AUTOMATICKÉ ŘEŠENÍ ÚLOH

MARTIN KULDA

OBSAH

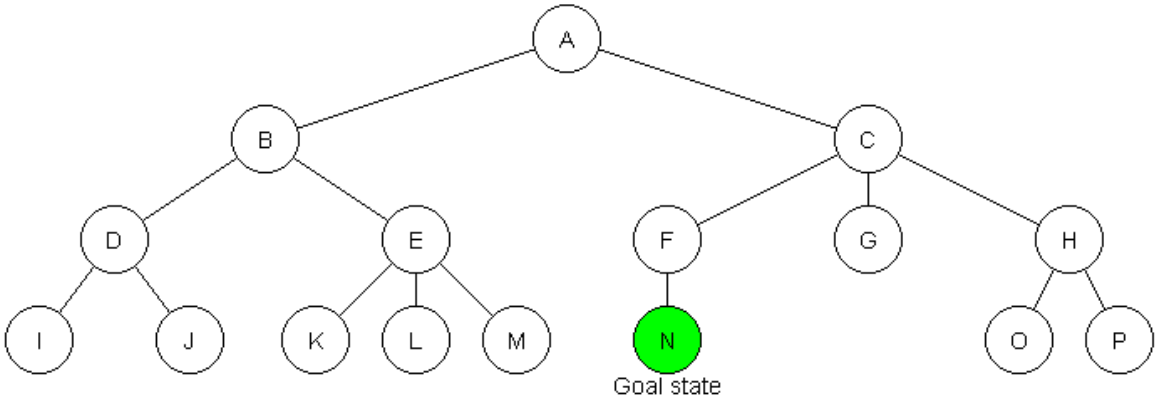
FORMALIZACE ÚLOHY

Pokud chceme využít umělou inteligenci při řešení problému, je zapotřebí reálnou (či teoretickou) situaci zformulovat do přístroji srozumitelného modelu. Určíme si počáteční stav (s_0), cílový stav (s_c) a dostupné operátory (φ_n). Tyto operátory reprezentují operace/akce, které má umělá inteligence k dispozici. Úkolem umělé inteligence je rozvinout tento model, tj. nalézt všechny stavy, kterých lze dosáhnout aplikováním operátorů na jednotlivé stavy a poté určit ideální posloupnost operátorů, jenž nás dovedou do námi požadovaného stavu nejrychleji či nejefektivněji za daných podmínek. Množina všech možných stavů se nazývá stavový prostor a posloupnost vedoucí k vyřešení úlohy se nazývá plán úlohy.

STAVOVÝ PROSTOR

Stavový prostor je nejčastěji reprezentován orientovaným stromem s uzly, jenž je speciální typ stavově prostorového grafu. Jednotlivé uzly reprezentují objevené stavy a spojnice těchto uzlů vždy reprezentují aplikaci některého z operátorů. Je důležité připomenout, že tyto spojnice se díky vlastnostem a uspořádání stromu nebudou nikdy protínat. Výhodou je to, že z daného uzlu k jeho bezprostřednímu následníkovi vede pouze jediná cesta.

Pokud chceme zkoumat stavový prostor, potřebujeme nejdříve nějaký k dispozici. Nejprve se aplikují operátory na počáteční stav (A). Při aplikaci operátorů hovoříme o expanzi uzlu, tím daný uzel rozvíjíme a získáváme *n* počet dalších stavů (B, C). Aplikaci operátorů opakujeme vždy při zkoumání nových stavů. Při zkoumání stavu B objevíme dva nové stavy D, E. Mohou nastat situace, kdy se nově objevený stav v paměti již vyskytuje. Pak je programem zapomenut, protože jeho rozvíjením by se dostal do nekonečné smyčky. Také se může stát, že některý z operátorů nelze aplikovat, v tom případě pokračujeme následujícím operátorem v pořadí.



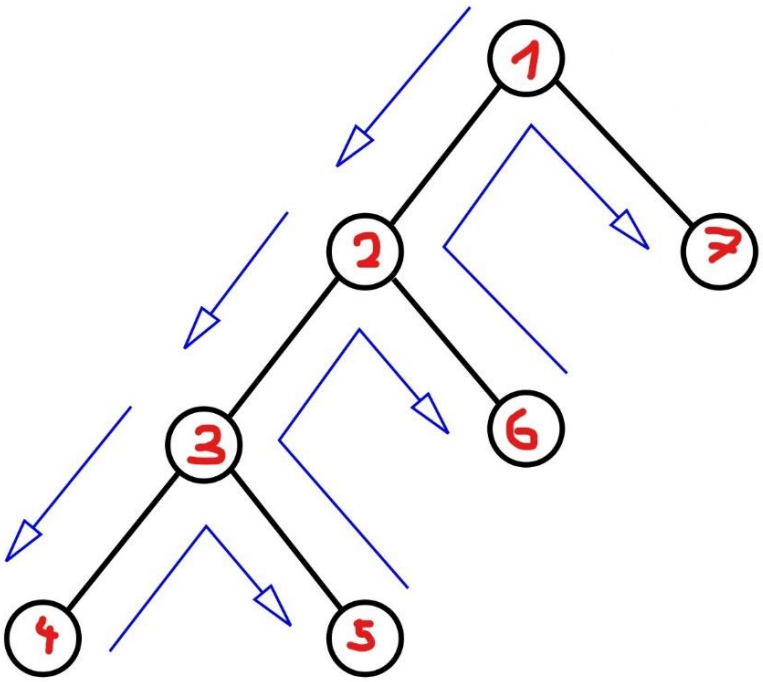
Každý objevený stav má svoji hloubku. Počáteční stav (kořen) má hloubku 0 a každé následující rozvinutí generuje stavy s hloubkou o 1 větší. Pokud tedy zkoumáme stav, který jsme získali aplikací 4 operátorů, nalezneme ho v hloubce 4 a naopak. Hloubka se poté využívá v prohledávacích strategiích (hledání nejkratší možné cesty = nejmenší počet operátorů).

PROHLEDÁVACÍ STRATEGIE

Při prohledávání se využívají dva základní typy strategií. Prvním typem je metoda zpětného navracení (backtracking) a druhým typem jsou metody prohledávání stromů do šířky, do hloubky, s minimální cenou nebo s heuristikami. Tyto strategie se liší nároky na paměť a rychlostí prohledávání. Každá má své výhody, nevýhody a vhodnou aplikaci.

ZPĚTNÉ NAVRACENÍ (BACKTRACKING)

Pokud se rozhodneme pro metodu zpětného prohledávání, počítač si bude pamatovat vždy jen jedinou cestu vedoucí od počátečního stavu k poslednímu vygenerovanému. Tím klade velmi malé nároky na paměť, ale může trvat déle než ostatní strategie, jelikož se některé stavy zkoumají znovu. Program na každý nový stav aplikuje první operátor a teprve až se v dané větvi zastaví nebo narazí na chybu, tak se vrátí o krok zpět a vyzkouší další operátor v pořadí.

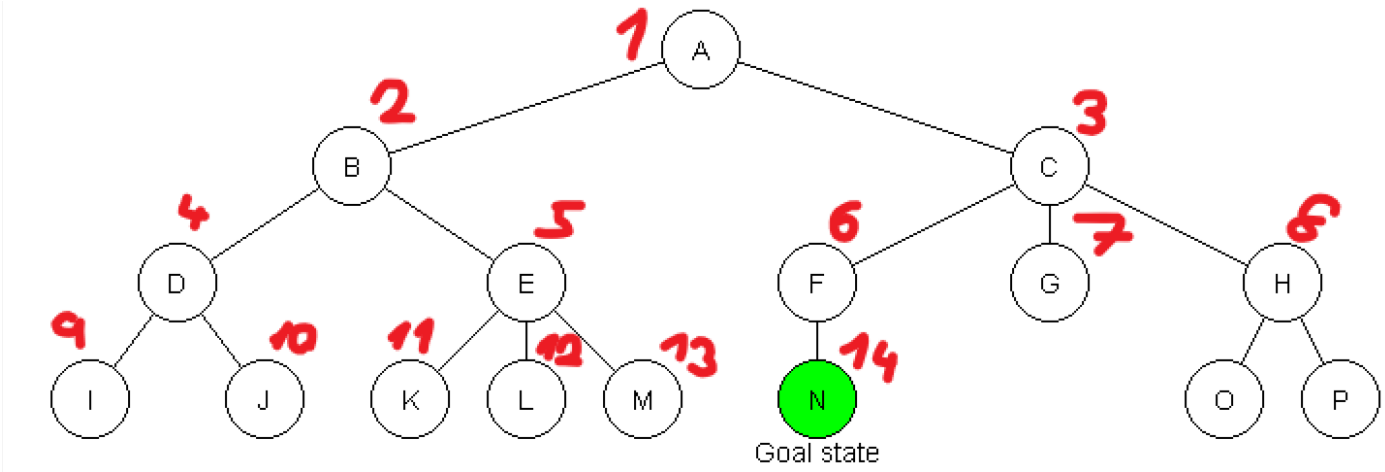


PROHLEDÁVÁNÍ STROMŮ

Následující metody v paměti uchovávají celou postupně generovanou strukturu stromu. Nároky na paměť jsou zde tedy mnohonásobně vyšší, ale pokud program dospěje k výsledku, máme jistotu, že nalezl nejkratší možnou trasu, protože zkoumá najednou všechny stavy, které mohou nastat.

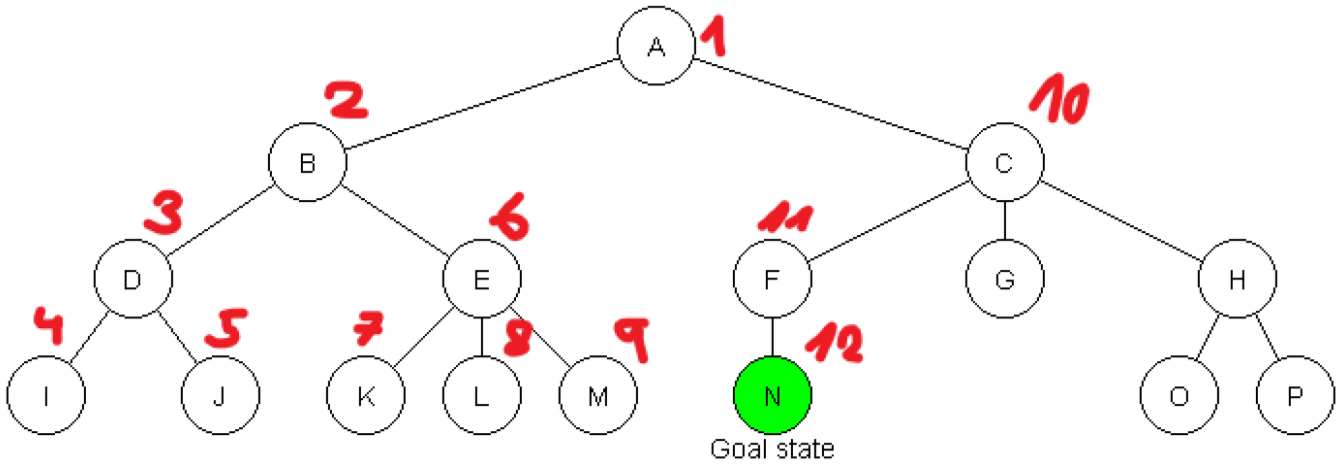
1. DO ŠÍŘKY

Při prohledávání stromu do šířky se na právě zkoumaný stav aplikují všechny operátory najednou. Tím získáme opět určitý počet nových stavů a jako další stav pro expanzi se zvolí ten s nejmenší hloubkou. Ve výsledku se graf rozvíjí po řádcích (po hloubkách). Tato strategie je ideální pro nalezení nejkratší možné cesty.



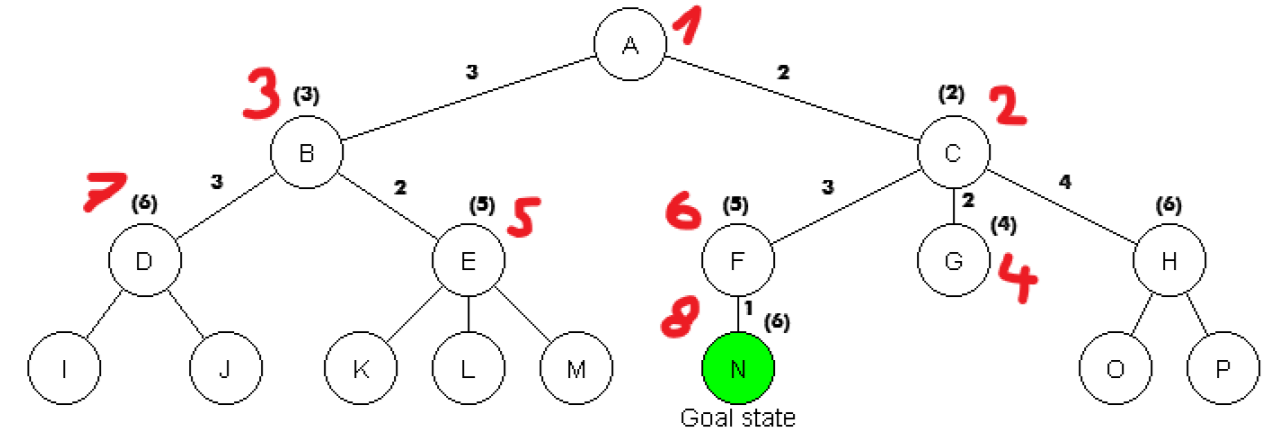
2. DO HLOUBKY

Při prohledávání stromu do hloubky na právě zkoumaný stav aplikujeme postupně všechny operátory jako v předchozí strategii, nicméně na rozdíl od ní se jako následující stav pro expanzi zvolí ten s největší hloubkou. Při troše představivosti se graf v tomto případě rozvíjí po sloupcích zleva doprava. Pokud tuto strategii využíváme u rozsáhlejších úloh, je často potřeba určit si maximální zkoumanou hloubku, abychom nehledali do nekonečna ve „špatné větvi“. Existuje i modifikace, kdy se tato maximální hloubka postupně zvyšuje.



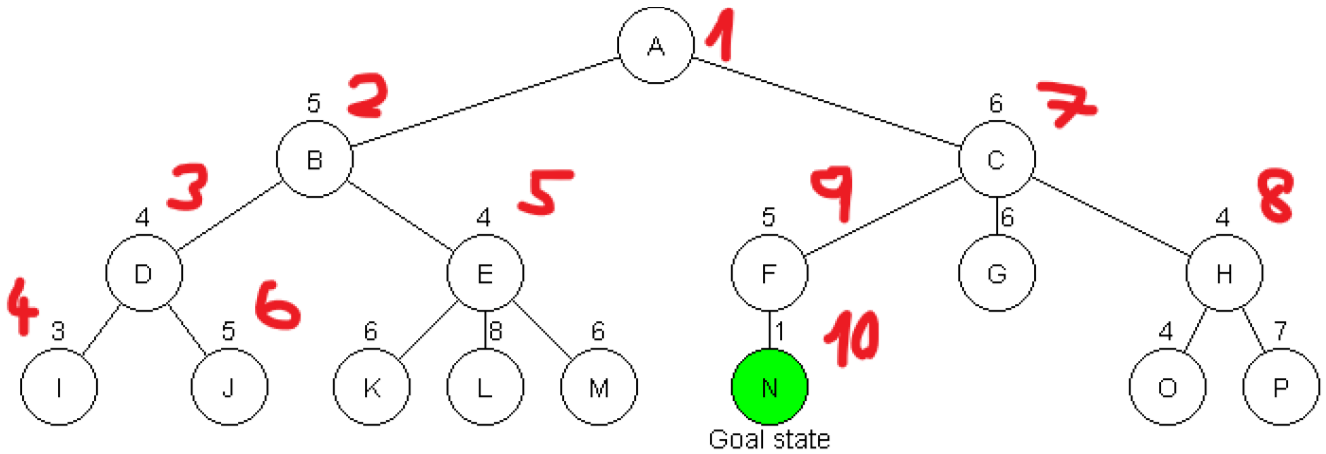
3. S MINIMÁLNÍ CENOU

Následující strategii lze použít například pro počítání s určitým typem energie. Energie je reprezentována cenou jednotlivých operátorů a stavy se pak ohodnocují sumou nákladů (součtem energií) za operace z počátečního do aktuálního stavu a jako další zkoumaný stav se zvolí ten s nejmenší cenou. Během této strategie může program ve stromu řešení hodně „skákat“.



4. S HEURISTIKAMI

V případě složitějších úloh můžeme pro efektivnější řešení přidat heuristiky (dodatečné znalosti). Klademe důraz na zajištění cesty s minimální cenou a nejmenšími nároky na prohledávání. Skvělým příkladem je GPS navigace, kdy hledáme nejkratší cestu z bodu A do bodu N a jako odhad využijeme vzdálenost vzdušnou čarou mezi aktuálním stavem a cílovým stavem. Na tomto základě pak volíme jako další zkoumaný stav ten s nejnižší hodnotou (pokud jich je více, zároveň i s nejnižší hloubkou). Tím máme s největší pravděpodobností zaručeno, že nalezené cesta bude nejkratší (pokud se skutečná vzdálenost nějak drasticky neliší od té vzdušné). Zde narážíme na možnost chyby (nalezení cesty, která se zdá být nejvýhodnější, ale někde existuje ještě lepší). Pro tyto účely se pak stanovují různé rovnice a podmínky. Pokud tyto podmínky splní, program lze nazvat A* (A-star), u něhož máme vždy jistotu, že našel nejvýhodnější existující řešení.



ZDROJE

https://hackernoon.com/search-algorithms-in-artificial-intelligence-8d32c12f6bea

https://www.minigranth.com/artificial-intelligence/problem-solving-in-artificial-intelligence/

http://www1.osu.cz/~volna/Umela_inteligence_skripta.pdf

http://www.kky.zcu.cz/uploads/courses/zui/ZUI-02_prohledavani_bez_min_ceny.pdf

https://www.kiv.zcu.cz/studies/predmety/uir/predn/P2/FThema2new.pdf

http://how2examples.com/artificial-intelligence/tree-search