Automatizované řešení úloh s omezeními

Martin Kot

Katedra informatiky, FEI,
Vysoká škola báňská – Technická universita Ostrava
17. listopadu 15, Ostrava-Poruba 708 33
Česká republika

1. listopadu 2012

Dopředné hledání

Search

- I při uplatnění různých algoritmů pro zajištění konzistentnosti sítě apod. často nakonec nezbyde nic jiného, než použít metodu "pokus – omyl"
- Jedná se o prohledávání prostoru možných řešení
- Často se prohledávání kombinuje s různými způsoby dedukce
- Pojem prohledávání (search) v oblasti CSP zahrnuje mnoho algoritmů, které řeši problémy "hádáním" dalšího kroku, někdy v kombinaci s nějakými heuristikami
- V CSP je prohledávání téměř výhradně spjato s variantami backtrackingu

Backtracking

- Aktuální částečné řešení se vždy rozšíří přiřazením nějakých hodnot dalším proměnným
- Začíná se s "nějak zvolenou" první proměnnou
- Po jedné se dalším proměnným přiřazují provizorní hodnoty a kontroluje se, jestli právě přiřazená hodnota je konzistentní vzhledem k dříve přiřazeným
- Dead-end - stav, kdy žádná hodnota v doméně právě přiřazované proměnné není konzistentní s dříve přiřazenými hodnotami
- Backtracking - po dead-end situaci změníme hodnotu přiřazenou bezprostředně před tou, která k dead-end situaci vedla
- Algoritmus končí nalezením požadovaného počtu řešení, nebo zjištěním, že žádné řešení neexistuje nebo že již byla nalezena všechna řešení
- Výhodou je polynomiální prostorová složitost, nevýhodou exponenciální časová složitost

Backtracking

- Hodně výzkumu se zabývalo backtrackingem
- Byla snaha zmenšit prohledávaný prostor snížením vlastního prostoru řešení nebo efektivnějšími způsoby procházení
- Velikost prostoru řešení závisí hlavně na stupni konzistentnosti sítě a pořadí proměnných při přiřazování hodnot
- Vznikly dva druhy procedur
  - spouštěné před prohledáváním - zmenšení prostoru řešení
  - spouštěné během prohledávání - rozhodnutí, které části prostoru řešení algoritmus vynechá

Stavový prostor

- **Stavový prostor** (state space) je dán 4 prvky:
  - množinou stavů S
  - množinou operátorů O, přiřazují stavy stavům
  - počátečním stavem $s_0 \in S$
  - množinou cílových stavů $S_g \subseteq S$
- Základní cíl je nalezení řešení - sekvence operátorů, které převedou počáteční stav do cílového stavu
- Stavový prostor lze efektivně reprezentovat orientovaným grafem
- **Prohledávací graf** (search graph)
  - uzly reprezentují stavy
  - orientovaná hrana z s_i do s_j znamená, že existuje operátor transformující s_i na s_j
  - koncové uzly (listy) můžou být cílové (reprezentují řešení) a necílové (reprezentují dead-end)
- Prohledávací algoritmy jsou vlastně algoritmy procházející tímto grafem

Prohledávání

- Věta: Nechť R' je striktnější síť než R, kde obě reprezentují stejnou množinu řešení. Pro jakékoliv uspořádání proměnných platí, že každá cesta vyskytující se v prohledávacím grafu vytvořeného z R' se vyskytuje také v grafu vytvořeného z R
- Nabízí se tedy možnost zvážit uplatnění vynucení konzistentnosti před prohledáváním
- Ne vždy je čas využitý na zajišťování konzistence vyvážen úsporou času při prohledávání
- Přidání nových omezení při zajištování vyšších stupňů konzistence také může značně spomalit prohledávání - po přiřazení hodnoty každé proměnné je potřeba kontrolovat splnění těchto omezení
- Pro binární omezení nikdy nebude více než $O(n)$ operací ověřujících konzistenci pro každý stav
- Při obecných omezeních můžeme mít až $O(n^{r-1})$ omezení, kde r je mez arity těchto omezení
- Backtrack-free síť pro uspořádání proměnných d je síť jejíž prohledávací graf má jen cílové listy (tedy každý list odpovídá

Backtracking

- Prochází prohledávací strom do hloubky
- Má dvě fáze:
  - Dopředná - je vybrána další proměnná a aktuální částečné řešení je rozšířeno přiřazením konzistentní hodnoty vybrané proměnné
  - Zpětná - když neexistuje konzistentní hodnota pro aktuálně přiřazovanou proměnnou, vrací se algoritmus k proměnné přiřazované v minulém kroku
- Algoritmus - slidy R. Dechter, chapter 5, slide 12

Složitost rozšíření aktuálního částečného řešení

- Uvažujme omezení uložená v tabulkách
- e je počet omezení, t je maximální počet n-tic v omezení, k maximální velikost domény, r je maximální arita omezení (tedy $t \leq k^r$))
- Omezení je možné uložit tak, aby bylo možné nalézt konkrétní n-tici v logaritmickém čase $\log t \leq r \log k \leq n \log k$
- Protože proměnná může být v rozsahu až e omezení, časová složitost v nejhorším případě procedury CONSISTENT je $O(e \log t)$ (nebo také $O(er \log k)$)
- Procedura SELECT-VALUE může volat CONSISTENT až $k$krát, čili složitost je $O(ek \log t)$ (nebo také $O(ekr \log k)$)
- V binárních sítích se aktuálně přiřazená hodnota ověřuje maximálně vůči n dříve přiřazeným proměnným, tedy CONSISTENT má složitost $O(n)$ a SELECT-VALUE má $O(nk)$
- CONSISTENT dělá i jiné operace než vyhledávání ntic omezení v tabulce. K těmto operacím je nutno přihlédnout, pokud můžou ovlivnit asymptotickou složitost

Vylepšení backtrackingu

- trashing - opakované objevování stejných nekonzistentností a částečných řešení během hledání
- Efektivní řešení problému trashing je nepravděpodobné – jde o NP-úplný problém
- Je ale možné snížit množství trashingu a tím vylepšit výkon prohledávací procedury
- Vylepšení dělíme na dvě skupiny
  - pro dopředný pohyb (look-ahead schemes)
    - Volají se při přípravě na přiřazení hodnoty další proměnné
    - Snaží se zjistit jak přiřazení ovlivní další prohledávání
    - Použitá inference může vést k výběru proměnné k přiřazení nebo k výběru hodnoty pro některou proměnnou
  - pro zpětný pohyb (look-back schemes)
    - Volají se po dead-endu
    - Rozhodují, jak daleko se vrátit analýzou příčin dead-endu (až k příčině problému) - backjumping
    - Můžou zachytit příčinu dead-endu do nového omezení - ukládání omezení (constraint recording), učení omezení (constraint learning)

Strategie pro dopředný pohyb

- Zvyšují čas potřebný pro instanciaci proměnné, ale můžou přinést rychlejší nalezení řešení
- Například můžeme zjistit, že žádná z hodnot aktuálně přiřaditelných by nebyla konzistentní s následujícími a rovnou provést backtrack
- Čím více šíření omezení se použije pro pohled dopředu, tím menší prostor bude potřeba prohledávat, ale tím více času navíc ztratíme
- Je možné při pohledu dopředu proměnným dosud neinstanciovaným díky testům konzistentnosti zmenšit domény a při jejich instanciaci již potom nebude potřeba kontrolovat konzistentnost
- Málokdy se zlepší asymptotická složitost v nejhorším případě
- Klíčové je správné vyvážení mezi režií vnesenou přidanými algoritmy a jejich efektivností
- Algoritmus GENERALIZED-LOOK-AHEAD (slidy R. Dechter, chapter 5, slide 15) - obecný framework pro všechna vylepšení dopředného pohybu, změny strategie se projeví různými procedurami SELECT-VALUE

Forward-checking

- Jedna z metod pro dopředný pohyb pro výběr hodnoty
- Propaguje se efekt výběru hodnoty na každou následující proměnnou
- Když některá z domén následujících proměnných bude prázdná, právě zvažovaná hodnota se nevybere a zkusí následující
- Využívá proceduru SELECT-VALUE-FORWARD-CHECKING (slidy R. Dechter, chapter 5, slide 18)
- Pokud uvažujeme konstantní složitost ověření jednoho omezení na jedné dvojici hodnot, má SELECT-VALUE-FORWARD-CHECKING složitost ek^2 (k je kardinalita největší domény, e je počet omezení)

Arc-Consistency Look-Ahead

- Zajišťuje plnou hranovou konzistentnost všech dosud neinstanciovaných proměnných pro každou možnou hodnotu aktuálně přiřazované proměnné
- Využívá proceduru AC-1 pro hranovou konzistentost, kde některé proměnné již mají přiřazenou hodnotu
- S využitím nejoptimálnějších algoritmů pro hranovou konzistentnost je složitost při jedné proměnné $O(ek^3)$ (k hodnot, pro každou $O(ek^3)$ konzistentnost)
- Označován také jako **real full look-ahead**
- Oblíbenou variantou je MAINTAINING-ARC-CONSISTENCY (MAC)

MAC

- Provádí plnou hranovou konzistentnost kdykoliv je zamítnutá některá hodnota z domény
- Příklad: Vybíráme hodnotu pro x z domény $\{1, 2, 3, 4\}$. Nejprve dáme $x = 1$ a aplikujeme hranovou konzistentnost. Když se někde objeví prázdná doména, $x = 1$ je zamítnuto a doména se redukuje na $\{2, 3, 4\}$. Znovu se provede hranová konzistentnost.

Full Look-Ahead, Partial Look-Ahead

- Obě dělají více práce než forward-checking a méně než plná hranová konzistentnost
- Partial Look-Ahead - složitost je $O(ek^3)$, stejná jako Arc-Consistency Look-Ahead, nevystihuje to ale realitu

LVO

- Informaci získanou šířením omezení je možné použít i k ohodnocení neodmítnutých hodnot podle šance na to, že povedou k řešení
- Look-ahead value ordering (LVO)
- Může využít forward-checking nebo jakýkoliv vyšší stupeň šíření omezení
- Nepřiřadí se rovnou první možná hodnota, zkouší pro každou a zkoumá efekt forward-checking (nebo jiné metody) na domény dosud nepřiřazených proměnných
- LVO přístupy se liší nejen metodou šíření omezení, ale také heuristickými mírami použitými k ohodnocení hodnot. Některé jsou:
  - Min-conflicts (MC) - vybírá hodnotu, která odstraňuje nejméně hodnot z domén následujících proměnných
  - Max-domain-size (MD) - vybírá hodnutu, která bude mít největší minimální velikost domény na následujících proměnných
  - Estimate solutions (ES) - počítá horní mez na počet řešení násobením velikostí domén následujících proměnných po odstraněnní hodnot nekompatibilních s aktuálně uvažovanou. První volí tu s největším výsledkem.

Look-ahead pro uspořádání proměnných

- Uspořádání proměnných má obrovský význam na velikost prohledáváného prostoru
- Empirické a teoretické studie ukazují, že některá fixní uspořádání jsou v obecnosti efektivnější a generují menší prostory
- Např. min-width a max-cardinality jsou docela efektivní

Dynamické uspořádání proměnných

- Pořadí proměnných se určuje během prohledávání
- Běžná metoda je **fail-first** - vybrat vždy proměnnou, která má potenciál nejvíce omezit search space
- Při stejných ostatních faktorech se vybírá proměnná s nejmenší doménou - bude nejméně podstromů s kořeny v jednotlivých hodnotách z domény
- DVO - dynamic variable ordering
- Např. použití SELECT-VARIABLE (slidy R. Dechter, chapter 5, slide 26) v GENERALIZED-LOOK-AHEAD po iniciazaci $i \leftarrow 1$ a po kroku $i \leftarrow i + 1$
- DVFC - dynamic variable forward checking - DVO s využitím forward checking
- DVFC se v mnoha studiích ukázalo efektivní v porovnání přínosu a dodatečné výpočetní zátěže
- Algoritmus upravuje domény následujících proměnných, aby obsahovaly jen hodnoty konzistentní s aktuálním přiřazením. Vybere se proměnná s minimální doménou. (slidy R. Dechter, chapter 5, slide

Backtracking s náhodným výběrem

- Pořadí proměnných i hodnot má velký význam
- Je těžké najít jednu heuristiku, která bude ideální pro všechny případy
- Je možné využít náhodu
- Např. při min-domain bude pro výběr proměnné více kandidátů (se stejně velkou doménou) - zvolí se náhodně
- Náhodně lze volit i hodnoty, případně náhodu využít při "remíze" při využití heuristiky pro volbu hodnoty
- Potom je běžné zkoušet po (vhodně zvoleném) limitu zastavit hledání a spustit jej znovu. Limit je vhodné postupně zvyšovat a tím se zajistí jistota správného výsledku
- V praxi se ukázalo užitečné