Exaktní metody řešení rozhodovacího problému batohu

Marek Nevole

nevolmar@fit.cvut.cz

24. září 2021

Abstrakt

Problém 0/1 batohu je jedním z nejznámějších kombinatorických problémů. Cílém tohoto úkolu bylo seznámení se s tímto problémem a implementování metody hrubé síly k řešení rozhodovací verze tohoto problému, která v rozumném čase dokázala řešit instance pouze do $n = 25$. Vylepšení oproti metodě hrubé síly přinesla metoda větví a hranic, která dovolila řešit v rozumném čase náhodně generované instance až do $n = 40$ a instance generované proti algoritmům hrubé síly do $n = 25$.

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. Nejčastěji se jedná o konstruktivní verzi problému, ve které je úkolem poskládat věci do batohu tak, aby součet vah věcí nepřesahoval kapacitu batohu a zároveň součet hodnot věcí byl maximalizován. Avšak v tomto úkole byla řešena rozhodovací verze. Rozhodovací verze přidává další parametr B, který určuje minimální povolenou hodnotu celého batohu. Úkolem je poté rozhodnout, zda je možné poskládat předměty do batohu tak, aby byla splněna tato hodnota B. Odkaz na celé zadání zde.

2 Metoda hrubé síly

Nejjednodušším úplným algoritmem je metoda hrubé síly. Hrubou sílu lze využít při řešení téměř všech kombinatorických problémů. Jinými slovy se jedná o metodu, která prochází všechny možné kombinace a hledá kombinaci, která by splnila zadání. Pro problém batohu byl využit rekurzivní algoritmus zpětného vyhledávání (backtracking), který funguje obdobně jako algoritmus DFS. Postupně zkouší přidávat do batohu věci v pořadí, v jakém byly zadány, a v každém kroku kontroluje, zda není batoh přeplněný nebo nebylo dosaženo požadované minimální ceny, v případě neúspěchu se vrací a odebírá věci z batohu.

Algorithm 1: Backtrack(item_id, price, weight)

```
if weight > max_capacity then
    return False;
end
if min_price =< price then
    return True;
end
if ran out of items then
    return False;
end
# Add current item;
Backtrack( item_id + 1, price +
  item_price, weight + item_weight );
# Don't add current item;
Backtrack( item_id + 1, price,
  weight);
```

3 Metoda větví a hranic

Metoda větví a hranic je jednoduché rozšíření původního algoritmu hrubé síly, které výrazně prořeže a tím zmenší stavový prostor, který musí být následně prohledán. Implementace této metody pro rozhodovací problém se líší od té pro konstruktivní verzi problému. Jelikož se rozhoduje pouze o faktu, zda je možné poskládat věci do batohu nad určitou cenu, tak není potřeba ukládat seznam konkrétních věci a tedy ani nejlepší výsledek. Upravená metoda větví a hranic si ukládá hodnotu všech zbylých věcí, o kterých dosud nebylo rozhodnuto, pokud je tato hodnota při součtu s dosavadní cenou batohu menší než požadovaný limit, tak nemá smysl prohledávat tuto větev stavového prostoru a algoritmus se vrací.

Algorithm 2: BacktrackBNB(item_id, price, weight, remaining_price)

```
if weight > max_capacity then
    return False;
end
if min_price =< price then
    return True;
end
if price + remaining_price <
  min_price then
    return False;
end
if ran out of items then
    return False;
end
# Add current item;
BacktrackBNB(item_id + 1, price +
  item_price, weight + item_weight,
  remaining_price - item_price);
# Don't add current item;
BacktrackBNB(item_id + 1, price,
  weight, remaining_price -
  item_price);
```

4 Experimenty

Pro testování byly vygenerovány dvě testovací sady N a Z. Sada N byla zcela náhodně vygenerována. Sada Z byla vygenerována tak, aby byla pro tyto algoritmy hrubé síly, které nevyužívají heuristiky k chytřejšímu výběru předmětů, značně obtížnější k vyřešení. Obě sady obsahují stejný počet testů, co se parametru n týče, Tabulka 1: Průměrný počet zkontrolovaných konfigurací. BF - Hrubá síla, BNB - metoda větví a hranic.

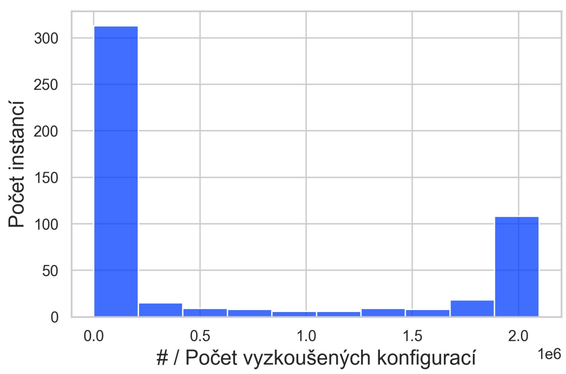
n	N BF	N BNB	Z BF	Z BNB
4	14,12	5,9	19,58	13,92
10	639,58	23,9	966,14	321,11
15	18831.07	113,43	29696,71	5680,35
20	621456,67	624,59	956429,02	115211,20
22	2214382,45	1704,51	3806039.26	358539,30
25	2637584,74	7479,4	-	2238403,93
27	-	11429.07	-	-
30	-	39556,94	-	-
32	-	145090,48	-	-
35	-	320827,63	-	-
37	-	782639,43	-	-
40	-	3304695,22	-	-

konkrétně $n = 4\ 10\ 15\ 20\ 22\ 25\ 27\ 30\ 32\ 35\ 37\ 40$. Čas řešení byl měřen pomocí počtu konfigurací batohu, které byly otestovány. Testy byly spuštěny na následující platformě: AMD FX-9830P 4 Cores 3,00 GHz, Windows 10, Python 3.9.6.

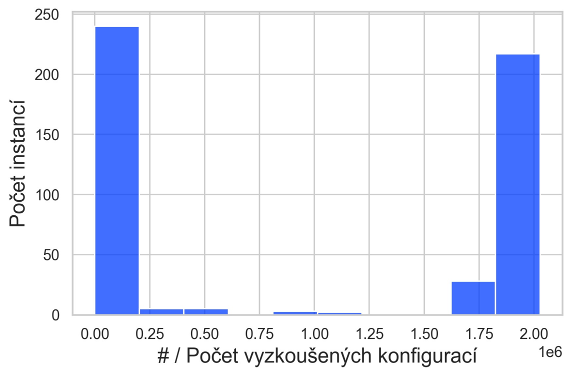
Z tabulky 1 lze pozorovat výrazné zrychlení pomocí metody větví a hranic na náhodně generovaných datech oproti samotné hrubé síle, avšak ani tato metoda nestačila na všechny testy z testovací sady Z a skončila u testů $n = 25$.

Ve všech velikostech testovacích sad se projevil jistý trend pozorovatelný na obrázku 1. O většině instancí šlo rozhodnout během prvních 10 procent vyzkoušených konfigurací z maximálního možného počtu. Tento trend se projevuje jednoznačněji na testovací sadě Z řešené pouze pomocí hrubé síly. Výsledky lze pozorovat na obrázku 2.

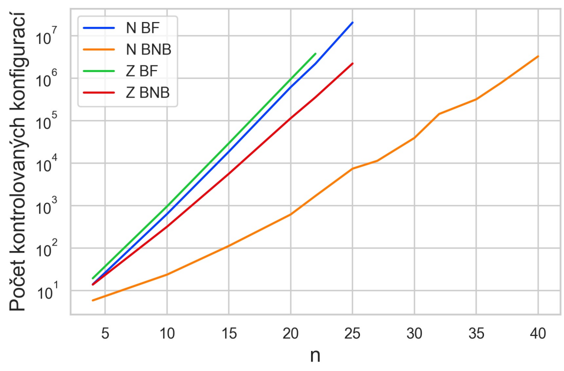
Dle obrázku 3 lze vyvodit, že algoritmy v závislosti na parametru n vyžadují exponenciálně rostoucí počet navštívených konfigurací. Maximální možný počet navštívených konfigurací činí 2^{n+1}, což lze pozorovat na obrázcích 1 a 2.



Obrázek 1: Histogram počtu vyzkoušených konfigurací pro $n = 20$ za použití hrubé síly na náhodně generovaných datech.



Obrázek 2: Histogram počtu vyzkoušených konfigurací pro $n = 20$ za použití hrubé síly na zlomyslně generovaných datech.



Obrázek 3: Počet vyzkoušených konfigurací v závislosti na parametru n a všech testovaných datasetech a metodách.

5 Závěr

V tomto úkole byly implementovány 2 z možností, jak řešit kombinatorický problém 0/1 batohu. První metodou byla hrubá síla a druhou bylo vylepšení té první, metodou větví a hranic.

Z provedených experimentů je patrné, že druhý přístup je výrazně rychlejší než první na náhodně generovaných datech. Na druhé testovací sadě je mezi metodami rozdíl dle parametru n nižší, ale stále znatelný. Z testů je zřejmé, že druhá testovací sada je zaměřená proti algoritmům hrubé síly, jelikož algoritmus hrubé síly musel v téměř polovině instancích vyzkoušet přes 90 % všech možných konfigurací. Metoda větví a hranic si vedla lépe, ale stále je to velice naivní algoritmus, který postrádá heuristiky pro lepší výběr předmětů. Střední hodnotu počtu vyzkoušených konfigurací silně zkreslují krajní hodnoty zobrazené na obrazcích 1 a 2. Dle obrázku 3 je zřetelný rozdíl středních hodnot ve všech testovacích sadách a rozhodně na nich závisí, což je způsobem, jakým byly generovány testovací sady a zvolenými algoritmy.