Report k projektu

Srovnání algoritmů lokálního prohledávání

Jan Jeníček (jenicja1)

Úvod

Cílem této práce je hlubší pochopení algoritmů lokálního prohledávání u optimalizačních problémů a následné porovnání dvou konkrétních přístupů pro hru Sudoku. Tím prvním je základní lokální prohledávání (neboli hill-climbing), což je nejjednodušší z optimalizačních algoritmů. Tím druhým je ß-hill-climbing, což je relativně nový metaheuristický algoritmus publikovaný v roce 2016 [1]. Existuje spousta jiných populárních metaheuristických přístupů k lokálnímu prohledávání (např. simulované žíhání, tabu prohledávání). ß-hill-climbing však údajně dosahuje nejlepších výsledků ze všech a není zdaleka tak populární, proto se projekt zaměřuje právě na něj. Jelikož se jedná o poměrně nový algoritmus, nalezl jsem pouze jednu studii z roku 2017 zabývající se aplikací tohoto algoritmu na Sudoku [2]. Současným cílem tohoto projektu tedy je i implementace algoritmu podle této studie a pokus o jeho aplikaci jiným, novým způsobem, který by dosahoval dobrých výsledků.

Popis algoritmů

Jelikož se v obou případech jedná o lokální prohledávání, fungují algoritmy z obecného pohledu obdobně. Na začátku je vygenerováno náhodné řešení a algoritmus se z něho snaží pomocí náhodných kroků posouvat směrem k lepšímu řešení. Pokaždé, když se posune, je nové řešení ohodnoceno definovanou evaluační funkcí a v případě, že je řešení lepší, než je to předchozí, přesune se algoritmus do onoho lepšího řešení. Z něho dále algoritmus generuje nová řešení a snaží se tímto způsobem dospět k tomu nejlepšímu.

Nejzásadnějším rozdílem mezi oběma algoritmy je jejich přístup k exploraci, tedy k prozkoumávání nových oblastí prostoru všech řešení. Samotný hill-climbing je čistě exploitativní, tedy pouze prozkoumává jednu a tu samou oblast. V každém svém kroku se posune pouze o jedno řešení předem definovaným způsobem a může tak snadno předčasně konvergovat k zdánlivě ideálnímu řešení, jelikož jiné oblasti prozkoumávat nechce. Jeho výsledek je proto závislý na vygenerované počáteční konfiguraci. ß-hill-climbing hledá kompromis mezi explotací a explorací. Zatímco se tedy v prozkoumávané oblasti snaží směřovat k optimálnímu řešení, objevuje i nové oblasti a může se tedy předčasné konvergenci snadno vyvarovat.

ß-hill-climbing

Jelikož se jedná o metaheuristický algoritmus, stručně nejprve popíšu jeho funkčnost. Na začátku je každý atomický prvek řešení vygenerován zcela náhodně. Následně se řešení obdobně, jako základní hill-climbing pomocí náhodných kroků posouvá směrem k optimu. V každém tomto kroku jsou na současné řešení postupně aplikovány dva operatory. Nejprve je aplikován N-operátor (sousední operátor), který každý z atomických prvků řešení s určitou pravděpodobností změní na sousední prvek. Poté je aplikován ß-operátor, který zcela odznovu vygeneruje prvky sousedního řešení, každý prvek opět s nějakou (jinou) pravděpodobností. Řešení z přegenerovanými atomy se poté porovná s tím původním a vybere se pro další postup lepší z nich.

Implementované algoritmy

V projektu jsou implementovány tři různé algoritmy

- hill-climbing (tedy základní lokální prohledávání)
- ß-hill-climbing podle studie [2]
- vlastní verze ß-hill-climbingu

hill-climbing

Nejprve je tedy vygenerováno počáteční řešení. To se generuje po řádkách a postupně vyplňuje náhodně všechna volná políčka tak, aby se každém řádku vyskytovala všechna čísla pouze jednou. Pro posun k sousednímu řešení se náhodně vybere jeden řádek, v němž se prohodí libovolné 2 pole, která byla vyplněná algoritmem. Jako evaluační funkce slouží počet chyb v celém Sudoku, tedy čím méně, tím lépe. Jelikož samotný hill-climbing postrádá exploraci, je řešení vždy hledáno opakovaně a algoritmus je několikrát restartován. Díky tomu začíná v různých místech prostoru řešení a zvětšuje se tak šance nalezení globálního optima alespoň v jednom z běhů.

ß-hill-climbing podle studie

Jako atomické prvky řešení jsou chápány jednotlivá pole. Ty jsou tedy z počátku vygenerovány zcela náhodně bez ohledu na pravidla Sudoku. N-operátor je definován jako posun k vedlejšímu číslu. Pokud je tedy pole vybráno, je k jeho hodnotě přičtena, či odečtena 1. ß-operátor je intuitivní a pokud je pole vybráno, zcela náhodně mu vygeneruje novou hodnotu. Jako evaluační funkce je zvolena odchylka součtu hodnot všech polí v řádku/sloupci/čtverci od hodnoty 45 (optimum). Výsledkem této funkce je součet absolutních hodnot odchylek pro všechny tyto oblasti. Tu se snaží algoritmus minimalizovat.

vlastní verze ß-hill-climbingu

Jedná se o kompromis mezi dvěmi výše zmíněnými metodami. Jako atomické prvky jsou voleny řádky a jejich inicializace probíhá jako u základního hill-climbingu, jsou tedy vygenerovány náhodně, ale v rámci řádku korektně. Posun je take stejný a pokud je řádek zvolen, jsou v něm náhodně prohozena dvě pole. ß-operátor řádek v případě zvolení zcela přegeneruje, ovšem však pouze unikátními čísly. Jako evaluační funkce je zvolen počet chyb v Sudoku, opět obdobně, jako u prvního algoritmu.

Experimenty

Jelikož se mi však způsob hledání řešení prezentovaný v druhém algoritmu (ß-hill-climbing podle studie) jevil jako nefunkční a po implementaci algoritmus nehledal optimální řešení, jsou veškeré experimenty provedeny pouze na zbylých dvou algoritmech (tedy hill-climbing a vlastní verze ß-hill-climbingu). Součet odchylky dle mého názoru nezaručuje unikátnost čísel v zdánlivě optimálním řešení a čísla se v něm tak většinou opakují. Pokud je tedy v experimentech řeč o hill-climbingu, je na mysli první algoritmus a v případě ß-hill-climbingu je na mysli třetí algoritmus.

Hledání ideálních N a ß pravděpodobností

První experiment hledal ideální hodnoty pro pravděpodobnosti aplikace N-operátoru a ß-operátoru na atomy u ß-hill-climbingu. Jako ideální hodnoty se nakonec jak z pohledu úspěšnosti, tak z pohledu rychlosti hledání řešení ukázalo N = 0.25 a ß = 0.05. V každému kroku má tedy každý řádek nejprve 25% šanci, že v něm budou prohozena dvě pole a následně 5% šanci, že bude celý přegenerován.

Příliš nízké hodnoty u obou parametrů hledaly velmi dobře optimální řešení, trvalo to ale velmi dlouho. Příliš velké hodnoty způsobovaly velké skoky mezi řešeními, optimum tak často nebylo ani nalezeno.

Hledání ideálního počtu restartů

Oběma algoritmům byl dán celkový počet kroků, který mají k dispozici a ten mohly rozdistribuovat mezi libovolný počet restartů. To bylo velmi efektivní především pro hill-climbing, který díky čisté explotaci velmi rychle konverguje, velmi často však pouze k lokálnímu optimu. Pokud tedy hill-climbing optimální řešení najde, stane se to většinou během prvních 2500 iterací a vyplatí se ho tak často restartovat. Naopak ß-hill-climbing řešení nalezne téměř vždy, v průměru k tomu potřebuje ale skoro 10x tolik iterací. Výsledek je tak očekávaný.

Srovnání času běhu každého restart

V tomto experimentu se potvrdilo, že hill-climbing je závislý na počáteční konfiguraci. Zatímco v nejlepších případech byl schopen nalézt řešení rychleji, než ß-hill-climbing, velmi často uvázl. Jeho výsledky proto byly velmi nekonzistentní. ß-hill-climbing tak v tomto testu dosahoval v průměru lepších výsledků především díky tomu, že téměř vždy našel optimální řešení před dosažením maximálního počtu kroků.

Srovnání rychlosti hledání řešení

Řešení v průměru lépe hledal základní hill-climbing s restarty. Ačkoliv v dílčích testech ukázal ß-hill-climbing silné stránky metaheuristických lokálních prohledávání, má implementace tohoto algoritmu nedokázala hill-climbing porazit. Nejpravděpodobnější problem vidím ve zvolení příliš velkých atomických prvků řešení pro již tak triciální problem, jako je řešení sudoku.

Výše zmíněné experimenty pracovaly se Sudoku 9x9. U menšího Sudoku 4x4 dosahoval mnohem lepších řešení ß-hill-climbing, jelikož není mnoho kombinatorických možností pro řešení tohoto problému a explorace je tak mnohem důležitější. Sudoku 16x16 a větší nebyl ani jeden z algoritmů schopen v rozumném čase upočítat.

Aplikace

V konzolové aplikaci je možnost spuštění všech testů, popřípadě ukázka řešení Sudoku 9x9 pomocí algoritmů použitých v experimentech. K spuštění aplikace stačí spustit python script main.py

Odkazy

[1] https://www.researchgate.net/publication/319886025_b-Hill_Climbing_Algorithm_for_Sudoku_Game

[2] https://www.bau.edu.jo/UserPortal/UserProfile/PostsAttach/98216_992_1.pdf