Experimentální hodnocení kvality algoritmů

Marek Nevole
nevolmar@fit.cvut.cz

17. listopadu 2021

Abstrakt

Cílém tohoto úkolu bylo provést detailní experimenty na algoritmech řešících konstruktivní verzi problému batohu. Konkrétně testovanými algoritmy byly metoda větví a hranic, dynamické programování s dekompozicí podle ceny a nakonec hladová heuristika řadící předměty podle poměru cena/váha. Cílem experimentů bylo odhalit závislosti implementovaných algoritmů na statických parametrech generovaných instancí a na zápisu instancí generovaných pomocí přiloženého generátoru.

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 bereme v potaz 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. Odkaz na celé zadání zde. Z předešlých prací je evidentní, že způsob zápisu a generování instancí má vliv na výkon algoritmů. Cílem této práce bylo provést detailní analýzu a odhalit závislosti.

2 Řešící algoritmy

V této části jsou popsané implementované metody, a k nim přiložené pseudokódy.

2.1 Metoda větví a hranic

Metoda větví a hranic je jednoduché rozšírení algoritmu hrubé síly, které výrazně prořeže a tím zmenší stavový prostor, který musí být následně prohledán. Implementace této metody pro konstruktivní verzi problému, oproti rozhodovací, je rozšířena o ukládání nejlepšího současně dosaženého řešení. Při rekurzivním průchodu stavového prostoru je poté kontrolováno, zda pomocí zbývajících předmětů lze překonat cenu současného nejlepšího řešení. Pokud je hodnota zbylých věcí při součtu s dosavadní cenou batohu menší než cena nejlepší dosud nalezeného řešení, tak nemá smysl prohledávat tuto větev stavového prostoru a al-

goritmus se vrací.

Algorithm 1: BacktrackBNB(item_id, price, weight, remaining_price)

```
1  if weight > max_capacity then
2  |  return False;
3  end
4  if current_solution is better then
5  |  best_solution ← current_solution ;
6  end
7  if price + remaining_price <
     best_price then
8  |  return False;
9  end
10 if ran out of items then
11 |  return False;
12 end
13 # Add current item;
14 solution[item_id] = 1;
15 BacktrackBNB(item_id + 1, price +
     item_price, weight + item_weight,
     remaining_price - item_price);
16 # Don't add current item;
17 solution[item_id] = 0;
     BacktrackBNB(item_id + 1, price,
     weight, remaining_price -
     item_price);
```

2.2 Hladová heuristika

Hladová heuristika je velice jednoduchý způsob, kterým lze řešit problém batohu. U algoritmu ovšem není zaručeno, že vždy vratí optimální maximalizující řešení. Principem heuristiky je seřadit předměty podle poměru ceny/váhy od největšího a postupně je přidávat do batohu, pokud není přesažena kapacita batohu. Po lineárním průchodu přes všechny věci je vráceno řešení.

Algorithm 2: Greedy(sorted_items)

```
1  weight = 0;
2  price = 0;
3  solution = bit vector filled with
     zeroes;
4  foreach item in sorted_items do
5  |  if item.weight + weight <=
   |    capacity then
6  |  |  weight + = item.weight;
7  |  |  price + = item.price;
8  |  |  solution[item.original_index]
   |  |    = 1;
9  |  end
10 end
11 return solution, price, weight
```

2.3 Dynamické programování

Zcela jiným přístupem je dynamické programování. Dynamické programovaní využívá paměti k memoizování již provedených výpočtů a tedy není nutné provádět stejné výpočty vícekrát, což na úkor paměti, při správném použití, ušetří mnoho výpočetního času. Dynamické programovaní je vždy spojeno s rozkladem neboli dekompozicí problému na menší problémy. Pro problém batohu dle dekompozice podle ceny je vytvořeno pole o velikosti počet předmětů + 1 × součet cen předmětů + 1. Tabulku W vyplníme podle následujících pravidel:

$W(0, 0) = 0$,
$W(0, c) = \infty$ pro všechna $c > 0$,

$W(i+1,c) = min(W(i,c), W(i,c-c_{i+1}) + w_{i+1})$ pro všechna $i > 0$, kde w_i je váha i-tého předmětu a c_i je cena i-tého předmětu.

Každou buňku tabulky navštívíme právě jednou, tedy složitost tohoto algoritmu se odvíjí podle počtu předmětů, ale hlavně podle součtu cen všech předmětů, který je vždy vyšší než počet předmětů. Složitost činí O(n*součet cen).

Algorithm 3: DP(items, sum_of_prices)

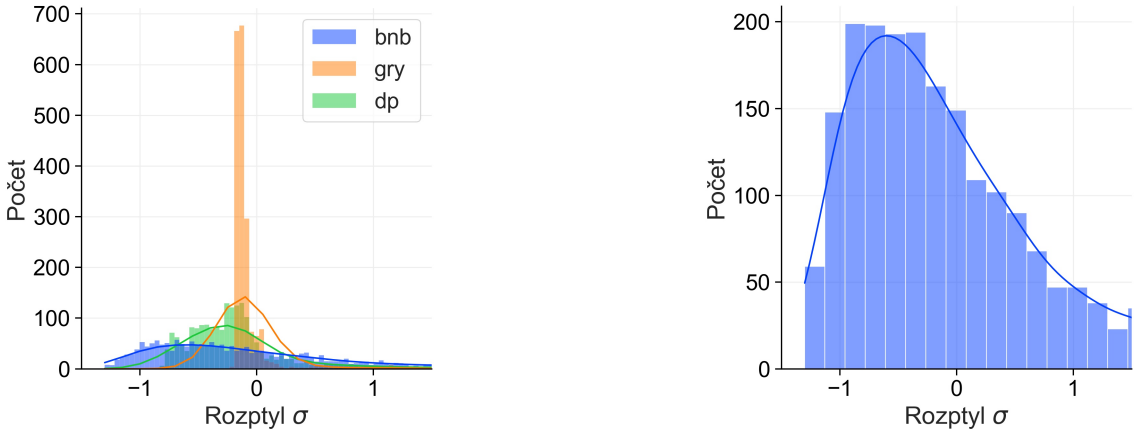
1 initialize W;
2 W[0][0] = 0;
3 W[0][c] = ∞ for c ¿ 0;
4 W[i+1][c] = min(W[i][c], W[i][c - c_{i+1}]+w_{i+1}) for c>=0 and i > 0;
5 Find best price and weight in last column of W table;
6 Construct solution vector from W table;
7 **return** *price, weight, solution vector*

3 Experimenty

Předchozí úkoly se zabývaly výpočetní složitostí jednotlivých algoritmů v závislosti na velikosti instance, tedy dle parametru n. V tomto úkole byl parametr n zafixován na hodnotu 20. Testovací data byla generována, tak že všechny parametry generátoru byly zafixovány a měnila se hodnota pouze zkoumaného parametru. Základní nutné parametry generátoru byly nastaveny na hodnoty $N = 250$, $W = 250$ a $C = 250$. Následná manipulace s ostatními parametry je vždy uvedena před zhodnocením výsledků jednotlivých experimentů. V experimentech se neobjevuje algoritmus hrubé síly, který vždy vyzkouší všechny kombinace. 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.

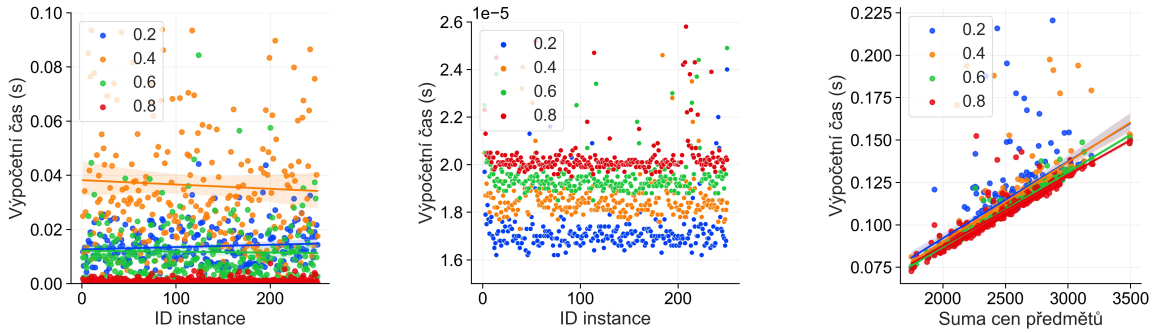
Prvním experimentem bylo testování robustnosti algoritmů. Pro testování robustnosti byl parametr N nastaven na hodnotu 1, a poté pomocí generátoru permutací bylo vygenerováno 2000 permutací. Výpočetní čas nebude nikdy identický, proto byla naměřená data standardizována. Robustnost je poté určena podle vzdálenosti těchto dat od střední hodnoty 0. Předpokládaným výsledkem tohoto experimentu je, že algoritmy DP a GRY jsou robustní narozdíl od B&B, který není robustní na permutace. Z obrázku 1a a 1b lze tyto předpoklady pozorovat. Metoda B&B opravdu není robustní proti permutacím, protože určité permutace předmětů zajistí silnější prořezání stavového prostoru. Naopak hladová heuristika je robustní, protože hned prvním krokem algoritmu je seřadit předměty podle poměru cena/váha. Tedy otázka je spíše zda je řadící algoritmus robustní. Zajímavým pozorováním je dynamické programování, které se nachází mezi B&B a heuristikou. Algoritmus by měl být robustní, protože pokaždé je vyplňována celá tabulka o velikosti $n\times$ suma cen předmětů.

Druhým experimentem, jehož výsledky lze pozorovat na obrázku 2, měl odpovědět na to, zda se výpočetní složitost bude měnit při změně poměru kapacity batohu k sumární váze předmětů u testovaných instancí. Algoritmy byly tes-



(a) (b) B&B

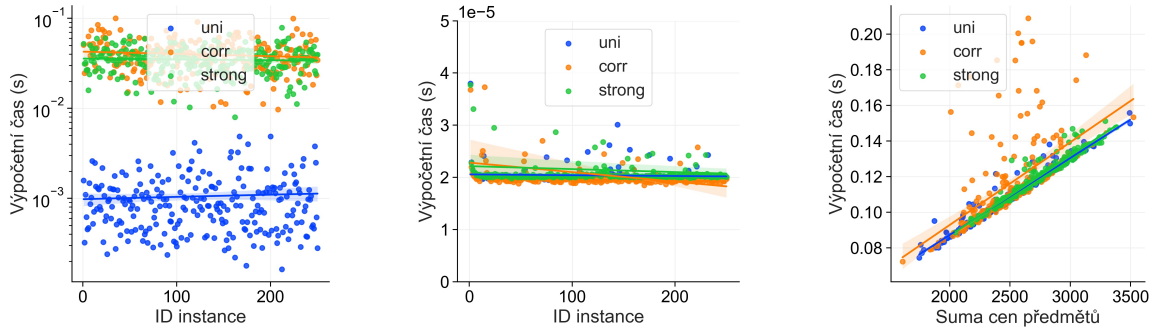
Obrázek 1: Testování robustnosti algoritmů. Výpočetní čas byl standardizován a vizualizován pomocí histogramů.



(a) B&B (b) GRY (c) DP

Obrázek 2: Časová výpočetní složitost algoritmů podle poměru kapacity batohu k celkové váze všech předmětů.

továny pro 4 hodnoty tohoto poměru a to konkrétně pro $\{0.2, 0.4, 0.6, 0.8\}$. Algoritmus B&B si nejlépe vedl pro hodnoty $\{0.2, 0.8\}$, což je způsobeno podmínkami prořezávání. Pro hodnotu 0.8 se algoritmus ihned zanoří hluboko v prohledávacím stromě a přes podmínku zda je možné získat lepší řešení ze zbylých předmětů prořeže většinu zbylých větví. Naopak pro hodnotu 0.2 je stavový prostor prořezáván podmínkou, která se vrací v prohledávání, když je přesažena kapacita batohu. Z pozorování se nachází nejtěžší instance pro tuto metodu kolem hodnoty 0.4. Hla-



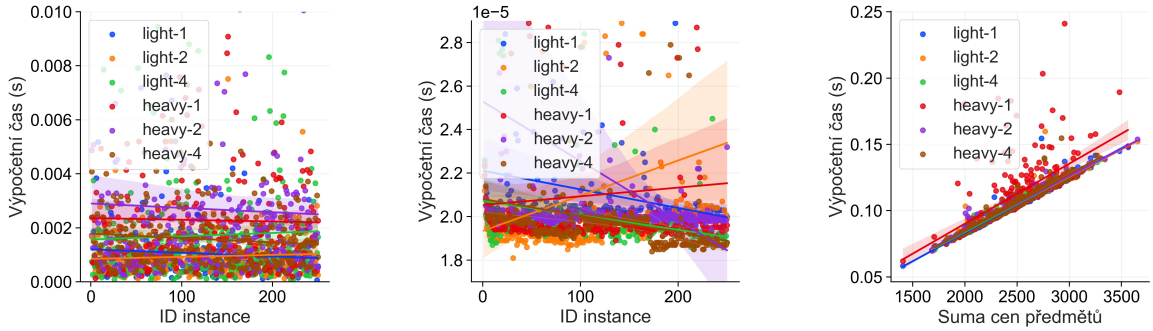
(a) B&B (b) GRY (c) DP

Obrázek 3: Časová výpočetní složitost algoritmů podle korelace ceny k váze předmětů.

dová heuristika na obrázku 2b si vedla nejlépe pro nejmenší velikost batohu. To je zapříčiněno způsobem, jakým tato heuristika postupuje při vytváření výstupní proměnné a celkové ceny. Heuristika vždy seřadí všechny předměty a lineárně všechny projde, a pokud se vejdou do batohu, tak je přidá do výsledku. Toto přidání je rozdíl, který na obrázku 2b můžeme pozorovat. U dynamického programování na obrázku 2c lze stále pozorovat závislost výpočetní složitosti na sumě cen předmětů. Podle předpokladů změna poměru kapacity batohu k předmětům nijak významně výpočetní čas nemění.

Třetí test zkoumal výpočetní závislost algoritmů podle korelace ceny s váhou předmětů. Generátor nabízí 3 nastavení pro tento test a to **uni** - žádná, **corr** - menší, **strong** - silná korelace. Z obrázku 3a si lze všimnout, že algoritmus B&B je řádově rychlejší na instancích, kde není korelace žádná. Při náhodném generování cen a vah je mnohem větší pravděpodobnost, že budou generovány předměty, které jejichž kombinace budou spouštět prořezávající podmínky. Naopak, pokud poměr cena/váha bude téměř stejná pro všechny předměty, tak bude existovat mnohem více kombinací a stavový prostor bude růst rychlostí kombinatorické exploze. Hladová heuristika a metoda využívající dynamické programování jsou podle předpokladů nezasaženy změnami těchto parametrů.

Čtvrtý experiment se zabýval instancema, kde převahovaly lehčí nebo těžší předměty, neboli granularitou. Exponent granularity k byl nastaven na hodnoty z $\{1, 2, 4\}$. Aby exponent měl vliv na generování, musel být ještě nastaven parametr w na hodnotu *light* nebo *heavy*, což mění hustotu pravděpodobnosti mezi $f(\text{váha}) = 1 - \text{váha}^k$ a $f(\text{váha}) = \text{váha}^k$. Výsledky tohoto experimentu lze pozorovat na obrázku 4. Výsledky jsou zcela nepřehledné, jediné, co je možné usoudit, je, že metoda dynamického programování na tyto parametry není citlivá. Pro vyhodnocení citlivosti algoritmů B&B a hladové heuristiky je vhodnější pohled na tabulku 1, ve které je průměrná výpočetní



(a) B&B (b) GRY (c) DP

Obrázek 4: Časová výpočetní složitost algoritmů podle granularity předmětů.

Tabulka 1: Průměrná časová výpočetní složitost algoritmů podle granularity předmětů.

	B&B			GRY			DP		
w \ k	1	2	4	1	2	4	1	2	4
light	1.04e-3	9.42e-4	1.71e-3	2.1e-5	2.1e-5	2.0e-5	0.104	0.104	0.104
heavy	2.28e-3	2.69e-3	1.58e-3	2.2e-5	2.0e-5	2.0e-5	0.113	0.105	0.106

doba přes všechny instance. Z tabulky je patrné, že hladová heuristika je také necitlivá na tento parametr. Výchylky pro tuto heuristiku, podle obrázku 4b jsou tedy s největší pravděpodobností pouze rozdílný počet přidávání předmětů do batohu, tedy postup tvorby výstupních proměnných. Zajímavé výsledky lze pozorovat na algoritmu B&B, kde výrazně težší předměty zkrátily průměrnou výpočetní dobu, tedy podmínka překročení kapacity byla nejspíše splněna častěji k prořezání prohledávacího stromu. Naopak výrazně lehčí předměty tuto podmínku splňovaly méně často a tedy prohledávání bylo delší.

Jelikož hladová heuristika není exaktní algoritmus a nevrací vždy optimální řešení, tak poslední experiment se zaměřil na relativní chybu heuristiky při změně všech parametrů z předchozích experimentů. Instance ověřující robustnost jsou vynechány, jelikož je zřejmé, že chyba bude identická. Z výsledků v tabulce 2 si lze všimnout, že heuristika vrací nejlepší řešení v případě, kdy batoh pojme téměř všechny předměty nebo naopak, když se váha předmětů blíží ke kapacitě batohu, tedy řešení je složeno z mála těžkých předmětů. Nejhůře si heuristika vedla při silné korelaci mezi cenou a vahou předmětů.

4 Závěr

V této práci byly detailněji zkoumány algoritmy B&B, hladová heuristika a dekompozice podle ceny pomocí dynamického pro-

Tabulka 2: Průměrná relativní chyba dle zkoumaných parametrů.

(a) Poměr kapacity batohu k sumě vah předmětů.

m	rel. ch. (%)
0.2	1.62
0.4	0.92
0.6	0.55
0.8	0.27

(b) Korelace ceny s váhou předmětu.

c	rel. ch. (%)
uni	0.27
corr	0.83
strong	5.04

(c) Granularita

	rel. ch. (%)		
w \ k	1	2	4
light	0.29	0.34	0.34
heavy	0.19	0.15	0.10

gramování. Experimenty se zabývaly pozorováním výpočetní složitosti v závislosti na změně parametrů generátoru instancí. Těmito parametry byly permutace instancí, poměr kapacity batohu k sumě vah všech předmětů, korelace mezi cenou a vahou předmětů a granularita.

Z výsledků experimentů je zřejmé, že algoritmus B&B je citlivý na všechny uvedené parametry. Naopak hladová heuristika a metoda dynamického programování nevykazovaly žádné změny při hýbání s parametry.

U hladové heuristiky byla navíc pozorována relativní chyba, jelikož nezaručuje vždy optimální řešení. Nejmenší chyba byla, pokud součástí řešení byly téměř všechny předměty nebo jen málo těžkých. Nejhůře dopadla heuristika na instancích, kde byla silná korelace mezi cenou a váhou předmětu.