Seznámení se se zvolenou pokročilou iterativní metodou na problému batohu

Marek Nevole
nevolmar@fit.cvut.cz

11. listopadu 2021

Abstrakt

Problém 0/1 batohu je jedním z nejznámějších kombinatorických problémů. Cílem tohoto úkolu bylo implementovat metodu pokročilejšího iterativního prohledávání k řešení konstruktivní verze tohoto problému. Pro tuto práci byla vybrána metaheuristika simulovaného ochlazování známější pod názvem simulovaného žíhání, jelikož napodobuje proces žíhání oceli. Metaheuristika přináší velice dobré výsledky z hlediska výpočetního času a relativní chyby řešení, ovšem nezaručuje optimálnost řešení.

1 Úvod

Problém 0/1 batohu je známý kombinatorický problém. Běžná instance tohoto problému je zadána parametry n, M, kde n je počet dostupných předmětů a M je kapacitu batohu. Dále následuje seznam předmětů, které jsou reprezentovány dvojicemi. Dvojice se skládá ze dvou čísel. První udává váhu předmětu a druhé jeho hodnotu/cenu. Pro účely tohoto papíru byla brána v potaz konstruktivní verze problému, ve které je úkolem poskládat věci do batohu tak, aby součet vah věcí nepřesahoval kapacitu batohu a zároveň součet hodnot věcí byl maximalizován. Problém bude řešen metaheuristikou simulovaného žíhání. Odkaz na celé zadání zde.

2 Simulované ochlazování(žíhání)

Jak název napovídá, tato metaheuristika simuluje žíhání oceli, což je proces, při kterém dochází k opakovanému zahřívání a chlazení kovů za účelem snížení tvrdosti, zvýšení tažnosti a napomáhající eliminaci vnitřních pnutí.

Simulované žíhání je pokročilá metoda lokálního prohledávání, která staví na běžném stoupání do kopce, které rozšiřuje o možnost přijmutí zhoršujícího stavu, za pomocí zavedeného parametru teploty t, jímž se dokáže vymanit z lokálního optima a dále divergovat k globálnímu optimu. Pro ohodnocení stavu je využita účelová(cenová) funkce f, jejíž hodnotu se snaží metaheuristika minimalizovat/maximalizovat.

Algorithm 1: Simulated annealing

```
1  state ← initialize();
2  best ← ∅;
3  t ← initialize_temp();
4  while !stop_condition do
5  |   new_state ←
   |    find_candidate_state();
6  |   Δ ← f(new_state) − f(state);
7  |   if f(new_state) > f(state) or
   |    exp(Δ/t) > random(0,1) then
8  |   |   state ← new_state;
9  |   end
10 |   if f(state) > f(best) then
11 |   |   best ← state;
12 |   end
13 |   t ← cooling_schedule(t);
14 end
15 return best
```

V následujících částech konkretizujeme části z pseudokódu na hledání řešení konstruktivní verze problému 0/1 batohu.

Cenová funkce

Cenová funkce f pro problém batohu je jednoduše součet cen c všech předmětů, které se v daném stavu nacházejí v batohu.

$$f = \sum_{bag} c_i$$

Inicializace počátečního stavu

Inicializace počátečního stavu je velice důležitý proces, který značně ovlivňuje dobu výpočetního času. V této práci probíhal následovně:

1. Vytvoř náhodnou permutaci předmětů
2. Projdi všechny předměty v pořadí dáné permutací a přidávej do batohu předměty, pokud se do něj vejdou

Tento způsob se snaží naplnit batoh předměty, tak aby byl zpočátku naplněn blízko ke kapacitě, což ve většině případech povede k vyšší počateční hodnotě cenové funkce, ale zároveň záchovává náhodnost inicializace, tedy možnost pro restartování algoritmu a začátek z jiného stavu. Další možností je využití rychlé heuristiky Redux, která ovšem tuto náhodnost nedovoluje, a při restartování algoritmu by začínal z totožného stavu.

Inicializace teploty

Teplota hraje důležitou roli v prohledávání vytvořeného stavového prostoru. V případě, kdy kandidátní stav zhoršuje cenovou funkci, tedy $f(\text{kandidátní}) < f(\text{současné})$, je tento zhoršující stav přijmut s pravděpodobností:

$$\mathrm{P}(\text{kandidátní}) = \exp(\tfrac{\Delta}{t})$$

, kde Δ je rozdíl cenových funkcí současného a kandidátního stavu. Z tohoto vztahu plyne, že při velké teplotě t jsou přijímana velká zhoršení, naopak při nizké teplotě jsou zhoršení přijímana s nízkou pravděpodobností.

Jednou z možností, jak vypočítat počáteční teplotu, je nalezení kandidátního stavu, takového který nejvíce zhorší cenovou funkci předešlého stavu v celém stavovém prostoru a následně dosazením do vzorce:

$$t_0 = |\tfrac{\Delta}{\ln(x)}|$$

, kde x je pravděpodobnost, s jakou je tento zhoršující stav přijmut, pokud by byl nalezen v první iteraci běhu algoritmu. Obvykle je x nastaveno na hodnotu z rozmezí [0.5, 1].

Ochlazovácí rozvrh

Možností, jak měnit teplotu mezi iteracemi, je velké množství. Nejběžnějšími uváděnými jsou pomocí logaritmického a geometrického ochlazování. Ochlazování pomocí logaritmického ochlazovácího rozvrhu probíhá dle následujícího vzorce:

$$t_i = \frac{t_0}{\log(i+1)}$$

Logaritmické ochlazování oproti geometrickému konverguje mnohem pomaleji, což ve většině případech známená pouze delší výpočetní čas při relativně stejné kvalitě řešení. Geometrický ochlazovací rozvrh je sestaven podle vzorce:

$$t_i = \alpha t_{i-1}$$

, kde α je zadaný parametr zvolený z rozsahu (0,1) určující rychlost chlazení.

Důsledkem ochlazování teploty je přijimání méně zhoršujících stavů v průběhu algoritmu a nastává tzv. intenzifikace, neboli algoritmus více konverguje k finálnímu řešení.

Hledání kandidátního stavu

Hledání kandidátního stavu pro problém batohu je následující.

Při současném stavu je vybrán náhodný předmět, který není v batohu. Pokud je možné tento předmět vložit do batohu a není překročena kapacita, pak je tak učiněno. Pokud je však překročena kapacita batohu, pak jsou odebírány náhodně předměty z batohu, dokud se místo pro vybraný předmět neudělá. Tímto operátorem kandidátního stavu je vytvořen celý stavový prostor, který je prohledáván.

Ukončení

Simulované žíhání lze ukončit několika způsoby. Pro problém batohu je možné zavést časový limit, jednoduchý iterační limit nebo iterační limit v optimu. Iterační limit v optimu ukončí algoritmus, pokud po nastaveném počtu iterací není nalezen lepší stav než současný, což se jeví jako ideální, dynamické ukončení algoritmu.

3 Experimenty

Experimenty byly provedeny na 500 instancích o velikosti $n \in \{32, 35, 37, 40\}$, které byly náhodně generované pro 2. úkol. Čas řešení byl měřen jako system + CPU time, z výsledného času je vynechán sleep time. Testy byly spuštěny na následující platformě: Intel Core i5-3350P 4 Cores 3.10 GHz, Windows 10, Python 3.8.8.

Experimenty byly rozděleny na 2 části. Konkrétně na white box a black box fázi.

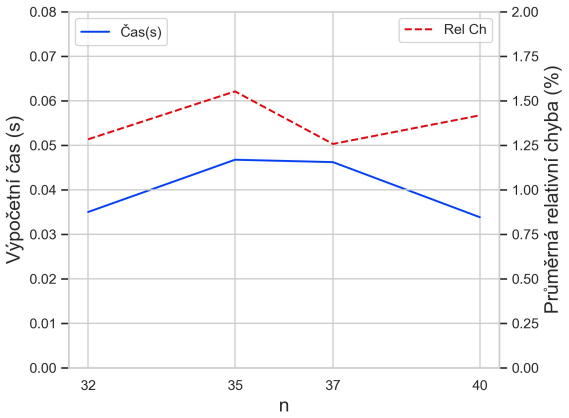
White box fáze

Experiment, jehož výsledky lze pozorovat z tabulky 1, se zabýval nastavením parametrů α, určující rychlost chlazení pomocí geometrického ochlazovácího rozvrhu, a p, podle kterého se nastavuje teplota, aby nejhorší možný zhoršující stav v prvním kroku byl přijat s pravděpodobností p.

Tabulka 1: Porovnání průměrné relativní chyby a výpočetního času v závislosti na zvolených parametrech α a p. Testy proběhly na 500 náhodně vygenerovaných instancích $n = 40$.

	err(%)	t(s)	err(%)	t(s)	err(%)	t(s)	err(%)	t(s)
p \ α	0.8		0.9		0.95		0.99	
0.4	2.63	0.030	2.38	0.037	2.37	0.028	1.55	0.031
0.6	2.34	0.028	2.33	0.027	1.94	0.025	1.46	0.031
0.8	2.48	0.028	2.71	0.025	2.25	0.024	1.41	0.033

Z výsledků v tabulce 1 lze vyčíst, že algoritmus vracel výsledky s nejmenší relativní chybou při nastavení parametrů $\alpha = 0.99$ a $p = 0.8$. Toto nastavení parametru p umožnilo s větší pravděpodobností unikat z lokálním optim při prohledávání. Parametr $\alpha = 0.99$ znamenal pomalejší konvergenci, což se projevilo na výpočetním čase, který byl ze všech nastavení zde logicky nejvyšší.



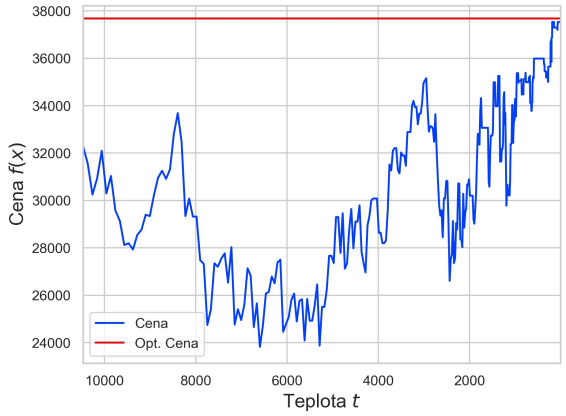
Obrázek 1: Průměrná relativní chyba a vypočetní čas v závislosti na parametru velikosti instance n.

Black box fáze

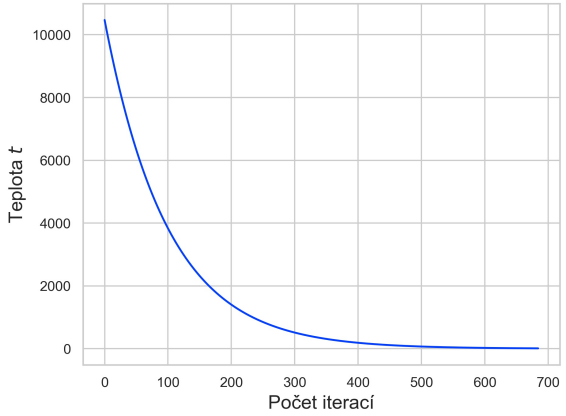
V druhé části experimentů byl algoritmus simulovaného ochlazování testován s parametry $\alpha = 0.99$ a $p = 0.8$, jejichž hodnoty byly vybrany v předešlých testech. Testy byly provedeny na 500 instancích o velikosti $n \in \{32, 35, 37, 40\}$.

Z obrázku 1 je patrné, že výpočetní čas algoritmu není závislý pouze na n. V tomto experimentu byl nejmenší průměrný výpočetní čas pro instance $n = 40$. Příčinou tohoto výsledku je způsob, kterým je algoritmus ukončen. Iterační limit, který při přesáhnutí vynucuje ukončení algoritmu, není staticky nastaven, ale dynamicky upravován, pokud je nalezeno nové nejlepší řešení je tento limit zvýšen. Lze předpokládat, že při důkladnějším testování na větším počtu testovacích dat, by byla lépe viditelná výpočetní závislost na parametru n, jelikož složitost operátoru kandidátního stavu, který je použit v každé iteraci, je $O(n)$. Průměrná relativní chyba je v rozmezí 1–2 %.

Na obrázku 2 a 3 lze sledovat běžný průběh algoritmu. Obrázek 2 ukazuje vývoj cenové funkce f v závisloti na teplotě t. Červená přímka představuje optimální výsledek, ke kterému f konverguje. Obrázek 3 je pouhé znázornění geometrického ochlazovácího rozvrhu a ukazuje teplotu t v závislosti na počtu iterací algoritmu.



Obrázek 2: Průběh algoritmu a vývoj ceny a teploty. Červená přímka představuje optimální řešení.



Obrázek 3: Vývoj teploty podle geometrického ochlazovacího rozvrhu podle počtu iterací.

4 Závěr

V této práci byla implementována metoda simulovaného ochlazování, pomocí které byla řešena konstruktivní verze problému 0/1 batohu.

Experimenty byly rozděleny na fázi hledání vhodných hodnot parametrů a na samotné testování algoritmu s nalezenými parametry.

Z výsledků je patrné, že algoritmus si lepé vedl, z hlediska průměrné relativní chyby, při pomalejší konvergenci, tedy nastavení parametru α na hodnoty bližší k 1. Nastavení parametru p, který měl za úkol pomoci algoritmu uniknout z lokálních optim, bylo opět vyhodnější pro hodnoty blíže k 1. Z experimentů vyšlo optimální nastavení těchto parametrů na hodnoty $p = 0.8$, $\alpha = 0.99$.

Výpočetní čas a relativní chyba se pro instance o velikosti 32, 35, 37 a 40 výrazně nelišily. Výpočetní čas se pohyboval kolem 40ms a relativní chyba byla v průměru 1,4 %.