KnapsackSwift - 2nd report

- Knapsack problem solver written in Swift 4.0
- MI-PAA semestral project @ CTU FIT university
- Below is the report for the 2nd part (in Czech language)
- All measurements was made on MacBook Pro (13-inch Mid 2017) (CPU: Intel Core i5-7360U, RAM: 16 GB, OS: macOS High Sierra)
- Author: Petr Chmelar
- Date: 11/11/2017

Úloha

- Cílem úlohy je implementace a analýza řešení problému batohu metodami větví a hranic, dynamickým programováním a aproximativním algoritmem FPTAS.

Popis řešení

Metoda větví a hranic

- Metoda solveBranchAndBound vychází z rekurzivní metody solveBrute-Force (viz report1). Má navíc čtvrtý vstupní parametr reprezentující nejlepší zatím dosažené řešení. Při větvení rekurze je pak vždy provedena kontrola, zda u nově vytvářené větve je potenciál pro nalezení lepšího řešení než nejlepšího zatím dosaženého. V negativním případě je tato větěv odříznuta. Metoda větví a hranic výpočet často značně urychlí, ale asymptotická složitost zůstavá 2^n.

Metoda dynamického programování

- Tato metoda může využívat dekompozici problému podle ceny, nebo podle váhy. Vzhledem k následné implementaci FPTAS algoritmu jsem zvolil dekompozici podle ceny. Řešení podproblémů jsou v tomto případě uložené v tabulce o rozměrech n (počet věcí) x maxValue (suma cen všech věcí) - asymptotická složitost je n * maxValue.
- Algoritmus začíná ve sloupci 0 a postupně plní tabulku doprava dle následujících pravidel:

```
table[0][0] = 0
table[0][j] = Int.max ... pro všechna j > 0
table[i][j] = min(table[i-1][j], table[i-1][j-items[i-1].value] + items[i-1].weight) ..
```

- Výsledné řešení poté nalezneme v posledním sloupci, jeho hodnota je menší než maxWeight a má maximální index v rámci sloupce.

Aproximativní algoritmus FPTAS

- Zde využijeme metody dynamického programování nad zjednodušeným modelem. Zjednodušení dosáhneme vydělením cen konstantou, čímž se sníží hodnota maxValue (suma cen všech věcí) a tedy i složitost metody dynamického programování.
- Pro zvolenou maximální relativní chybu epsilon má konstanta hodnotu:

  ```
  coeff = (eps * maxValue) / n
  ```

- Ceny modifikujeme tak, že je vydělíme získanou konstantou:

  ```
  item.value = floor(item.value / coeff)
  ```

- Pro spočítání relativní chyby algoritmu jsou potom použité původní ceny řešení, které vzniklo z upravených cen.

Naměřené výsledky

Průměrný čas výpočtu [s]

Počet instancí	BruteForce	BranchAndBound	Decomposition
4	4.72E-05	4.52E-05	4.70E-05
10	0.002212371826	0.000564956665	0.001222410202
15	0.0642175436	0.002381997108	0.0112460804
20	2.078239183	0.007297010422	0.02949497223
22	8.327893114	0.01334373951	0.03868565559
25	65.13239241	0.02243446827	0.0573835659
27	NaN	0.03353640556	0.07064874649
30	NaN	0.0672105217	0.09264957905
32	NaN	0.1134650469	0.1066818571
35	NaN	0.2375982618	0.1345773458
37	NaN	0.3127718019	0.1523378181
40	NaN	0.8838491535	0.1812687922

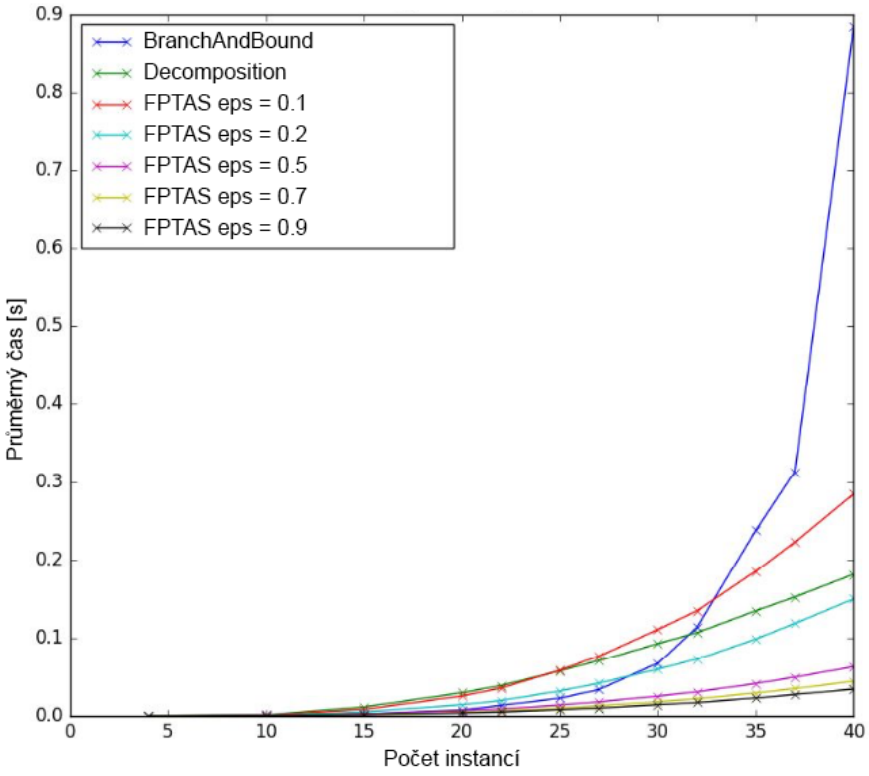
Počet instancí	FPTAS eps = 0.1	FPTAS eps = 0.2	FPTAS eps = 0.5	FPTAS eps = 0.7	FPTAS eps = 0.9
4	4.05E-05	3.84E-05	3.79E-05	3.59E-05	3.44E-05
10	0.0010044431690	0.0008202505112	0.0005429220200	0.0004427385330	0.0003872442245
15	0.0081525802610	0.0048973989490	0.0023483514790	0.0017423915860	0.0014000368120

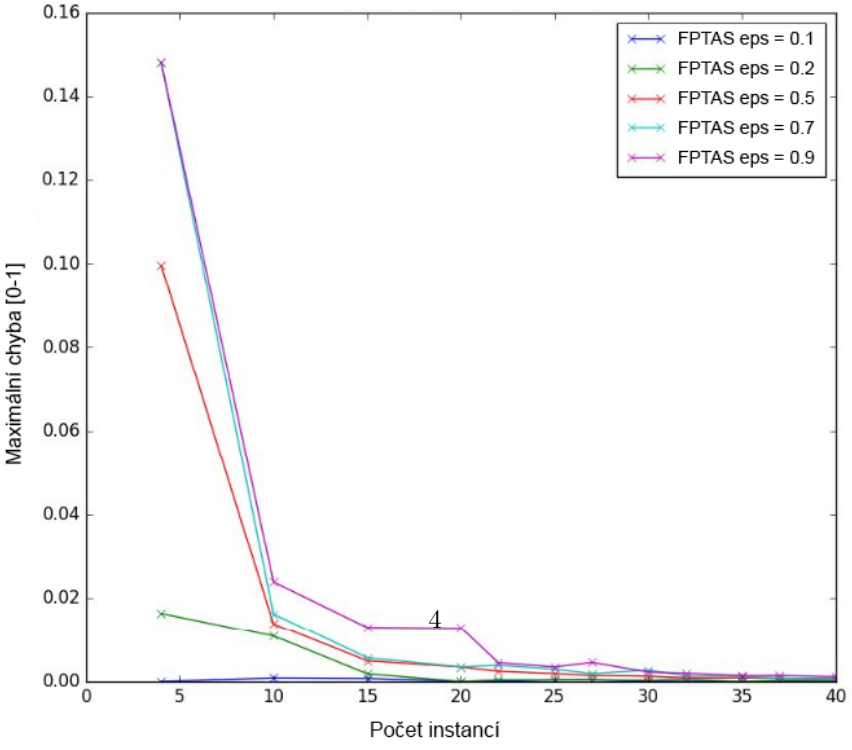
Počet instancí	FPTAS eps = 0.1	FPTAS eps = 0.2	FPTAS eps = 0.5	FPTAS eps = 0.7	FPTAS eps = 0.9
20	0.025297040941	0.014144492150	0.006385917664	0.004600334167	0.003597226143
22	0.0354944849	0.019537854190	0.008720974922	0.006270194054	0.004920625687
25	0.058133144380	0.031740951540	0.0137139225	0.009780721664	0.007627558708
27	0.076524939540	0.041728386880	0.017885593670	0.012747621540	0.009964566231
30	0.1106373072	0.059579877850	0.025159983630	0.017955336570	0.01411275387
32	0.1346078348	0.0725420332	0.030648012160	0.021741542820	0.01695840359
35	0.1852054739	0.098466458320	0.041281695370	0.029229779240	0.0226640749
37	0.2228449392	0.1184925175	0.049410939220	0.034810676570	0.02721654415
40	0.2844247484	0.1498497629	0.062735228540	0.044176559450	0.03418153286

FPTAS maximální chyba [0-1]

Počet instancí	FPTAS eps = 0.1	FPTAS eps = 0.2	FPTAS eps = 0.5	FPTAS eps = 0.7	FPTAS eps = 0.9
4	0	0.016304347830	0.099567099570	0.148148148100	0.148148148100
10	0.000836820080	0.010810810810	0.013761467890	0.016129032260	0.023758099350
15	0.000724112960	0.001833740830	0.0049170252	0.005652911240	0.012935883010
20	0	0	0.003426124190	0.003426124190	0.012767897860
22	0	0.000390472470	0.002416626390	0.003842459170	0.004456327980
25	0.000369003690	0.000390472470	0.001875468860	0.002927186240	0.003426791270
27	0.000394788780	0.000394788780	0.001426024950	0.001851851850	0.004507628290
30	0	0.000275938180	0.001247660630	0.002698145020	0.002228274320
32	0	0.000519345620	0.000867302680	0.001304801670	0.001965739960
35	0	0	0.000875912400	0.001168565580	0.001393728220
37	0	0.000283848990	0.000804073970	0.000831255190	0.001408450700
40	0	0.000218007410	0.000643776820	0.000643776820	0.001116320600

Grafy


BranchAndBound
Decomposition
FPTAS eps = 0.1
FPTAS eps = 0.2
FPTAS eps = 0.5
FPTAS eps = 0.7
FPTAS eps = 0.9
Průměrný čas [s]
Počet instancí


FPTAS eps = 0.1
FPTAS eps = 0.2
FPTAS eps = 0.5
FPTAS eps = 0.7
FPTAS eps = 0.9
Maximální chyba [0-1]
Počet instancí

Závěr

- Z grafu průměrných časů vypočtů je vidět, že pro větší instance je nejrychlejším přesným algoritmem dynamické programování.
- Maximální chyba algoritmu FPTAS je v praxi daleko menší, než teoretická maximální průměrná cena, přičemž výpočetní čas je pro větší epsilon násobně nižší než výpočetní čas dynamického programování.