Soft Computing (SFC)

Migrating Bird Optimization na řešení problému batohu

Daniel Konečný (xkonec75)
Vysoké učení technické v Brně
Fakulta informačních technologií

27. listopadu 2020

1 Úvod

Cílem práce je naprogramovat aplikaci, která aplikuje algoritmus Migrating Birds Optimization (MBO) na řešení praktické úlohy. MBO je algoritmus založený na tvaru hejna migrujících ptáků. Ptáci využívají tvaru písmene V pro efektivnější let, a touto myšlenkou se dá inspirovat při řešení optimalizačních problémů. V této práci se využívá MBO na řešení problému batohu (Knapsack Problem 01), což je problém řadící se do třídy NP. Optimalizační algoritmus dosahuje optimálních výsledků na složitějších problémech do 50 předmětů a na jednodušších problémech do 100 předmětů.

2 Problém batohu

Cílem řešení problému batohu (Knapsack Problem 01) je naplnit ho předměty tak, aby nebyla přesažena jeho kapacita. Každý předmět má danou hmotnost a hodnotu. Optimální řešení je takové, které dosáhne nejvyššího možného součtu hodnot předmětů v batohu při nepřesažení jeho kapacity součtem hmotností předmětů v batohu. Tato práce se soustředí na binární jednodimenzionální problém batohu. Každý předmět se tedy může nacházet v batohu maximálně jednou (binárnost) a jedinou omezující vlastností je hmotnost předmětů (jednodimenzionalita). Při výpočtech lze konfiguraci batohu definovat jako sekvenci 0 a 1. Dále se používá funkce měřící hmotnost předmětů, která zaručí, že nepřevyšují kapacitu. A nakonec fitness funkce ohodnocující kvalitu dané konfigurace jako součet hodnot předmětů.

3 Migrating Birds Optimization

Migrating Birds Optimization (MBO) neboli algoritmus optimalizující hejnem migrujících ptáků je založený na typickém přírodním úkazu, kdy ptáci při migrování létají ve formaci tvaru písmene V. Tato formace jim dovolí doletět dále díky ušetření energie, zároveň také může zjednodušovat orientaci ptáků při letu, ale to není v tomto algoritmu stěžejní. Ušetření energie je docíleno tím, že ptáci na čele víří vzduch takovým způsobem, že okolní ptáci na krajích formace jsou nadnášeni. Ptáci si postupně mění pozice a tím rovnoměrně rozloží náročnost dlouhého letu na celé hejno. Těchto principů je využito v MBO algoritmu.[1]

Inicializace algoritmu probíhá tak, že se vytvoří počáteční hejno uspořádané do písmene V tak, že jeden pták je v čele a z každé strany je obklopen stejným počtem ptáků (hejno má tedy lichý počet ptáků). Celkově je v hejně z ptáků, kteří jsou náhodně inicializování, splňují však případná omezení. Následně se algoritmus soustředí na ptáka v čele, ke kterému vypočítá k alternativ. Tyto alternativy se vyhodnotí fitness funkcí, a nejlepší z nich je porovnána s hodnotou fitness funkce ptáka v čele. Pokud je vyšší, alternativní pták nahradí ptáka v čele. x alternativ s druhým nejlepším ohodnocením je sdíleno s ptákem nalevo od čelního a x za nimi následujících alternativ (dle ohodnocení) je sdíleno s ptákem napravo.

Dále už se jednotlivé větve hejna (levá a pravá) provádějí nezávisle na sobě a stejným způsobem. Ptákovi následujícímu za čelním ptákem je vypočítáno $k - x$ alternativ a k nim je přidáno x alternativ předaných od ptáka nad ním (v tomto případě čelního ptáka), celkem má tedy také k alternativ. Stejně jako u čelního ptáka, nejlépe ohodnocená alternativa je nabídnuta jako vylepšení tomuto ptákovi a x následujících nejlépe ohodnocených alternativ je předáno ptákovi za ním. Ptákům v druhé větvi však není už dále nic předáno. To je rozdílné od výpočtu u čelního ptáka, kde bylo předáváno x hodnot nalevo a x napravo. Je tedy třeba, aby bylo $k \geq 2x + 1$. U "nečelních" ptáků pak zůstává alespoň x alternativ nijak nevyužitých (zahazují se).

Jakmile je dokončem výpočet pro všechny ptáky, opakuje se znovu od čelního ptáka. Jedné této iteraci se říká "tour" a algoritmus jich provede m. Poté algoritmus vezme čelního ptáka a dá ho na konec levé větve, která se tímto celá posune dopředu. Pták, který byl původně prvním zleva pod čelním, je nově čelním ptákem. A provede se znovu m iterací tour. Poté je čelní pták přesunut na konec pravé větve a ta je stejným způsobem jako levá předtím posunuta směrem k čelu hejna. A znovu se provádí m tour. Tento postup se opakuje pořád dokola s tím, že se střídá posun levé a pravé větve a po $2z$ posunech je hejno ve stejném uspořádání, jako bylo na začátku.

Ukončovací podmínkou K je velikost problému umocněná trojkou. Každé generování množiny alternativ se počítá za 1. Jakmile je tedy generování alternativ provedeno $K = (velikost\_problemu)^3$, algoritmus je zastaven. Pták, který představuje podle fitness funkce nejlepší možné řešení, je stanoven jako výsledek.

Algoritmus tedy využívá několika principů, které mu pomáhají dosáhnout co nejlepšího řešení. Jedním z nich je princip sousedství, tedy hledání řešení,

které je blízko aktuálnímu, ale dosahuje lepšího výsledku. Také využívá principu předávání možných řešení dále.

4 Aplikování MBO na problém batohu

Každý pták v hejnu představuje jednu možnou konfiguraci předmětů v batohu splňující kapacitu. Je to tedy sekvence 0 a 1 o délce odpovídající celkovému uvažovanému počtu předmětů. Hejno lze tedy chápat jako podmnožinu všech takových konfigurací, které vyhovují podmínce kapacity batohu. Cílem algoritmu je nalézt alespoň jedním řešením optimum.

Algoritmus odpovídá postupu výše popsanému, pouze se musí konkretizovat jak jsou vybírány alternativy a jak se spočítá velikost problému. Algoritmus hledá alternativy k ptákům tak, že přidá do batohu náhodně jeden předmět, který tam ještě není. Pokud je převýšena kapacita, náhodně odebírá předměty, dokud není podmínka kapacity znovu neporušena. Velikost problému je rovna celkovému uvažovanému počtu předmětů.[2]

5 Ukázka výpočtu

Níže je ukázka výpočtu levé větve (liché indexy pro level-order číslování ptáků) u tour pro $z = 5$, $k = 3$ a $x = 1$. Problém batohu je definovaný takto, že $w = \{2, 5, 7, 3, 1\}$ představuje množinu hmotností předmětů, $p = \{10, 30, 70, 50, 1\}$ představuje možinu hodnot předmětů a $c = 10$ je kapacita batohu. Alternativy už jsou předem seřazené podle ohodnocení. Tučně jsou vyznačené konfigurace a hodnoty ptáků po této tour.[2]

Pták	Konfig.	Hodn.	Alternativy	Konfig.	Hodn.
S_0	**01011**	**81**	N_{01} - nevyužitý kandidát	00110	80
			N_{02} - sdíleno vlevo	00101	71
			N_{03} - sdíleno vpravo	11001	41
S_1	10100	80	N_{11} - využitý kandidát	**10101**	**81**
			N_{12} - sdíleno dále	00110	80
			$N_{13} = N_{02}$ - zahozeno	00101	71
S_3	11000	40	$N_{21} = N_{12}$ - využitý kandidát	**00110**	**80**
			N_{22} - sdíleno dále	00010	50
			N_{23} - zahozeno	11001	41

6 Implementace algoritmu

Algoritmus je implementován v jazyce C++ podle standardu C++11 a využívá standardní knihovny. Je využito objektově orientovaného návrhu (ne však čistě objektového), ve kterém se využívá objektu zastupujícího batoh, dále objektu symbolizujícího samotný MBO algoritmus a nakonec objektů reprezentujících jednotlivé ptáky. Soubory aplikace jsou následující:

- `sfc_project.cpp` - samotný projekt, překládaný do finálního spustitelného souboru;
- `params.cpp` - pro zpracování vstupních argumentů;
- `Knapsack.cpp` - symbolizuje problém batohu;
- `Mbo.cpp` - symoblizuje MBO algoritmus;
- `Bird.cpp` - symbolizuje jednoho ptáka;
- `Makefile` - slouží pro překlad a jednoduché spuštění;
- `test_script.sh` - několik ukázkových spuštění aplikace pomocí příkazů `make`;
- `/datasets` - složka obsahující mnoho zadání pro problém batohu určených k testování.

7 Přeložení a spuštění aplikace

Aplikaci je možné přeložit příkazem `make` v kořenovém adresáři aplikace. Pro spuštění je možné využít přímo spustitelný soubor `sfc_project`, `make help` vypíše, jaké parametry aplikace očekává a v jaké podobě. Pro zjednodušení jsou však připraveny další příkazy make.

Příkazem `make inform TEST=xx` spustíte konkrétní problém batohu označený dvouciferným číslem xx ze složky `datasets`. Dostupná jsou čísla 01-08[1], 10-27[2]. Ne všechny z těchto problémů dokáže aplikace řešit vždy správně. Navíc v této variantě spuštění aplikace vypíše konfiguraci batohu, MBO algoritmu i vyhodnocení nalezeného řešení.

Další možné spuštění je pomocí `make test TEST=xx RUNS=y`, kde je stejným způsobem jako v předchozím případě označován konkrétní problém batohu xx. Navíc je přidána možnost spustit algoritmus na daném problému vícekrát (y-krát). Po doběhnutí všech cyklů je vypsána statistika s nejlepším a nejhorším nalezeným ohodnocením a průměrem ze všech nalezených ohodnocení.

Spuštění různých variant těchto testů je možné pomocí skriptu `test_script.sh`. Konkrétně spustí problém batohu 21, který je velmi náročný, ale algoritmus by měl najít optimální řešení. Dále spustí problém 18, který má 2 možná řešení, aplikace si však s vyhodnocením umí poradit. Nakonec spustí problém 08, u kterého není nalezeno řešení vždy, spustí ho tedy desetkrát a porovná naměřené hodnoty.

Parametry algoritmu MBO je možné nastavit v souboru `sfc_project.cpp` v hlavičce, jedná se o makra. Aktuálně jsou nastavené na $z = 71$, $k = 5$, $x = 1$ a $m = 30$, což jsou doporučované hodnoty a i z testů vycházejí jako vhodné.

[1] Originálně dostupné zde: `https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html`

[2] Originálně dostupné zde: `http://hjemmesider.diku.dk/~pisinger/codes.html`

Zvýšením z (počtu ptáků) nedocházelo při testování ke zpřesnění výsledků, spíše pouze k prodloužení doby běhu, stejně tak u m. Jiná volba k a x neměla na řešení také přílišný dobrý vliv, dobu běhu také moc neovlivnila.

8 Vyhodnocení výsledků

V tabulce je uvedeno testování algoritmu na různě složitých problémech, různém počtu předmětů a různé velikosti uvažovaných hmotností a hodnot předmětů. Každý problém je řešen desetkrát, poté jsou zpracovány výsledné hodnoty. Tučně vyznačené výsledky jsou optimální řešení.

Problém	Předměty	Optimální	Nejlepší	Průměrné	Nejhorší
P_{06}	7	1735	**1735**	1735	1735
P_{07}	15	1458	**1458**	1457.1	1454
P_{08}	24	13549094	**13549094**	13520103.9	13494864
P_{18}	23	9767	**9767**	9767	9767
P_{21}	50	68946	**68946**	68946	68946
P_{22}	100	81021	**81021**	81021	81021
P_{24}	100	23982	**23982**	23968.3	23945
P_{26}	50	2586	**2586**	2586	2586
P_{27}	100	1011	1005	1002.6	1002

Algoritmus nalezne optimální řešení u všech testovaných problémů s počtem předmětů přibližně do 20. U složitějších problémů s počtem předmětů nad 20 nenalezne optimální řešení vždy. Jednoduché problémy o počtu 50, ale i 100 předmětů zvládne řešit také s velkou jistotou. Pro náročné problémy (vysoká korelace předmětů) o 50 a více předmětech však nemusí nalézt optimální řešení ani v mnoha iterací.

9 Závěr

V rámci práce došlo ke kompletní a úspěšné implementaci algoritmu Migrating Birds Optimization pro řešení problému batohu. Součástí implementace je i mnoho pomocných funkcí pro jednoduché vyhodnocení výsledků a testování různých instancí problémů. Algoritmus dosahuje dobrých výsledků a zvládne řešit poměrně složité varianty problému batohu, nedosahuje však schopnostmi na různé implementace soustředěné na co nejefektivnější řešení právě tohoto problému. Doba běhu programu je poměrně uspokojivá, pro většinu problémů, které zvládne řešit, nalezne řešení během několika sekund.

Citace

[1] Ekrem Duman, Mitat Uysal, and Ali Fuat Alkaya. Migrating birds optimization: A new meta-heuristic approach and its application to the quadratic assignment problem. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcázar, Juan J. Merelo, Ferrante Neri, Mike Preuss, Hendrik Richter, Julian Togelius, and Georgios N. Yannakakis, editors, *Applications of Evolutionary Computation*, pages 254–263, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[2] Erkan Ulker and Vahit Tongur. Migrating birds optimization (mbo) algorithm to solve knapsack problem. *Procedia Computer Science*, 111:71 – 76, 2017. The 8th International Conference on Advances in Information Technology.