1 Zadání úlohy

Celé zadání se skládá ze čtyřech částí:

1. Prozkoumejte citlivost metod řešení problému batohu na parametry instancí generovaných generátorem náhodných instancí. Máte-li podezření na další závislosti, modifikujte zdrojový tvar generátoru.

2. Na základě zjištění navrhněte a proveďte experimentální vyhodnocení kvality řešení a výpočetní náročnosti.

3. Zkoumejte zejména následující metody:

   - hrubá síla (pokud z implementace není evidentní úplná necitlivost na vlastnosti instancí)
   - metoda větví a hranic, případně ve více variantách
   - dynamické programování (dekompozice podle ceny a/nebo hmotnosti). FPTAS algoritmus není nutné testovat, pouze pokud by bylo podezření na jiné chování, než DP
   - heuristika - poměr cena/váha

4. Pozorujte zejména závislosti výpočetního času (případně počtu testovaných stavů) a rel. chyby (v případě heuristiky) na:

   - maximální váze věcí
   - maximální ceně věcí
   - poměru kapacity batohu k sumární váze
   - granularitě (pozor - zde si uvědomte smysl exponentu granularity)

2 Rozbor možných variant řešení

Pro exprimentální hodnocení algortimů jsem vypozoroval dva možné přístupy:

1. pro všechny zkoumané algoritmy provést sadu měření zaležených na instancích vytvořených generátorem, u kterého bych systematicky měnil vždy jednu proměnnou;

2. odhadnout možnou citlivost algoritmů na různě generované instance a tyto odhady experimentálně ověřit.

3 Popis postupu řešení

Jako postup řešení jsem vybral způsob odhadnutí citlivosti algoritmu na podobu vstupní instance. U všech algoritmů se zaměřím na závislosti vstupních dat na výpočetním čase a u heuristiky podle poměru cena/váha navíc určím citlivost

relativní chyby na vstupních datech. U všech metod vím, že jsou citlivé na velikosti vstupní instance jak již bylo ukázáno předchozích úlohách. Proto se budu v této práci zabývat jinými charakteristikami generovaných vstupních dat.V této kapitole postupně rozeberu postup zkoumání jednotlivých algoritmů.

3.1 Hrubá síla

U hrubé síly můžeme odpozorovat, že pokaždé projde všechny možné kombinace výskytu věcí v batohu. Z toho vyplývá, že je metoda citlivá jen na počet vkládaných věcí. Doba výpočtu je exponenciální.

3.2 Metoda větví a hranic

Není tajemstvím, že tato metoda je vylepšenou verzí hrubé síly. V nejhorším případě může být časová složitost dokonce ekvivalentní s hrubou sílou. Rychlost výpočtu závisí na počtu vkládaných věcí a na počtu ořezaných větví:

- **Ořezání zdola:** Větve se ořezávají zdola pokud maximální možný součet cen aktuálně generované větve je menší nežli dočasně maximální nalezená cena. Pokud se tedy budou nejdražší věci vyskytovat co možná nejblíže ke kořenu generovaného stromu, algoritmus bude moci ořezat více větví.
- **Ořezání shora:** S rostoucím poměrem kapacity batohu na sumární váze se bude snižovat pravděpodobnost ořezávání shora, neboť se více předmětů vejde do batohu. Dalším parametrem ja granularita. Čím více velkých věcí bude batoh obsahovat, tím více větví se odřízne.

3.3 Dynamické programování (dekompozice podle ceny)

Metoda dynamického programování při dekompozici podle ceny se zakládá na vytvoření tabulky o velikosti $(n + 1)C$, kde C je součet cen všech věcí. Tato metoda je tedy citlivá na součet cen veškerých věcí.

3.4 Dynamické programování (dekompozice podle váhy)

Podobně jako u dekompozice podle ceny je podstatnou částí výpočtu konstrukce tabulka. Tetokrát ovšem velikosti $(n + 1)M$, kde M je nosnost batohu.

3.5 Heuristika (poměr cena/váha)

Tato metoda vytvoří pole poměrů cen jednotlivých věcí k jejich vahám. Poté toto pole seřadí podle velikosti od největšího poměru po ten nejmenší a začne postupně vkládat věci do batohu dokud není batoh plný. Jelikož tato metoda neposkytuje exaktní výsledky zaměřím se nejenom na zkoumání výpočetního času, ale také na výpočet relativní chyby.

3.5.1 Výpočetní čas

Na první pohled se zdá, že výpočetní čas nebude závislý na charakteristikách generátoru jiných nežli je velikost instance.

3.5.2 Výpočet relativní chyby

S relativní chybou tomu bude nicméně jinak. Charakteristiky generovaných instancí jako jsou poměr nosnosti na celkové váze věcí, maximální hmotnost věcím maximální cena věcí nebo granularita mohou ovlivňovat relativní chybu, nicméně zatím nevím přesně jak.

4 Kostra algotritmu

V analytické části jsem měl podezření na možnou závislost B&B na uspořádání vygenerovaných instancí. Proto jsem přidal nový nepovinný parametr (sort) do generátoru instancí. Tento parametr může nabývat následujících hodnot:

- sort=-1 (instance seřazené podle ceny vzestupně),
- sort=1 (instance seřazené podle ceny sestupně).

Veškeré algoritmy mám již naimplementované z předchozích úkolů. Pro každou zkoumanou metodu jsem vytvořil skript, který prozkoumá danou metodu na zvolenou charakteristiku generovaných instancí a výsledky zapíše do souboru.

4.1 Výpočet relativní chyby

Pro měření relativní chyby jsem vytvořil separátní modul, který má na vstupu dva parametry:

1. soubor s naměřenými výsledky,
2. soubor se správnými výsledky.

Algoritmus přečte oba soubory a vytvoří pole relativních chyb ($\frac{opt(n)-apx(n)}{opt(n)}$) veškerých instancí obsažených v souboru. Následně vypočítá průměrnou relativní chybu, maximální relativní chybu a tyto hodnoty zapíše do souboru. Pro spouštění modulu se všemi soubory jsem vytvořil skript.

4.2 Měření času

Abych co možná nejlépe změřil čas, který algoritmus stráví na CPU, použil jsem java třídu ThreadMXBean. Pokud chci změřit čas, dám modulu na vstupu parametr kolikrát chci daný algoritmus zopakovat. Po každé iteraci změřený čas zapíšu do pole a po skončení všech iterací hodnoty v poli zprůměruji. Výsledek zapíši do souboru.

5 Naměřené výsledky

Pro implementaci jsem využil programovací jazyk Java. Testy běžely na operačním systému macOS. Výsledky jsou měřené na instancemi tvořenými generátorem, který pokud není specifikováno jinak má parametry:

- N=20;
- n=20;
- W=300;
- C=300;
- m=0.5;
- k=1;
- d=0;
- sort=0;

5.1 Hrubá síla

Na základě analýzy jsem testoval závislost hrubé sílý na velikosti instancí. Na základě tabulky 1 je zřejmé, že má metoda exponenciální složitost.

5.2 Metoda větví a hranic

Podle naměřených hodnot 2 můžu potvrdit, že výpočetní rychlost metody B&B závisí na poměru nosnosti batohu a maximální váhy veškerých věcí tak, že tím je tento poměr vyšší tím stoupá i doba běhu, neboť se vejde do batohu více věcí.

Čím více dražších věcí se nachází na začátku seznamu tím se odřeze více větví a tím je metoda rychlejší viz tabulka 3.

Závislost na granularitě věcí je zřejmá 1. S klesající pravděpodobností, že velké resp. malé věci jsou v batohu roste výpočetní doba.

5.3 Dynamické programování (dekompozice podle ceny)

Z naměřených hodnot 2 lze poznat, že čím větší je maximální cena věci a velikost instance, tím roste výpočetní doba. Je tomu tak díky citlivosti dynamického programováni s dekompozicí podle ceny na celkévě sumě vkládaných věcí. Celková suma roste s rostoucí maximální cenou vkládaných věcí.

5.4 Dynamické programování (dekompozice podle váhy)

Metoda dynamického programování při dekompozici podle váhy je ciltivá na nosnosti batohu a velikost instance. V mém případě můžu tuto charakteristiku instance rozdělit na citlivost na poměru nosnosti batohu a maximální váze veškerých věcí a na hodnotě maximální váze věcí. Tato metoda je opravdu závislá na těchto charakteristikách, což vyplývá z grafů 4 a 3.

5.5 Heuristika (poměr cena/váha)

5.5.1 Výpočetní čas

Z tabulky 4 je zřejmá závislost na velikosti instance.

5.5.2 Výpočet relativní chyby

Pro získání správných výsledků první na vygenerované instance pustím metodu B&B a potom až heuristiku. Z výsledných dat poté spočítám relativní chybu. Zkoumané charakteristiky:

- **Granularita** - z naměřených dat v tabulce 5 vyplývá, že čím více je v batohu velkých věcí, tím je relativní chyba menší.
- **Poměr nosnosti na sumární váze všech věcí** - z grafu 5 vyplývá, že relativní chyba je citlivá na poměr nosnosti batohu k sumární váze věcí,
- **Maximální váha** - z grafu 6 nelze vypozorovat, žádná přímočará závislost relativní chyby na maximální váze,
- **Maximální cena** - z grafu 7 nelze vypozorovat, žádná přímočará závislost relativní chyby na maximální ceně.

6 Závěr

Vypracovaná analýza se ukázala jako pravdivá. Veškeré metody jsou závislé na velikosti instance. Metoda větví a hranic je navíc závislá na poměru nosnosti batohu a sumární váhy věcí, granularitě a seřazenosti vstupních dat podle ceny. Dynamické programování dle dekompozice podle ceny je závislé na maximální hodnotě věci a dynamické programování dle dekompozice podle váhy je závislé na nosnosti batohu (maximální váze věci a poměru nosnosti batohu a sumární váhy věcí). Relativní chyba výsledku u heuristiky se ukázala závislá na poměru nosnosti batohu/sumární váha všech věcí a granularitě vkládaných věcí.

7 Přílohy

Počet položek	Hrubá síla
4	1,84 ms
10	2,231 ms
15	103,65 ms
20	6,22 s
22	20,87 s
25	4,91 min
27	22,74 min
30	1,02 h

Table 1: Srovnání doby běhu hrubé sílý

Poměr nosnosti a sumární váze věcí	Doba běhu
0.1	0,00339 s
0.2	0,00901 s
0.3	0,01847 s
0.4	0,03128 s
0.5	0,03509 s
0.6	0,04585 s
0.7	0,04566 s
0.8	0,04712 s
0.9	0,04632 s

Table 2: Závislost doby běhu algoritmu B&B na poměru nosnosti batohu a sumární váze všech věcí

Velikost instance	Dražší věcí na konci	Dražší věci na začátku	Náhodné uspořádání
5	0,00084 s	0,0015 s	0,00109 s
10	0,00205 s	0,00335 s	0,00296 s
15	0,00501 s	0,00771 s	0,00492 s
20	0,01788 s	0,07896 s	0,02975 s

Table 3: Závislost doby běhu algoritmu B&B na uspořádání vstupních věcí podle ceny

Velikost instance	Doba běhu
10	0.00096 s
20	0.00146 s
30	0.00167 s
40	0.00254 s
50	0.00266 s
60	0.00309 s
70	0.00386 s
80	0.00437 s
90	0.00485 s

Table 4: Závislost doby běhu algoritmu heuristika podle poměru cena/váha na velikosti instance

Parametr k	Více malých věcí	Více velkých věcí
0,1	0,62%	0,21%
0,2	0,62%	0,25%
0,3	0,55%	0,34%
0,4	0,53%	0,40%
0,5	0,52%	0,62%
0,6	0,47%	0,64%
0,7	0,47%	0,65%
0,8	0,44%	0,70%
0,9	0,43%	0,72%

Table 5: Závislost relativní chyby algoritmu heuristika podle poměru cena/váha na granularitě

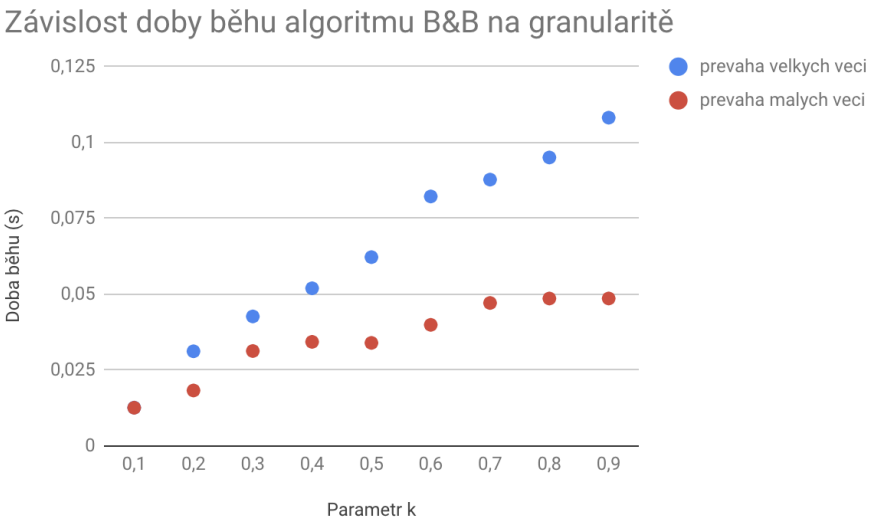

Figure 1: Srovnání doby běhu algoritmu B&B na granularitě

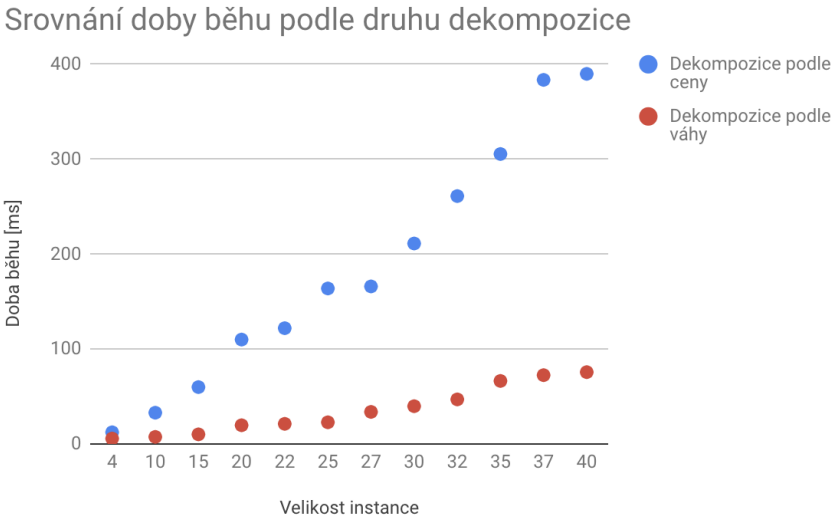

Figure 2: Závislost doby běhu dynamického programování (dekompozice dle ceny) na maximální ceně pro jednotlivé velikosti instancí

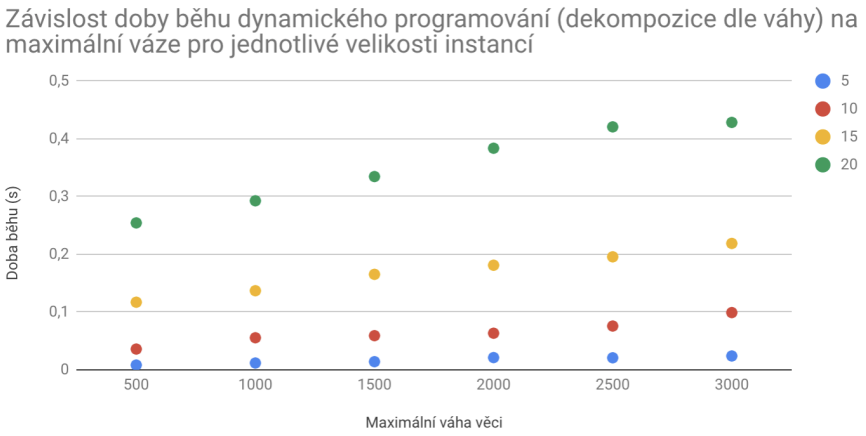

Figure 3: Závislost doby běhu dynamického programování (dekompozice dle váhy) na maximální váze pro jednotlivé velikosti instancí

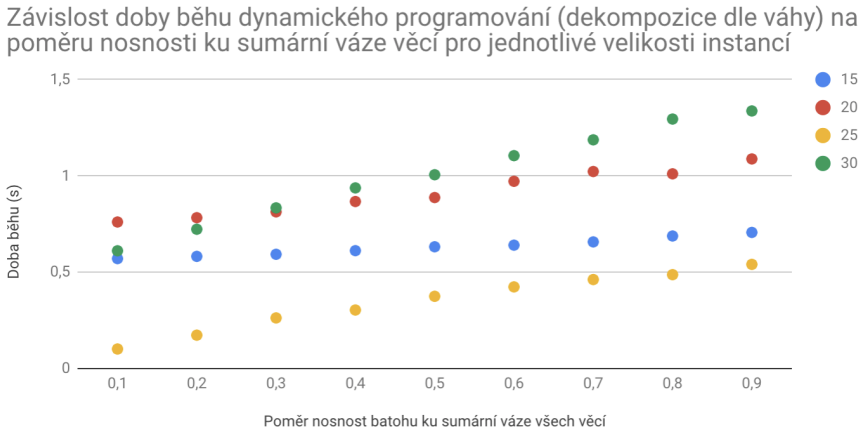

Figure 4: Závislost doby běhu dynamického programování (dekompozice dle váhy) na poměru nosnosti ku sumární váze věcí pro jednotlivé velikosti instancí

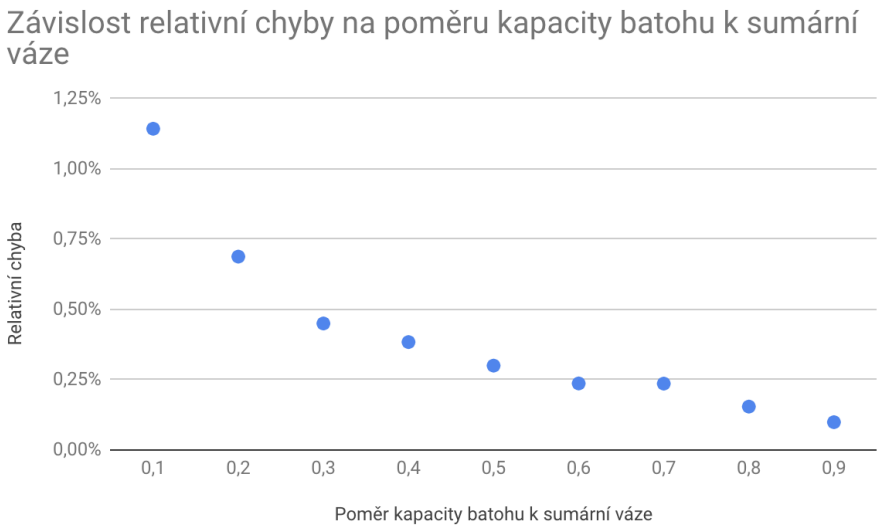

Figure 5: Závislost relativní chyby na poměru kapacity batohu k sumární váze

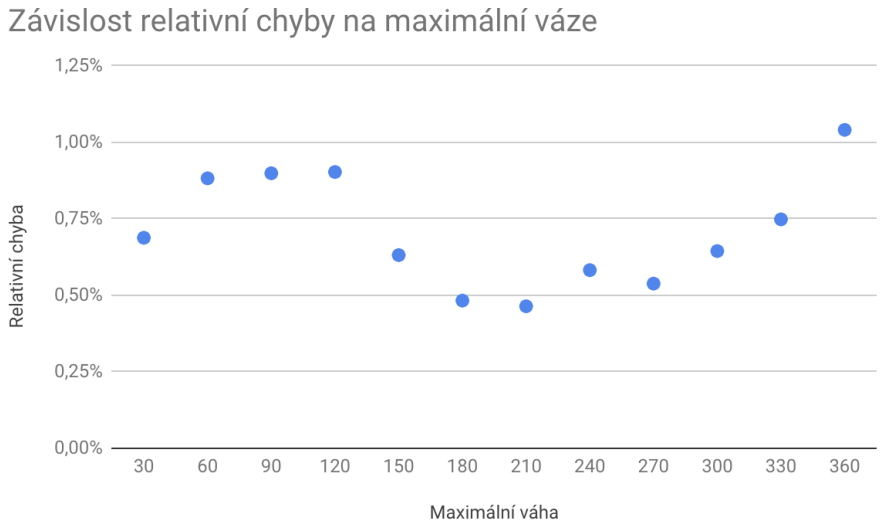

Figure 6: Závislost relativní chyby na maximální váze

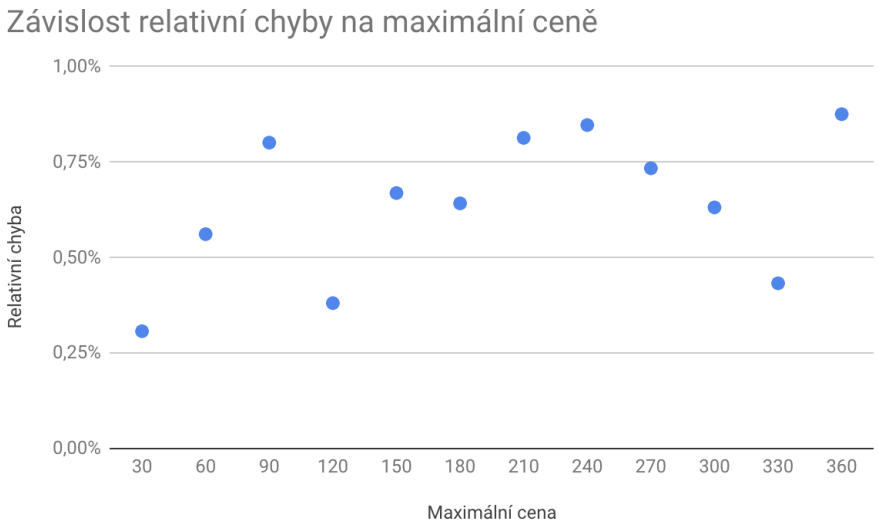

Figure 7: Závislost relativní chyby na maximální ceně