Optimalizační problémy

Optimalizační problémy

- V praxi se často rozlišují hard a soft omezení
- Analýza vede k síti omezení rozšířené o globální cenovou (neboli kriteriální) funkci
- Řešení musí splňovat všechna hard omezení a optimalizovat cenovou funkci
- Mnoho problémů z průmyslu je tohoto typu
- Většinu CSP problémů můžeme uvažovat i v optimalizační verzi - když není možné splnit všechna omezení, hledá se řešení splňující co nejvíce omezení
- Max-CSP - třída problémů vzniklá tímto způsobem
- Jedním z nejznámějších problému z Max-CSP je MaxSAT (snaha najit ohodnocení formule v KNF splňující co nejvíce klauzulí)
- Lze také omezením dát váhy a místo počtu minimalizovat sumu vah porušených omezení

Optimalizační problém

- Optimalizační problém s omezeními (COP - constraint omptimization problem) je síť omezení rozšířená o cenovou funkci
- $X = \{x_1, \ldots, x_n\}$
- $F_1, \ldots, F_l$ jsou reálné funkce definované nad $Q_1, \ldots, Q_l$, kde $Q_j \subseteq X$
- Nechť $\bar{a} = \{a_1, \ldots, a_n\}$, kde a_i je z domény x_i, nechť $\bar{a}_i$ je přiřazení prvním i proměnným. Globalní cenová funkce F je definována jako $F(\bar{a}) = \sum_{j=1}^{l} F_j(\bar{a})$, kde $F_j(\bar{a})$ znamená $F_j(\bar{a})$ aplikována na přiřazení $\bar{a}$ omezené na doménu funkce F_j
- Cenová síť (cost network) je $C = (X, D, C_h, C_s)$ kde (X, D, C_h) je síť omezení a $C_s = \{F_{Q_1}, \ldots, F_{Q_l}\}$ je množina cenových komponent nad doménami $Q_1, \ldots, Q_l$, $Q_i = \{X_{i_1}, \ldots, X_{i_l}\}$
- Cenová síť je jednou z možných reprezentací optimalizačního problému s omezeními
- Funkce v C_s jsou měkká (soft) omezení
- Optimalizační úloha je snaha najít $\bar{a^0} = \{a_1, \ldots, a_n\}$ splňující všechna omezení, tž. $F(\bar{a^0}) = max_{\bar{a}} F(\bar{a})$ (popřípadě místo maxima lze hledat i minimum)

Cenový graf

- Uzel pro každou proměnnou
- Hrany pro jakékoliv omezení (hard i soft)

Řešení COP jako série CSP

- Vždy se řeší úloha CSP s hard omezeními a s limitem cenové funkce, který se postupně zvyšuje (tedy omezení, že globalní cenová funkce je větší než limit)
- Postupně se tak zvyšuje kvalita nalezeného řešení
- Řešení je optimální v rámci limitu daného rychlostí zvyšování limitu
- Promyšlenější přístup - použití "binárního vyhledávání"
- Lze tedy použít pro optimalizaci jakékoliv techniky, které se používají (a byly probrány) pro CSP

Branch and Bound search

- Nejjednoduššní a nejnaivnější přístup - rozšířený backtracking
- Po prvním nalezeném řešení se backtracking nezastaví, ale prohledá i zbytek prostoru
- Pamatuje si aktuálně nejlepší nalezené řešení
- Může využít jakoukoliv techniku zlepšení backtrackingu na hard omezeních
- Je možné dále omezit prohledávaný prostor analýzou cost funkce
- Např. když součet funkcí nad již přiřazenými hodnotami je vyšší než dosud nejlepší (s minimální cenovou funkcí) řešení, tak ve větvi není potřeba pokračovat

Depth-first branch and bound

- Udržuje nejlepší řešení
- Počítá horní mez s využitím omezující funkce $f(\bar{a}_i)$, která nadhodnocuje nejlepší řešení, které je rozšířením $\bar{a}_i$
- Když i tento nadhodnocený odhad je menší, než dosud nejlepší nalezené řešení, není potřeba $\bar{a}_i$ dále rozšiřovat (pokračovat ve větvi výpočtu, která k němu vedla)
- Omezující funkce může být použita také jako heuristika pro výběr hodnoty, která se přiřadí aktulně zpracovávané proměnné
- Při hard omezeních využívá techniky zmíněné dříve, ale v každém uzlu prohledávacího stromu přidá kontrolu omezující funkce
- Obdobně je možné řešit minimalizaci

Omezující funkce

- Základním možným vylepšením branch-and-bound je zlepšení přesnosti omezující funkce
- Nejprve byly algoritmy vyvíjeny a zlepšovány pro třídu problémů známých jako celočíselné programování
- Mají lineární tvrdá omezení a globální cenovou funkci, proměnné mají celočíselné domény
- Některé idee se rozšiřují na obecná omezení a obecné cenové funkce

Russian Doll search

- Idea - n po sobě jdoucích branch-and-bound hledání, každé s přidanou jednou proměnnou (a relevantními omezeními)
- První podproblém zahrnuje jen $n$tou proměnnou, $i$tý posledních i proměnných
- Každý podproblém se řeší přes branch-and-bound, kde mezní funkce využívá znalostí z předchozích běhů
- Navíc dříve nalezaná optimální řešení jsou využívána jako heuristiky pro výběr hodnot k přiřazení a ke zlepšení počáteční horní (dolní) meze řešení
- V praxi se ukazuje, že i n prohledávání je efektivní, protože jednotlivá hledání jsou rychlejší díky omezení prohledávaného prostoru

Bucket elimination

- Využívá datovou strukturu bucket
- Proměnné předpokládáme uspořádány podle nějakého uspořádání
- Cenové funkce se dělí do košíků odpovídajících proměnným
- Začne od košíku poslední proměnné a dá do něj všechny funkce s touto proměnnou v doméně
- Do každého dalšího košíku dá funkce, které proměnnou onoho košíku obsahují v doméně, ale zatím nebyly nikam přiděleny
- Košíky se zpracovávají od poslední proměnné
- Zpracování košíku - sečtou se funkce v něm eliminuje se proměnná odpovídající košíku maximalizací
- Vzniklou funkci zařadíme do odpovídajícího nižšího košíku (patřícímu větší proměnné) a pokračujeme následující proměnnou (o jedna větší v uspořádání)
- Po skončení (maximalizaci funkce v košíku nejvyšší proměnné) dostane hodnotu cenové funkce optimálního řešení
- Zpětně od nejvyšší proměnné k nejnižší lze získat přiřazení hodnot všech proměnných pro optimální řešení

Složitost algoritmu bucket elimination

- Složitost je ovlivněna uspořádáním proměnných
- Čím více rozměrné funkce eliminací vznikají, tím více prostoru je potřeba pro jejich uložení (exponenciální růst prostoru)
- Arita funkcí závisíá na počtu proměnných v košíku
- S pomocí grafových technik (výpočet grafové šířky apod.) lze aritu odhadnout
- Složitost je lineární vůči počtu omezení (hard + soft), ale exponenciální vůči indukované grafové šířce

Počítání počtu řešení

- $\sharp P$ - třída problémů, které řeší nedeterministický turingův stroj a řešení je počet akceptujících výpočtů
- Obvykle problémy spojené s problémy v NP
- Některé problémy v rozhodovací verzi jsou polynomiálně řešitelné, ale počítání počtu jejich řešení je $\sharp P$-úplné - např. 2-SAT, splnitelnost formule v DNF, párování apod.
- Počítání počtu řešení lze elegantně řešit eliminací košíků
- Vstupní relaci reprezentuje 1 pro konzistentní a 0 pro nekonzistentní přiřazení
- Místo maximalizace při eliminaci košíku se používá součet, místo sčítání funkcí je násobení
- Lze také řešit prohledáváním, které projde celý prostor jakýmkoliv backtrackingem

Mini-bucket elimination

- Eliminace košíků potřebuje pro některé vstupy exponenciální prostor
- Lze se inspirovat z řešení CSP, kde místo plné konzistentnosti se dělá i-konzistentnost pro nějaké rozumně malé i
- V košíku ukládáme aproximaci prostřednictvím funkcí o menší aritě (minikošíky v rámci košíky)
- Maximalizujeme každou tuto funkci s menší aritou zvlášť a maxima sčítáme. Vzniklá funkce je aproximací původní funkce
- Maximalizovaný souet v prvním koši je potom horní mez
- Dolní mez se spočítá při druhé fázi algoritmu (dohledávání hodnot proměnných)
- Kvalita mezí závisí na stupni rozkladu do mini-košíků
- Je mnoho možností, jak rozdělovat proměnné do minikošíků, vedou k různě přesným mezím a dává proto smysl hledat vhodné heuristiky pro tento rozklad