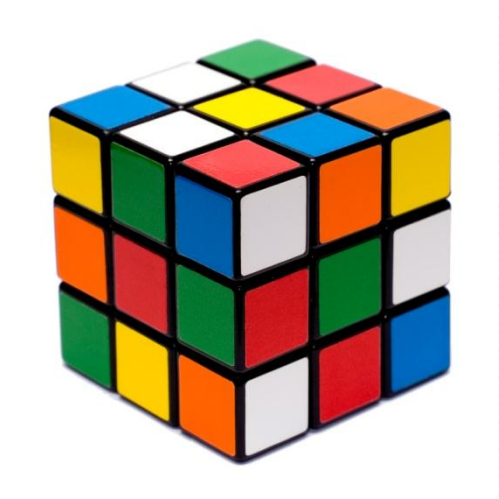
- Akce: ($B_4 = 1$, $C_2 = 7$) => $C_5 = 4$

Příklady plánovacích problémů

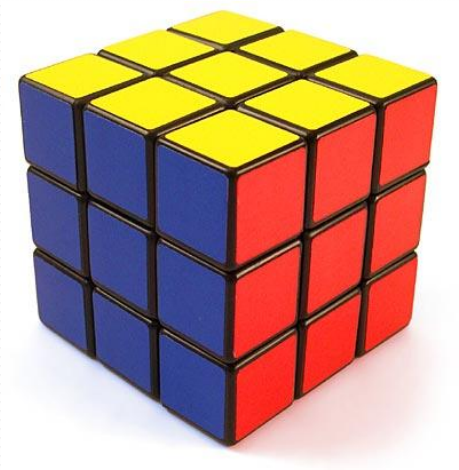
Příklady plánovacích problémů

- Rubikova kostka

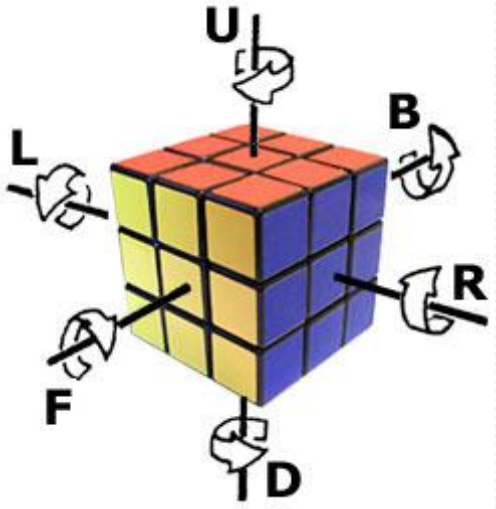
Počáteční stav:



Cílový stav:

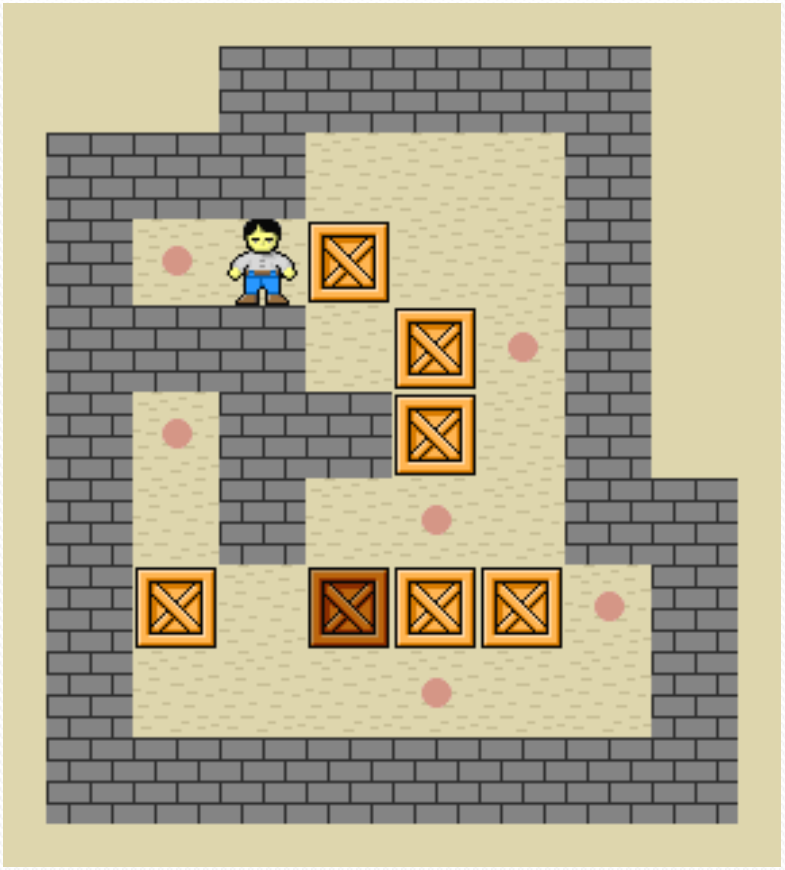


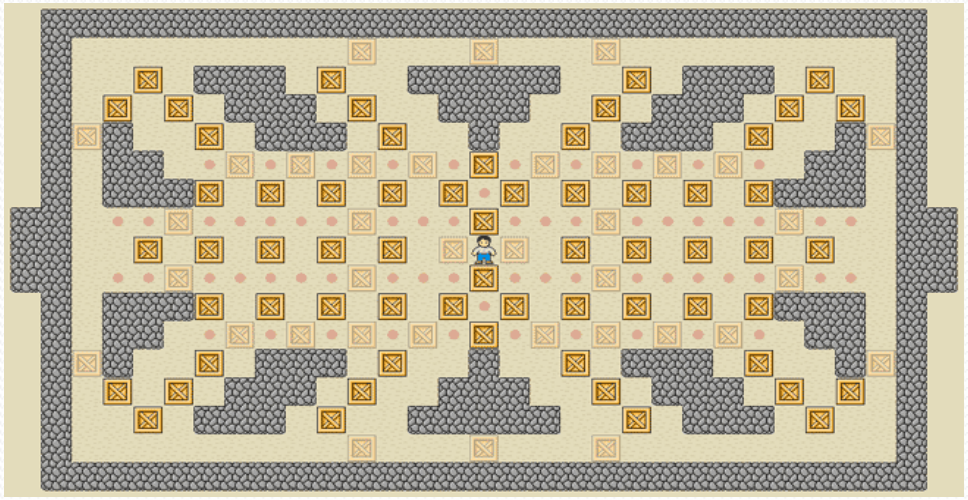
Přípustné akce:



Příklady plánovacích problémů

- Sokoban



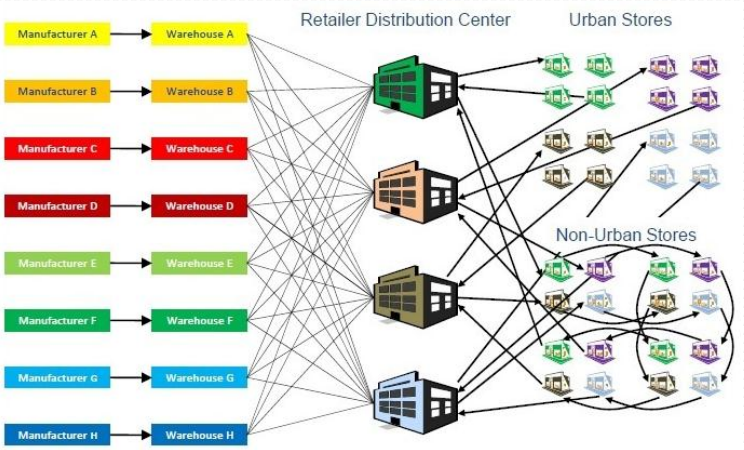


Příklady plánovacích problémů

- FreeCell

Příklady plánovacích problémů

- Logistika



Příklady plánovacích problémů

- Plánování výroby

Populární plánovací algoritmy

- Informované dopředné prohledávání
  - *A** s heuristikou
  - Hill-climbing s heuristikou
- Heuristická funkce – odhad vzdálenosti do cíle

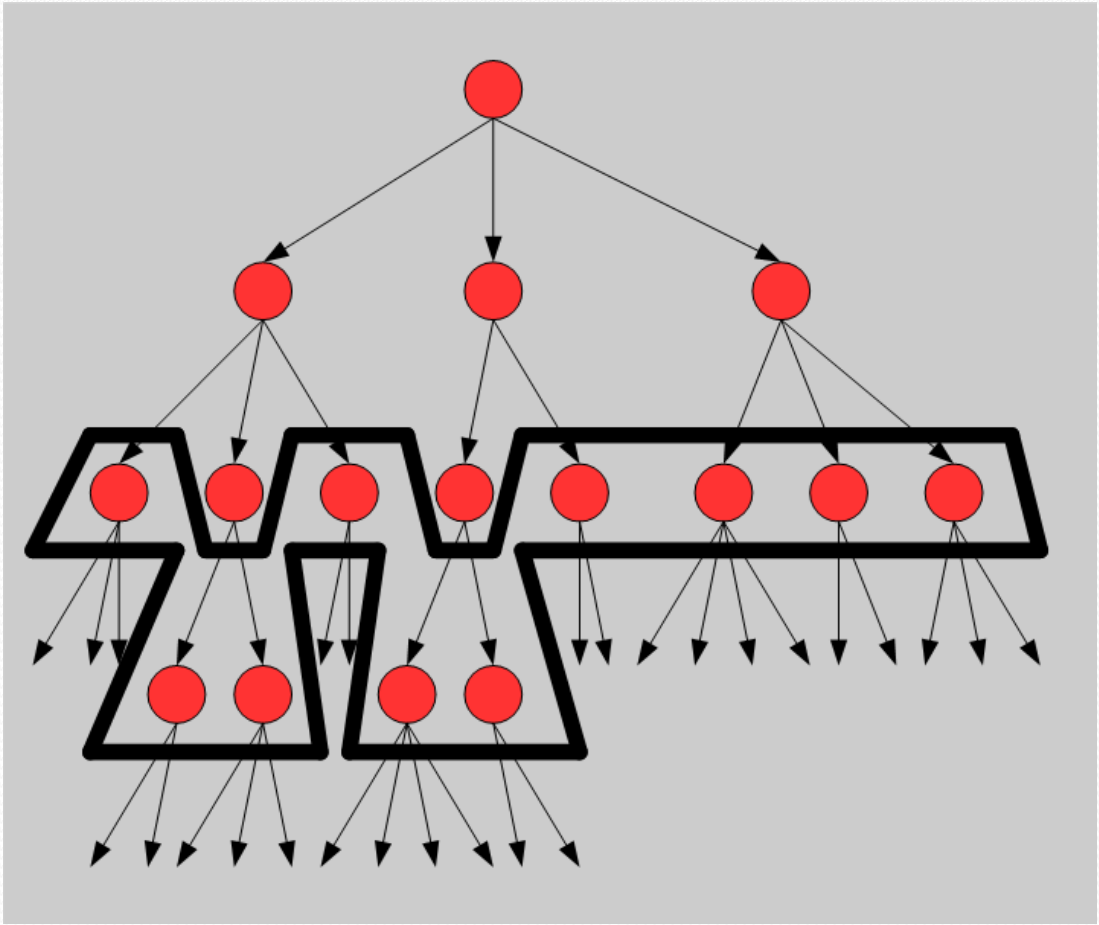
Použité značení

- s – počáteční stav
- n – vrchol stavového prostoru
- $g(n)$ – délka nejkratší cesty z s do n
  - Je známá během prohledávání
- $h^*(n)$ – délka nejkratší cesty z n do cíle
  - Není známá během prohledávání
- $h(n)$ – heuristický odhad vzdálenosti do cíle
- $f(n)$ – ohodnocení stavu n

Algoritmus A*

- Udržuje prioritní frontu stavů
- Expanduje stav s nejnižším ohodnocením
- Ohodnocení stavu
  - $f(n) = g(n) + h(n)$
- Pokud je použitá heuristika přípustná, vždy najde optimální řešení

Algoritmus A*



Vlastnosti heuristik

- Heuristika je **přípustná** pokud $h(n) \leq h^*(n)$
- Pokud $h(n) = h^*(n)$, pak vyřešíme problém v lineárním čase
- Polynomiální čas lze garantovat pokud
  - $h^*(n) - h(n) \in \mathcal{O}(\log(h^*(n)))$
- h_1 **dominuje** h_2 pokud $\forall n\ h^*(n) \geq h_1(n) \geq h_2(n)$
- Pokud h_1 dominuje h_2, pak A* při použití h_1 expanduje méně uzlů než při použití h_2
- Čím větší je hodnota $h(n)$ tím lépe (za předpokladu že je přípustná)

Heuristiky v prohledávání

- Doménově závislé heuristiky
  - Např. pro Lloydovu patnáctku:
  - h_1 = počet špatně umístěných kostiček
  - h_2 = součet vzdáleností kostiček od jejich cílů
- **Doménově nezávislé heuristiky**
  - Relaxace problému
    - Odhad hodnoty řešení původního problému pomocí hodnoty řešení zjednodušeného problému
    - Ignorování některých podmínek
    - Abstrakce, Databáze vzorů

Heuristiky v plánování

- Používané techniky:
  - Delete relaxace
  - Kauzální graf
  - Kritické cesty
  - Landmarky
  - Merge-and-shrink abstrakce
  - Databáze vzorů
  - Počítání kolikrát minimálně musím použít danou akci
  - Mnoho dalších..

Plánovací domény

- Umělé („*toy problems*“)
  - Mohou obsahovat neodhalené slepé uličky a „pasti“
  - Často je **těžké najít jakýkoliv plán**
  - Sokoban
- Praktické
  - Zahrnují optimalizaci
  - Většinou je **snadné najít suboptimální plán**
  - Optimalizace je hlavní problém
  - Ohodnocení plánu může zahrnovat rozvrhování
  - TSP s přidanými podmínkami

Optimalizační algoritmy

- Standardní metaheuristiky:
  - Monte-Carlo Tree Search
  - Genetické and evoluční algoritmy
  - Particle Swarm Optimization, Ant Colony Optimization, Estimation of distribution algorithm, …
  - „Nové“ algoritmy: Cuckoo search, Firefly algorithm, …
- Nestandardní techniky:
  - Hyper-heuristiky, Algorithm selection, Parameter tuning
  - Machine learning

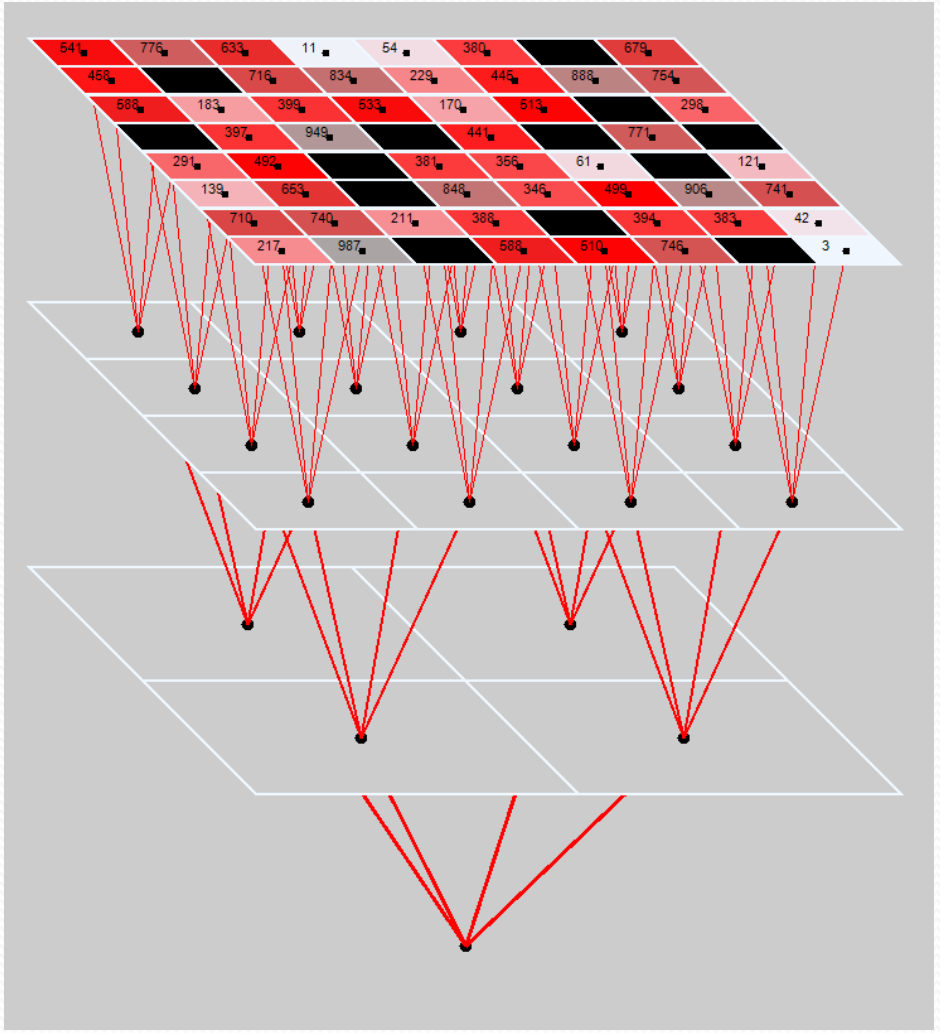
Optimalizační algoritmy v plánování

- Algoritmus pracuje na množině všech plánů $\underline{P}$ a minimalizuje účelovou funkci f
- Výhody:
  - Robustnost – nejsou potřeba informace o struktuře problému
  - Časová a paměťová efektivita (ve srovnání s A*)
  - Použitelné i pro vícekriteriární optimalizaci
- Nevýhody:
  - Je třeba najít přípustná kandidátní řešení (plány)
  - Efektivita silně závisí na tvaru účelové funkce
  - Negarantují optimalitu

Optimalizační algoritmy v plánování

- „Top-down“ přístup
- Algoritmus pracuje na množině všech plánů $\underline{P}$ a minimalizuje účelovou funkci f
- Předpoklady:
  - Prvky $\underline{P}$ je možné **jednoduše generovat**
  - Množina $\underline{P}$ je „**kompaktní**“
  - f má vysokou **lokalitu** (je blízká spojité funkci)

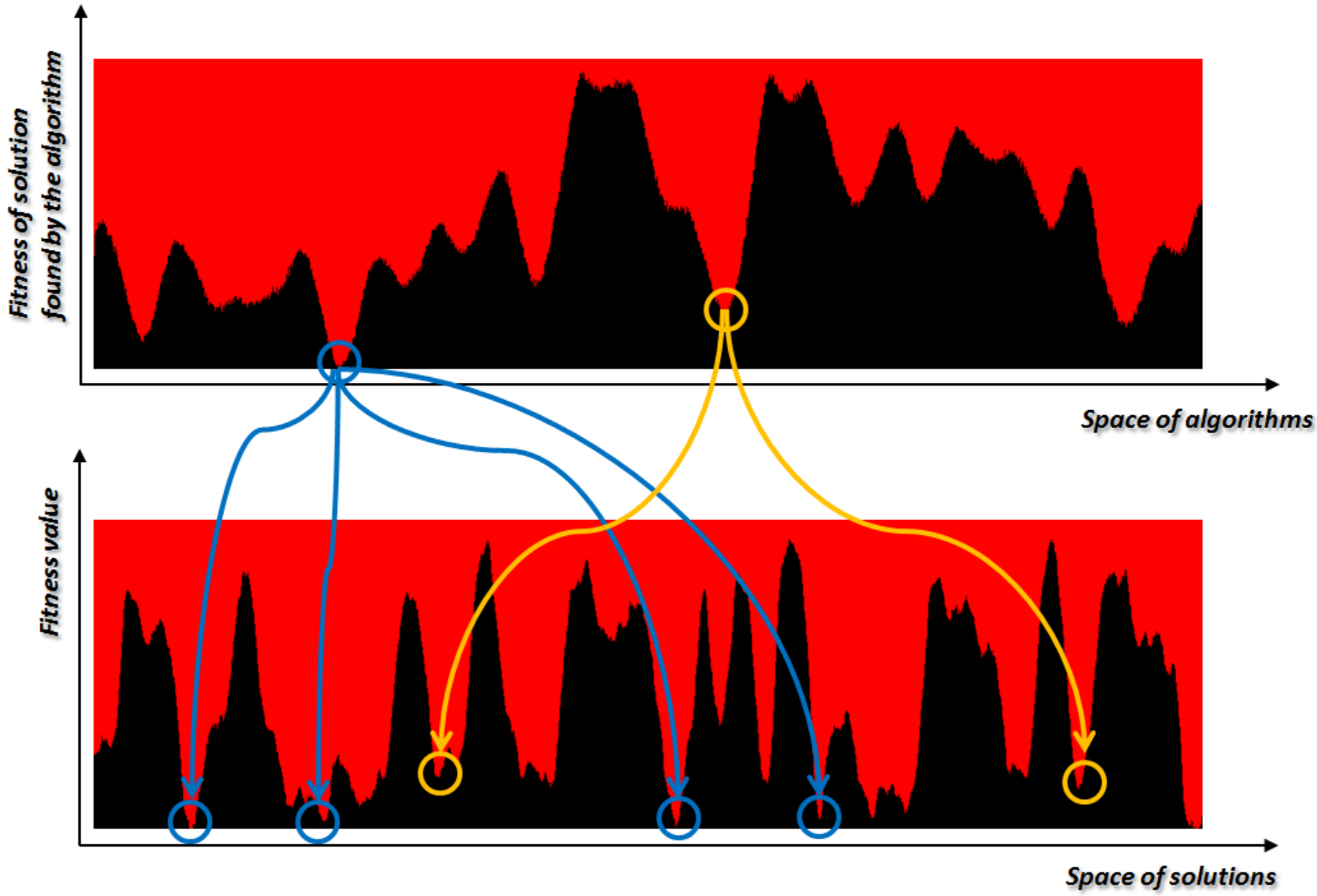
Optimalizační algoritmy v plánování



Překonání překážek

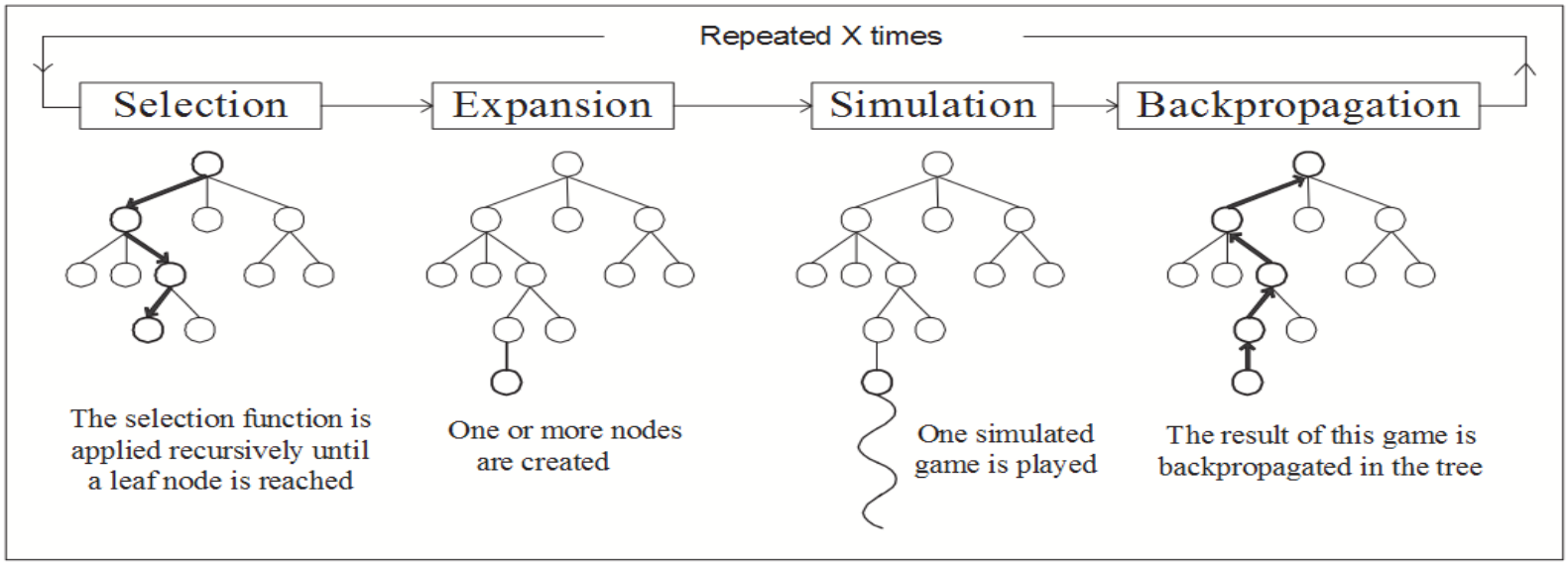
- Použití technik pouze na vhodné domény!
- Předzpracování problému (např. nalezení meta-akcí)
- Penalizace nepřípustných řešení pomocí heuristiky
  - Umožňuje využít mocné nástroje klasického plánování
- Využití reprezentací zachovávajících lokalitu, náhradních modelů nebo hyper-heuristik
  - Například posloupnosti hladových kroků podle různých heuristik

Hyper-heuristiky

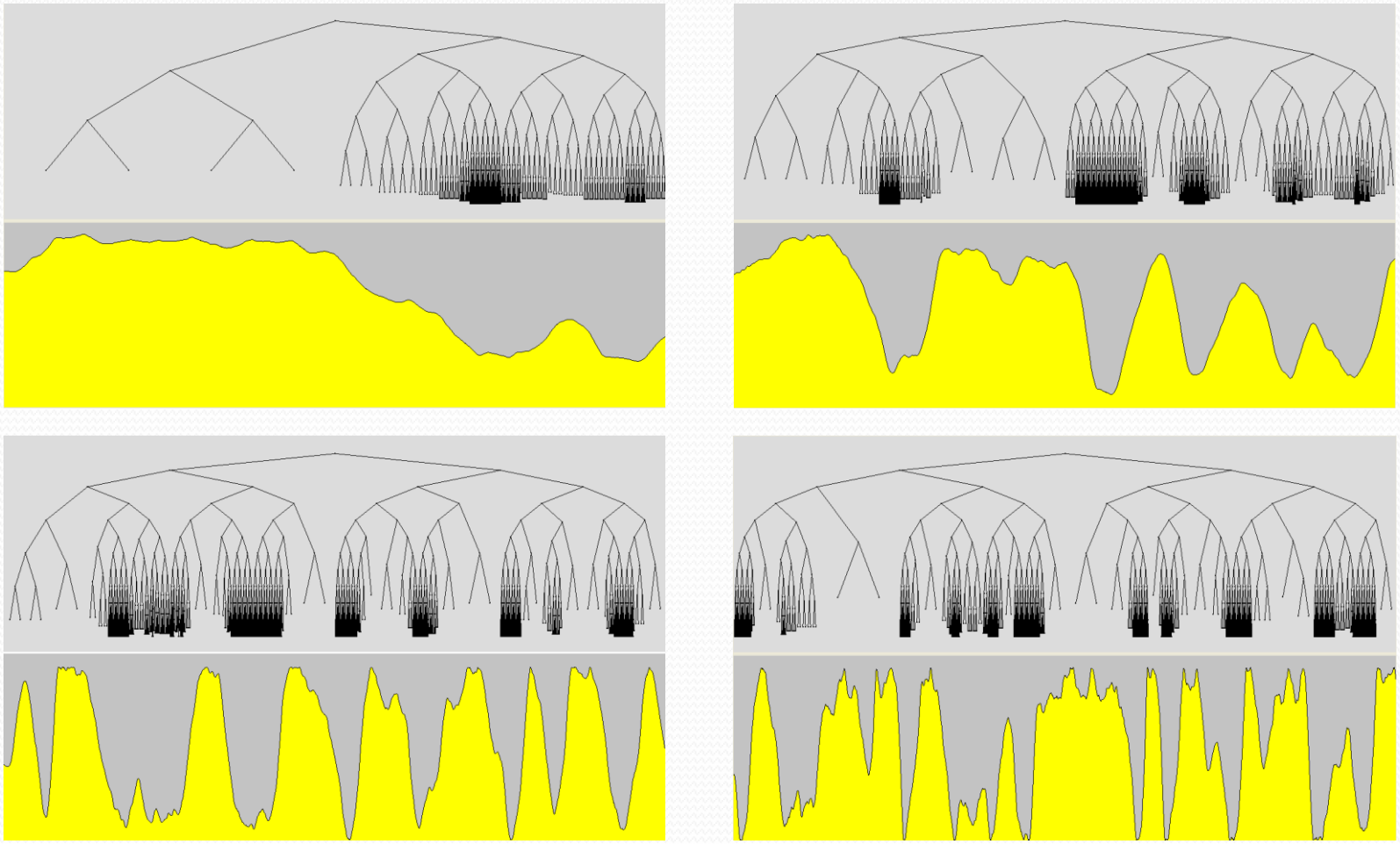


Algoritmus MCTS

- Anytime stochastický optimalizační algoritmus
- Stromové prohledávání
- Náhodné vzorkování namísto ohodnocovací funkce
- Postupně staví asymetrický strom
- Skvělé výsledky v oblasti hraní her (Go, Hex, SameGame)



MCTS – tvar stromu



MCTS v plánování

- Možné problémy ve fázi simulace
  - Žádné omezení na délku simulace
  - „Zacyklení“ v průběhu simulace
  - Slepé uličky a komponenty

Doménová (ne)závislost?

- Všechny problémy je možné řešit stejným algoritmem
- Doménově závislé techniky jsou vždy efektivnější
- Jak formálně popsat doménově závislou informaci?

- *Algorithm selection, hyper heuristics, parameter tuning, portfolios → Autonomous search*

Pomocná informace

- t – plánovací problém
- S – množina všech sekvencí akcí
- $P \subseteq S$ – množina všech plánů
- $f : P \mapsto \mathbb{R}$ – účelová funkce
- *time(t,h)* – kolik času vyžaduje řešení $\underline{t}$ *s pomocnou informací* $\underline{h}$
- H – množina přípustných hodnot pro $\underline{h}$
- *find(h)* – čas pro nalezení správného $\underline{h}$ v $\underline{H}$

- Příklad: automatická tvorba databází vzorů

Autonomní prohledávání

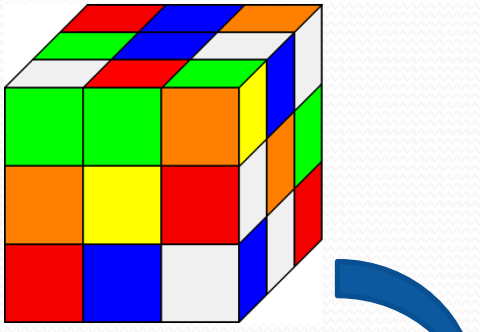
- Efektivní předzpracování:
  - *find(h) + time(t,h) < time(t, NoHelp)*
- *Autonomní prohledávání* kombinuje hledání *h* s hledáním řešení (s využitím *h*)
- Parameter tuning, parameter control

Databáze vzorů

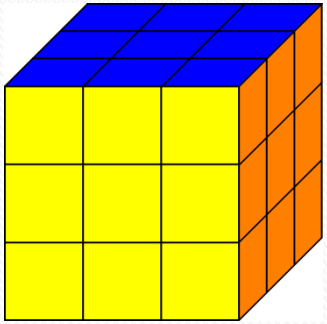
- Vzor S = množna proměnných
  - $S \subseteq Var$
- Plánovací problém P_S indukovaný vzorem S
  - Proměnné: S
  - Počáteční a cílový stav: stejné jako původní, omezené na množinu S
  - Akce: stejné jako původní, z předpokladů a efektů odstraníme všechny proměnné mimo S
- P_S vyřešíme hrubou silou pro všechny stavy
  - Najdeme skutečnou vzdálenost do cíle v P_S
  - Pro všechny stavy – pomocí zpětného prohledávání
  - Hodnoty uložíme do databáze

PDB na příkladu: Rubikova kostka

- Jak získat odhad vzdálenosti?

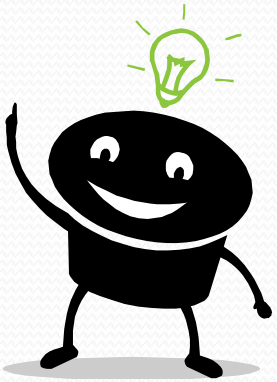


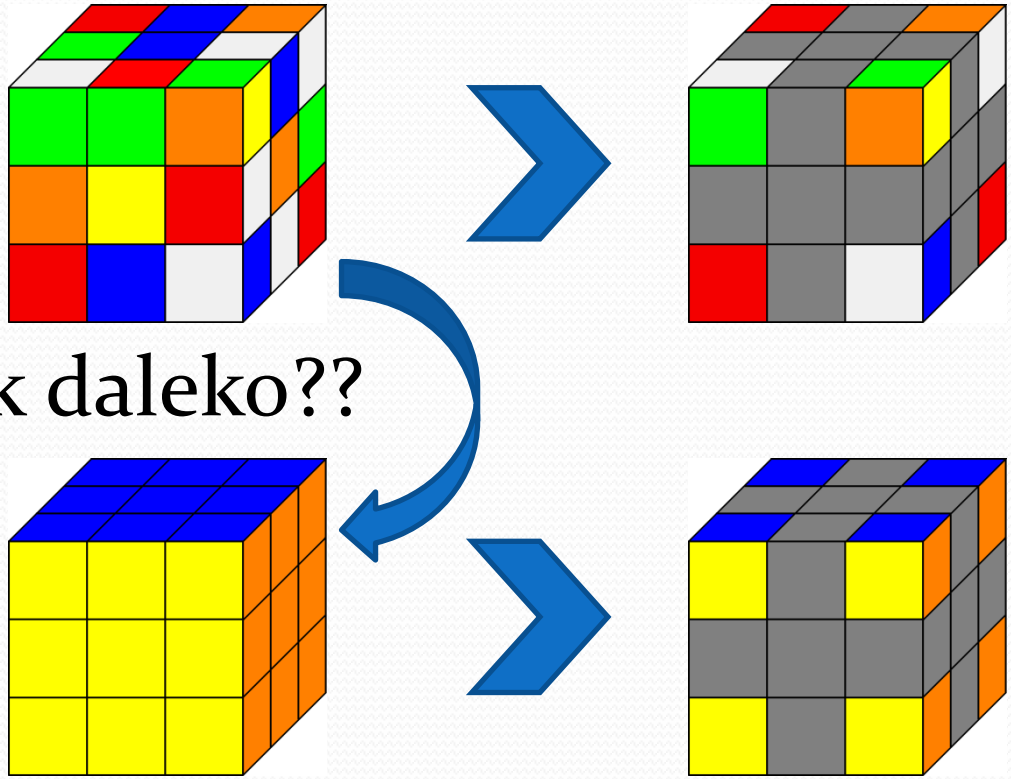
Jak daleko??



PDB na příkladu: Rubikova kostka

- Udělejme abstrakci…

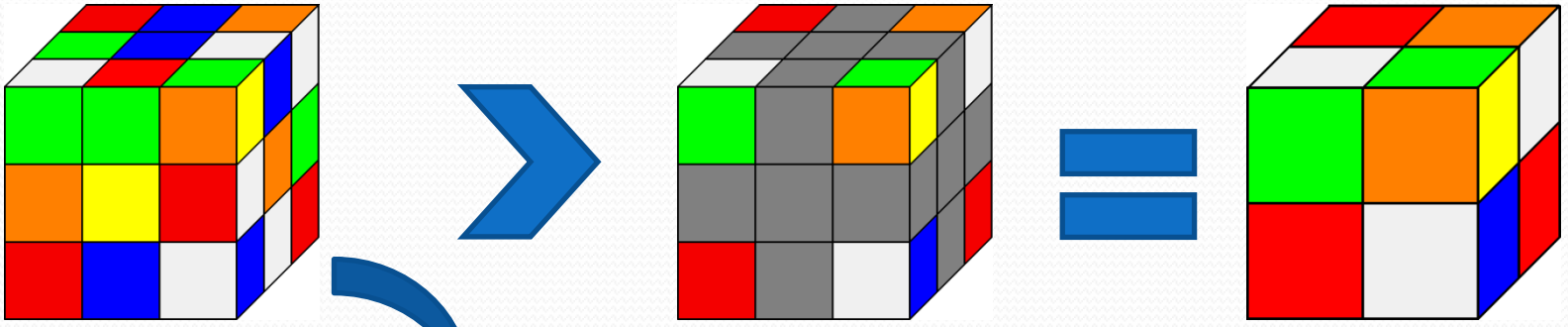




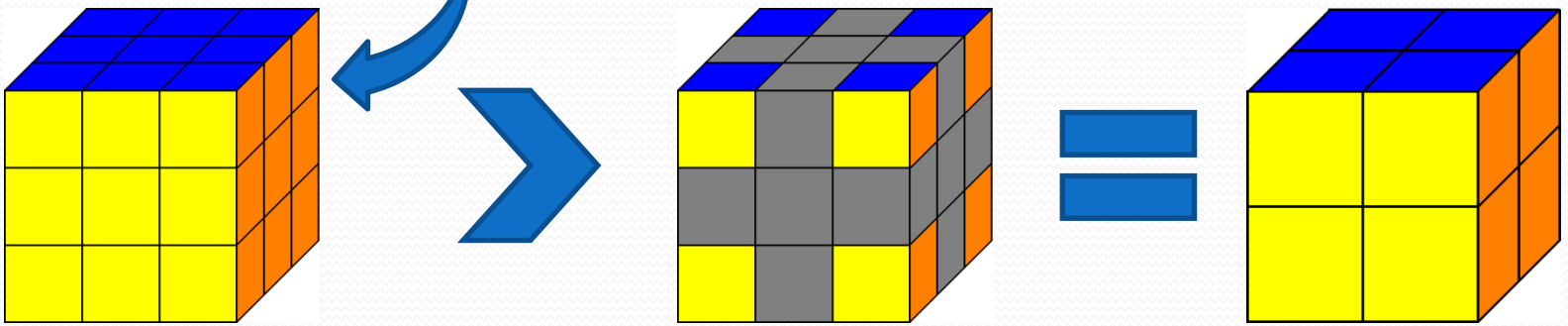
Jak daleko??

PDB na příkladu: Rubikova kostka

- … abychom dostali menší problém

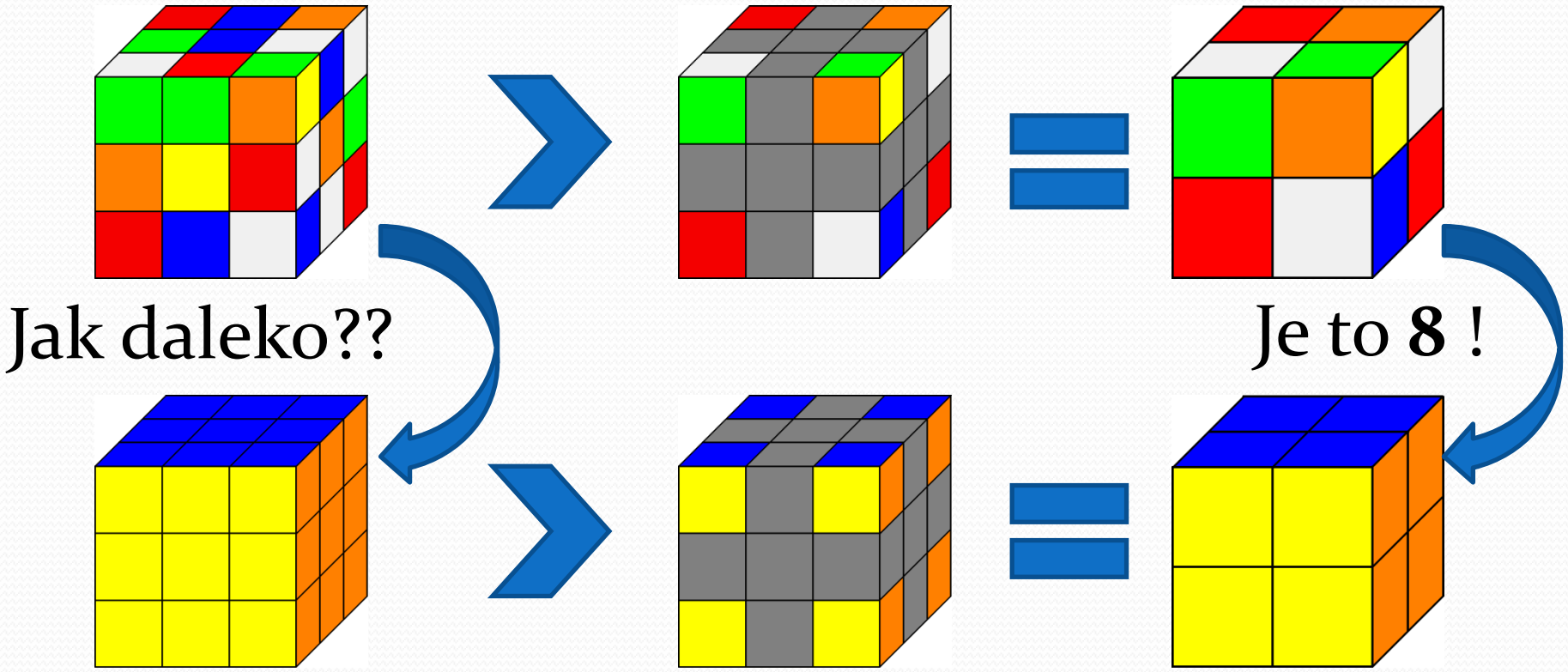


Jak daleko??



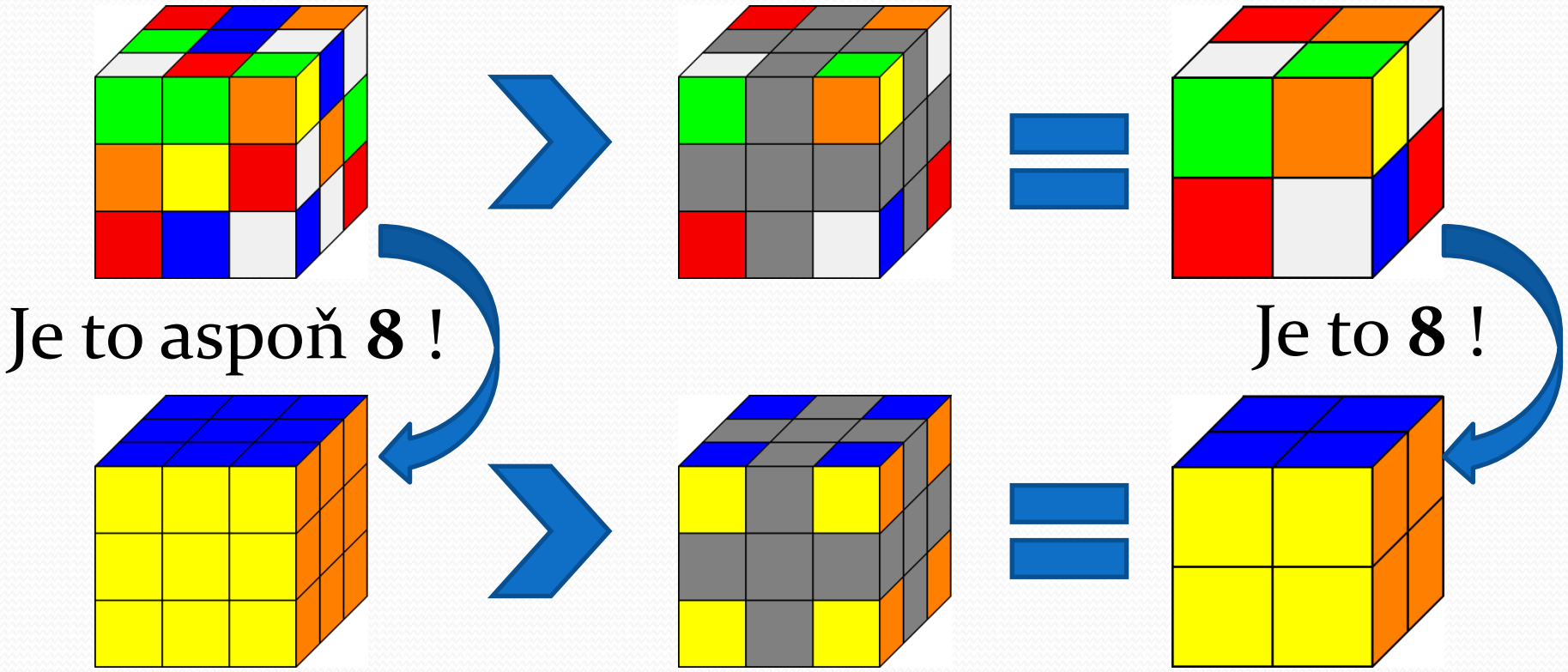
PDB na příkladu: Rubikova kostka

- Vyřešme menší problém

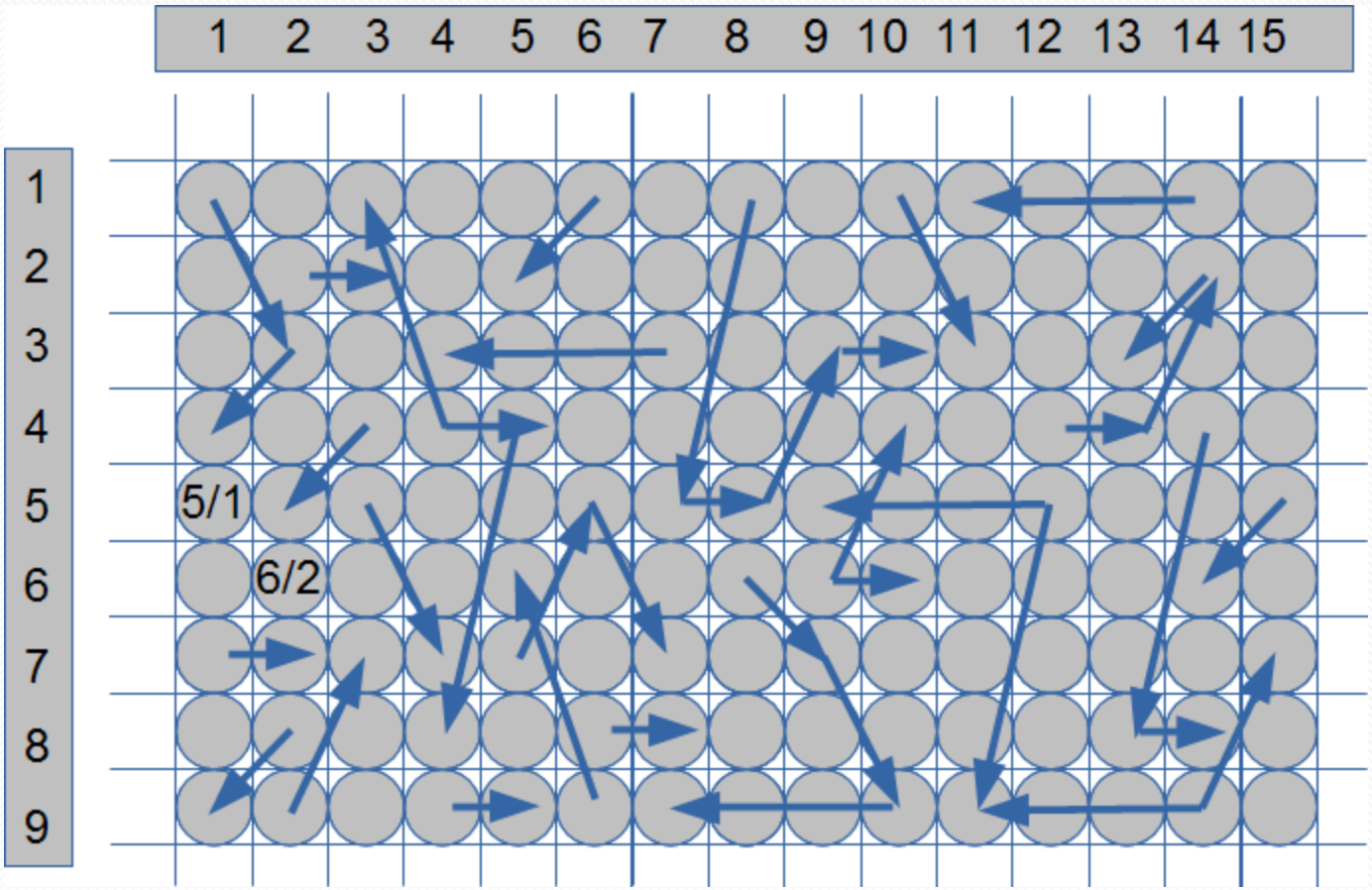


PDB na příkladu: Rubikova kostka

- Použijeme toto řešení jako odhad:

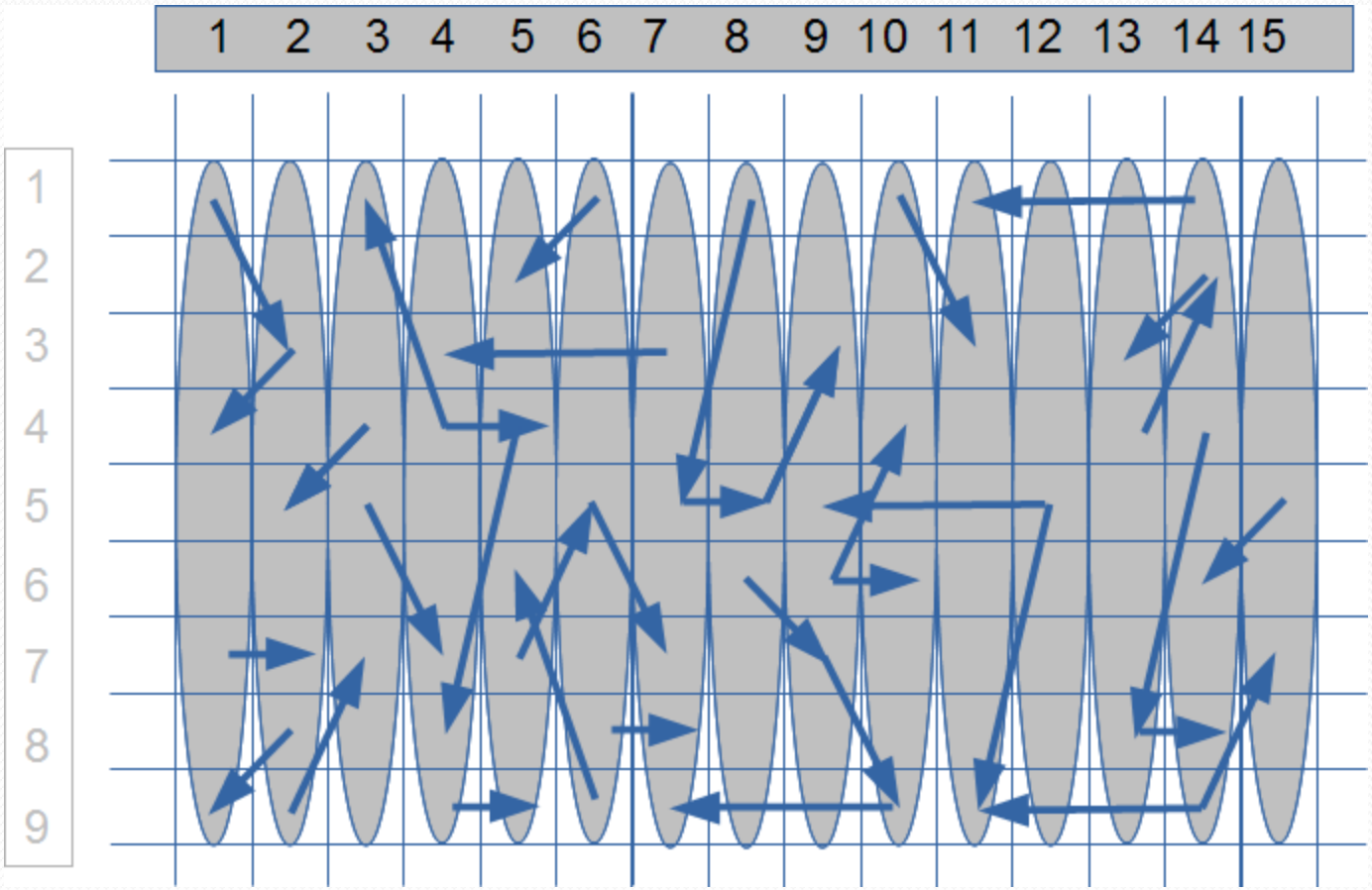


Příklad – stavový prostor



Prostor indukovaný vzorem

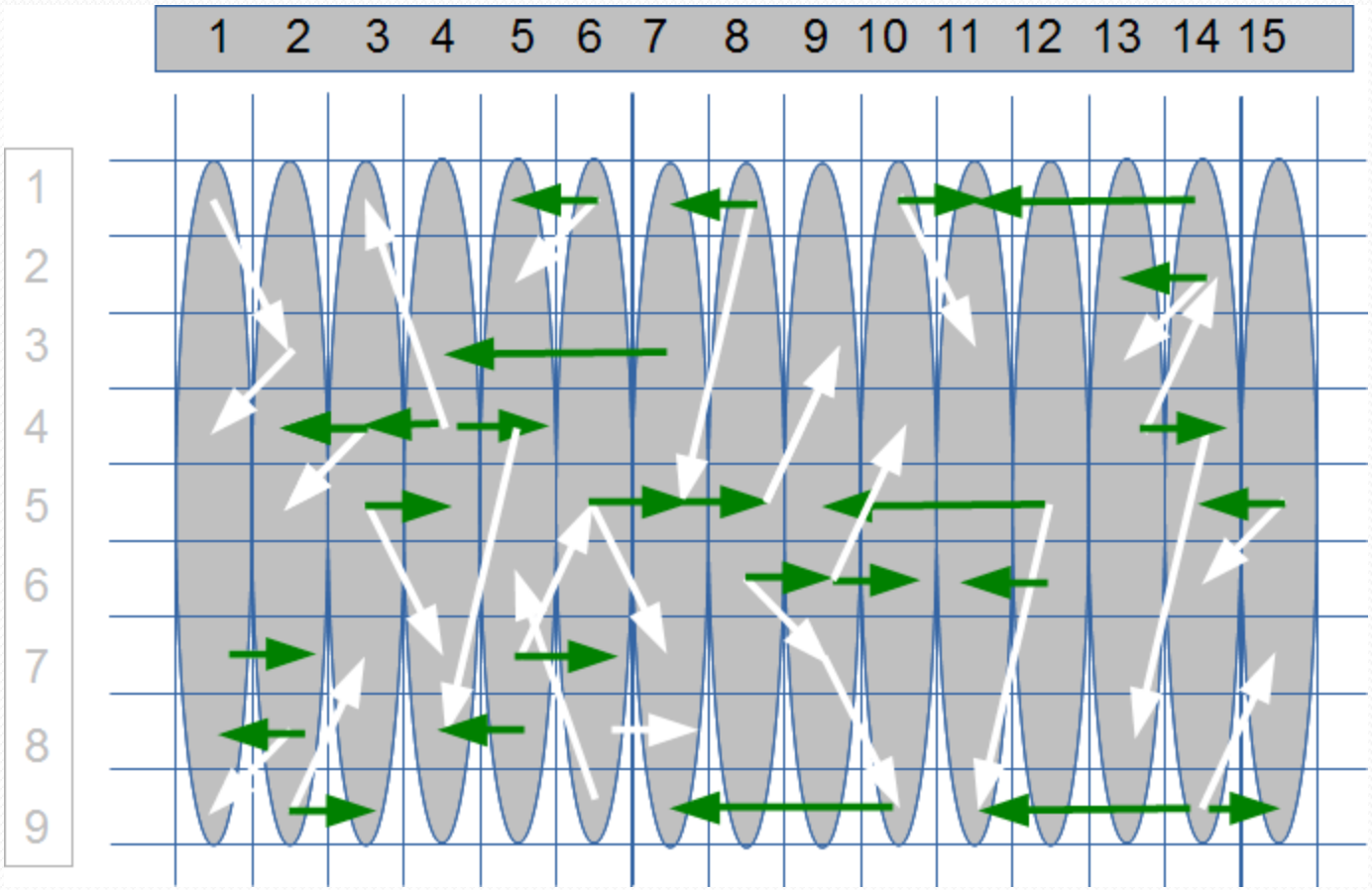
ekvivalence stavů



Prostor indukovaný vzorem

-

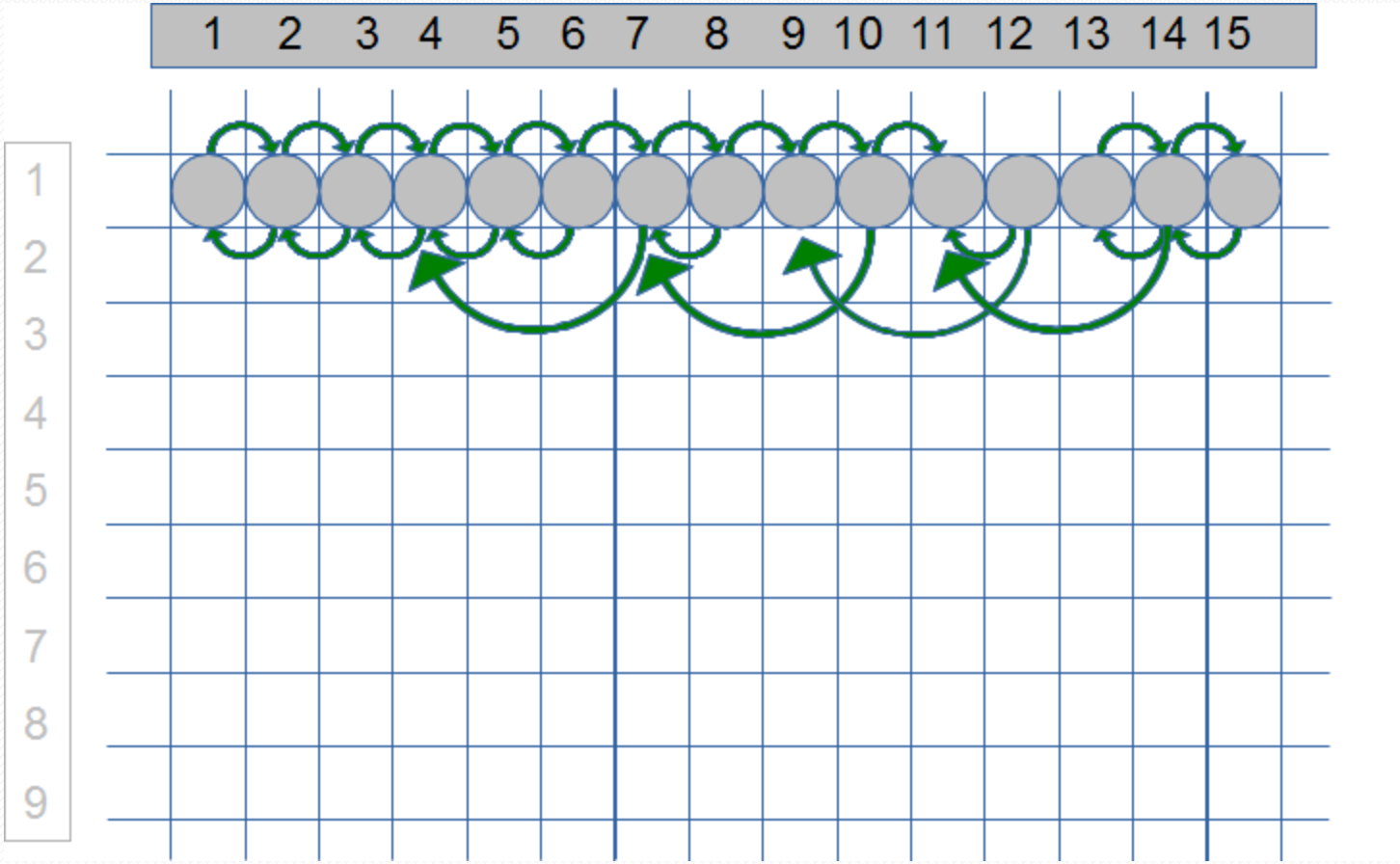
zjednodušení akcí



Prostor indukovaný vzorem

-

skutečná podoba



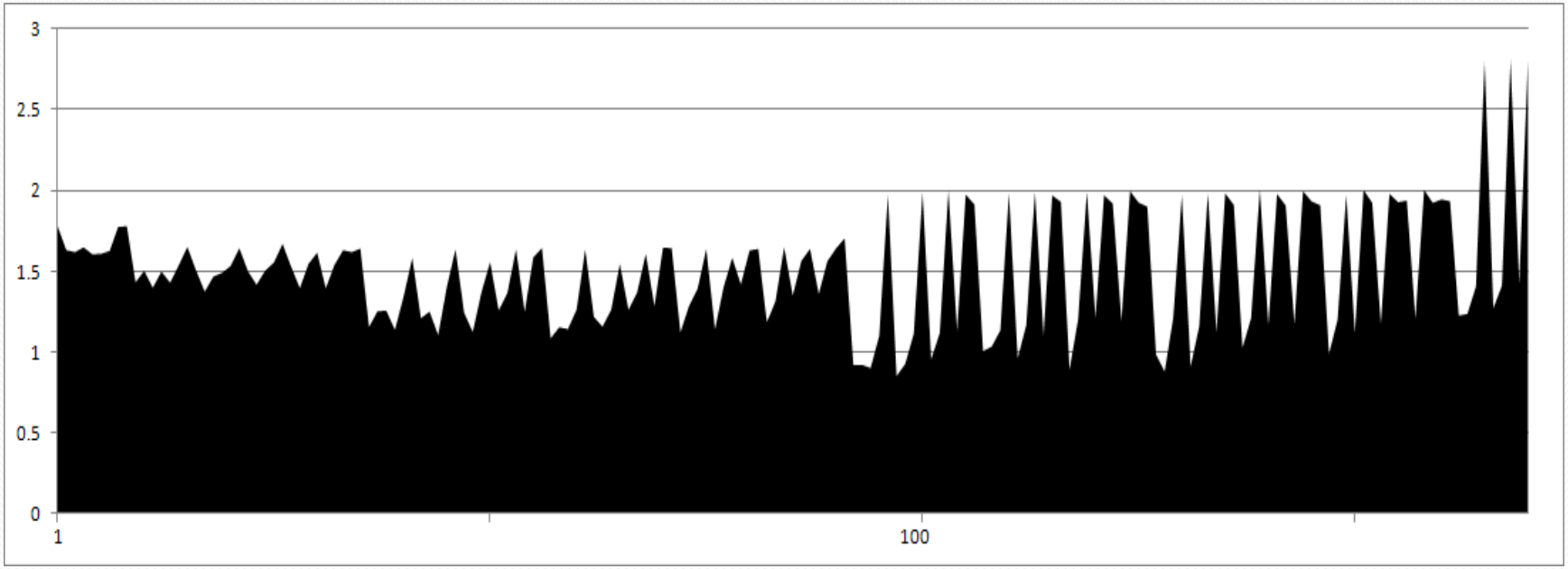
Vzory jako heuristika

- h_S - heuristika indukovaná vzorem S
  - t – stav původního prostoru
  - t_S - stav t omezený na množinu S
- $h_S(t) = \textit{délka nejkratší cesty z } t_S \textit{ do cíle v } P_S$
- Vlastnosti heuristiky:
  - h_S je přípustná
  - $h_\emptyset = 0$, $h_{Var} = h^*$
  - $S_1 \subseteq S_2 \Rightarrow h_{S_1} \leq h_{S_2}$
  - h_S je spočitatelná v konstantním čase
    - Ale vytvoření databáze nějaký čas trvá
  - Paměťová náročnost: $\prod_{v_i \in S} |Dom(v_i)|$

Hledání vzorů

- Jak vybrat množinu vzorů $\{S_i\}$ tak, aby
  - Databáze z $\{S_i\}$ se vešla do paměti
  - Heuristika $h_{\{S_i\}}$ byla co nejinformovanější
- Těžký optimalizační problém
  - Správné vzory můžou výrazně zrychlit prohledávání
  - Velký prohledávaný prostor
  - Obtížně vyčíslitelná ohodnocovací funkce
- Používané techniky:
  - Náhodná volba, případaně jednoduchá heuristika
  - Asistence experta na danou doménu
  - Lokální prohledávání, evoluční algoritmy

Experiment:
Vliv vzorů na rychlost prohledávání



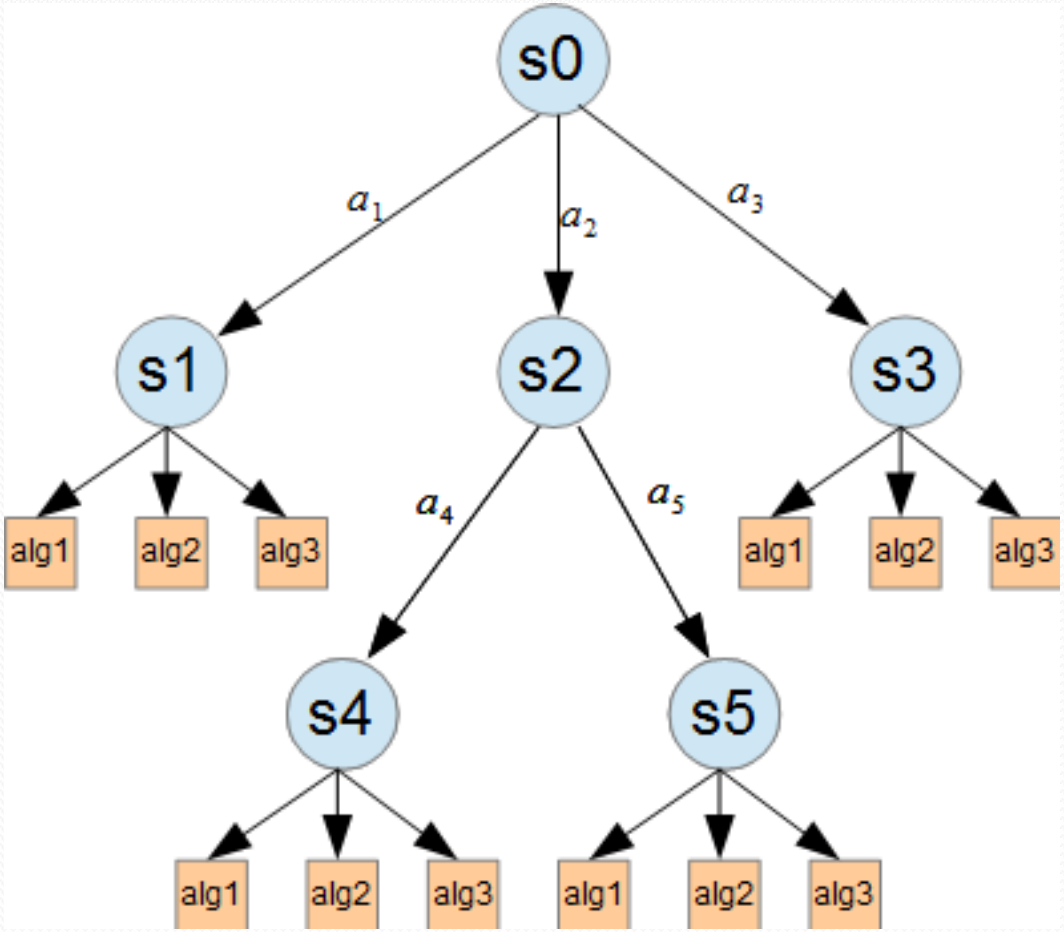
Jaké vzory zvolit?

- Záleží na tom?
  - **Ano!**
  - Nejlepší vzor:
    - Čas pro vytvoření: 1,77 sec, čas hledání: 4.34 sec
  - Nejhorší vzor:
    - Čas pro vytvoření : 79.44 sec, čas hledání : 13.47 sec
  - Slepé prohledávání
    - Čas pro vytvoření : 0 sec, čas hledání: 58.88 sec

Další cíle výzkumu

- Kombinace standardních plánovacích algoritmů s optimalizačními technikami
  1. Rozdělení výpočetního času
  2. Objektivní funkce ve fázi předzpracování
  3. Simulace v MCTS
  4. Kombinační operátory, které vytvářejí přípustná řešení
  5. Celkový návrh plánovače založeného na hyper-heuristice

MCTS jako hyper-heuristika



Děkuji za pozornost