Jan Klicpera

NI-KOP – Domácí Úkol 2

1. Specifikace úlohy

Detailní specifikace je dostupná na https://moodle-vyuka.cvut.cz/mod/assign/view.php?id=89697.
Úkolem je implementovat řešení konstruktivní verze problému batohu:

- Hrubou silou
- Metodou větví a hranic
- Metodou dynamického programování
- Jednoduchou greedy heuristikou
- Redux verzí greedy heuristiky
- FPTAS algoritmem

Na implementovaných verzích řešiče mají být následně spuštěny poskytnuté vstupní instance a porovnány průměrné doby běhu jednotlivých algoritmů. U aproximativních heuristik má být také porovnána naměřená relativní chybovost vzhledem k velikosti instance.

2. Implementace řešení

Pro implementaci byl zvolen jazyk C#, výstupem je konzolová aplikace. Aplikace je plně konfigurovatelná za pomocí přepínačů, chybné vstupy jsou ošetřeny.

První dva algoritmy jsou implementovány stejným principem jako u předchozí úlohy, nebudu je zde proto detailněji zmiňovat. Čas běhu byl měřen pomocí systémové třídy *Stopwatch*, která umožňuje velmi přesně měřit výpočetní čas. Při měření času je každá instance spouštěna pětkrát, výsledným časem je průměr těchto pěti běhů.

Výstupem pro každou zpracovanou instanci je také hodnota relativní chyby ε:

$$\varepsilon = \frac{\left|\mathrm{C}\left(\mathrm{APR(I)}\right) - C\left(OPT(I)\right)\right|}{\max\left(\mathrm{C}\left(\mathrm{OPT(I)}\right), \mathrm{C}\left(\mathrm{APR(I)}\right)\right)}$$

- $C(S)$ – hodnota optimalizačního kritéria řešení S (v tomto případě tedy cena)
- $APR(I)$ – aproximální řešení instance I
- $OPT(I)$ – optimální řešení instance I

Optimální řešení instance jsou načítána z poskytnutých řešení pro testovací sady, hodnota tedy slouží i jako kontrola správnosti algoritmu pro neaproximativní algoritmy (měla by být vždy nulová).

Metoda dynamického programování

Tento postup je založen na dekompozici problému na menší podproblémy a memoizaci výsledků těchto podproblémů – ukládání výsledků do tabulky. Každý podproblém je tedy zapotřebí počítat pouze jednou, což dokáže výrazně zrychlit rychlost výpočtu, pokud se dekomponované podproblémy „překrývají“. Nevýhodou tohoto postupu je vysoká paměťová náročnost. Problém batohu lze dekomponovat několika způsoby, implementovány byly dekompozice podle ceny a podle váhy. Implementován byl také postup, který vybere v závislosti na vstupní instanci dekompozici, která bude pro dané hodnoty méně časově a paměťově náročná.

Oba postupy používají pro uložení mezivýsledků 2D pole. Dekompozice podle ceny používá tabulku o rozměrech $C * n$, kde C je součet cen všech předmětů a n je počet předmětů. Každá buňka tabulky se souřadnicemi $(x, y) \mid 0 \leq x < n, 0 \leq y \leq C$ reprezentuje váhu, které lze dosáhnou v pod-instancí obsahující prvních $x + 1$ předmětů a mající celkovou cenou y. Algoritmus tyto buňky vyplňuje „*bottom-up*", začíná tedy na triviálních instancích o jednom předmětu a postupně řeší větší podproblémy. Pokud pro dané souřadnice existuje více možných vah předmětů (dojde ke kolizi), pak se vždy vybere váha, která je menší. Pokud váha překročí kapacitu batohu, pak se výsledek nezapisuje do tabulky. Optimální hodnota pro řešenou instanci se nachází v posledním sloupci – vybere se vyplněná buňka s maximální y-souřadnicí (cenou předmětů).

Dekompozice podle váhy je velmi podobná, hlavní rozdíl je ve způsobu uložení výsledků v tabulce. Místo ceny předmětů zde reprezentuje y osa váhu předmětů. Jednotlivé buňky poté reprezentují cenu, které lze dosáhnout v pod-instanci obsahujících prvních $x + 1$ předmětů s celkovou váhou y. Byla implementována také varianta, která na základě vstupní instance vybere vhodnější verzi dekompozice, tedy verzi, která vyžaduje menší tabulku.

Výpočetní i paměťová složitost tohoto algoritmu je $\mathrm{O(C * n)}$, resp. $\mathrm{O(W * n)}$. Hodnoty C a W však nejsou závislé na velikosti vstupní instance, algoritmus je proto pouze pseudopolynomiální.

Greedy heuristika

Tato metoda seřadí předměty sestupně podle jejich poměru cena/hmotnost. Následně v tomto pořadí přidává předměty do batohu, dokud zbývá místo na další předmět. Výpočetní složitost se odvíjí od zvoleného algoritmu pro řazení, nejlépe tedy $O(n\,log(n))$. Není však zaručeno, že výsledná konfigurace předmětů bude optimální, metoda je tedy pouze aproximativní. Relativní chyba algoritmu navíc není nijak shora omezena.

Tuto heuristiku lze mírně vylepšit tím, že se zároveň pokusíme přidat pouze jeden nejdražší předmět (který se vejde do batohu) a jako výsledek označíme konfiguraci s větší celkovou cenou.

FPTAS (Fully Polynomial Time Approximation Scheme)

Toto aproximační schéma bylo implementováno za pomocí dekompozice dle ceny. Algoritmus na vstupu přijímá hodnotu ε, která stanovuje maximální relativní chybu, která by mohla nastat pro libovolnou vstupní instanci. Na této hodnotě je také přímo závislá výpočetní složitost algoritmu. Hlavní myšlenkou algoritmu je zmenšení všech cen předmětů na základě stanovené přesnosti. Každá cena předmětu se tedy vydělí koeficientem, který je spočten následovně:

$$K = \frac{\varepsilon\, C_M}{n}$$

- C_M – cena nejdražšího předmětu z instance
- $0 < \varepsilon \leq 1$

Takto upravená instance se poté vyřeší pomocí klasického DP algoritmu s dekompozicí dle ceny. Jelikož je výpočetní složitost DP algoritmu závislá na celkové ceně všech předmětů, tak bude výpočetní složitost klesat s volbou vyššího ε. Výsledná cena se získá vynásobením vektoru vybraných předmětů původní cenou těchto vybraných předmětů. Jelikož se ve vstupní instanci mohou objevit předměty s neúměrně velkými hodnotami, které by negativně mohly ovlivnit přesnost, tak se na začátku algoritmu provádí kontrola, která z instance dočasně odstraní všechny předměty, jejichž váha překračuje kapacitu batohu.

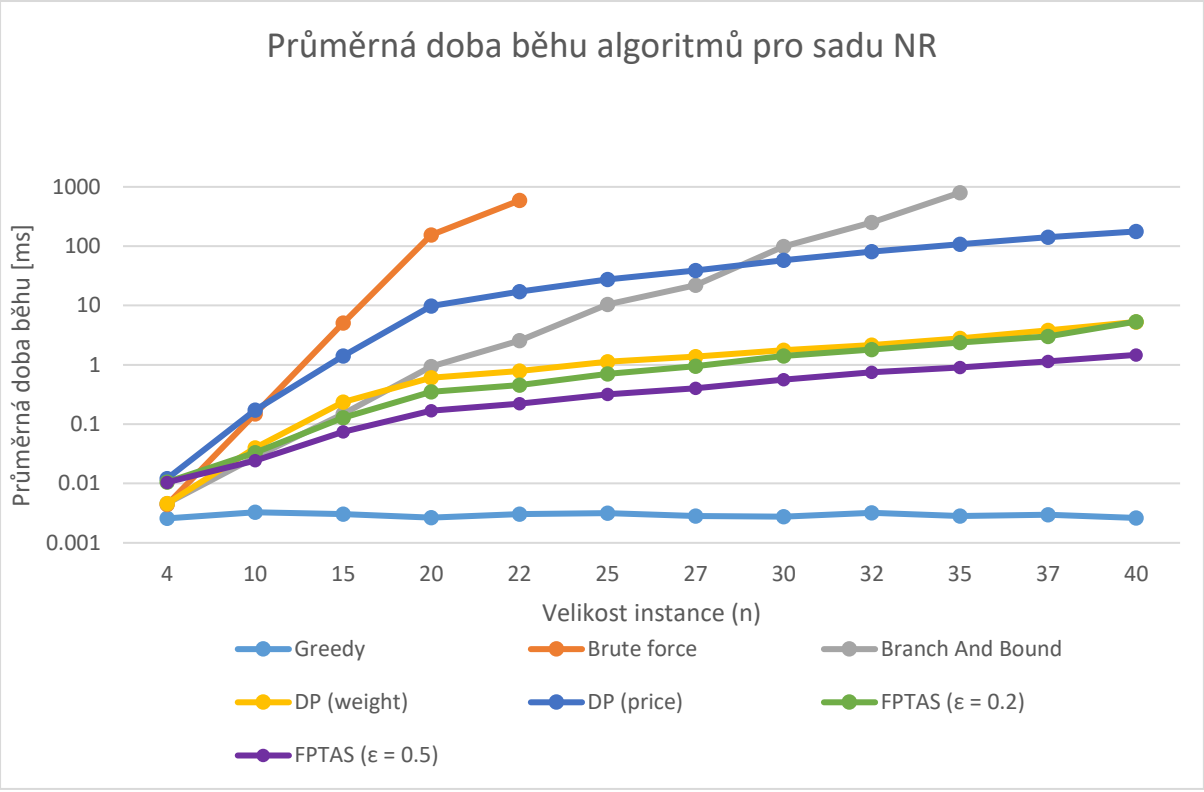
3. Naměřené výsledky

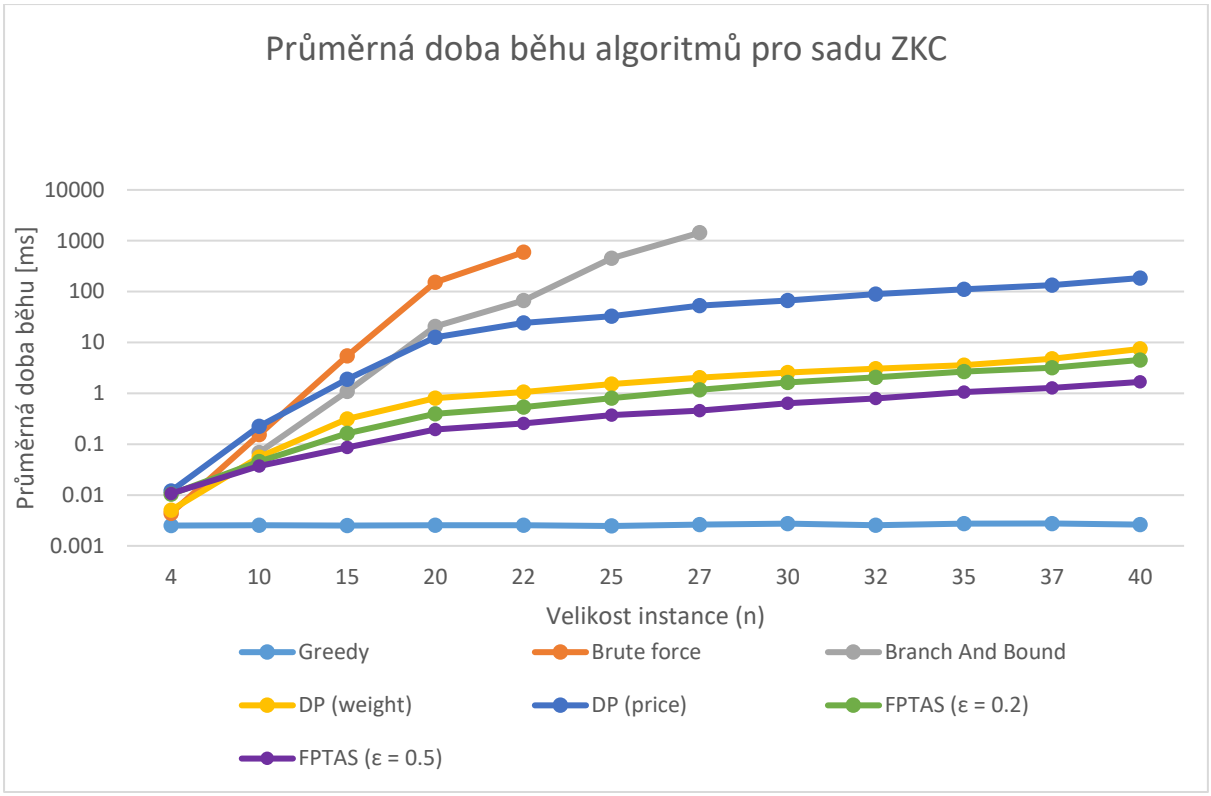
Výsledky byly naměřeny v OS Windows 10 64bit s procesorem AMD Ryzen 5 2600 six-core (3.4 GHz). Běh nebyl paralelizován, bylo tedy vždy využíváno pouze jedno jádro procesoru.

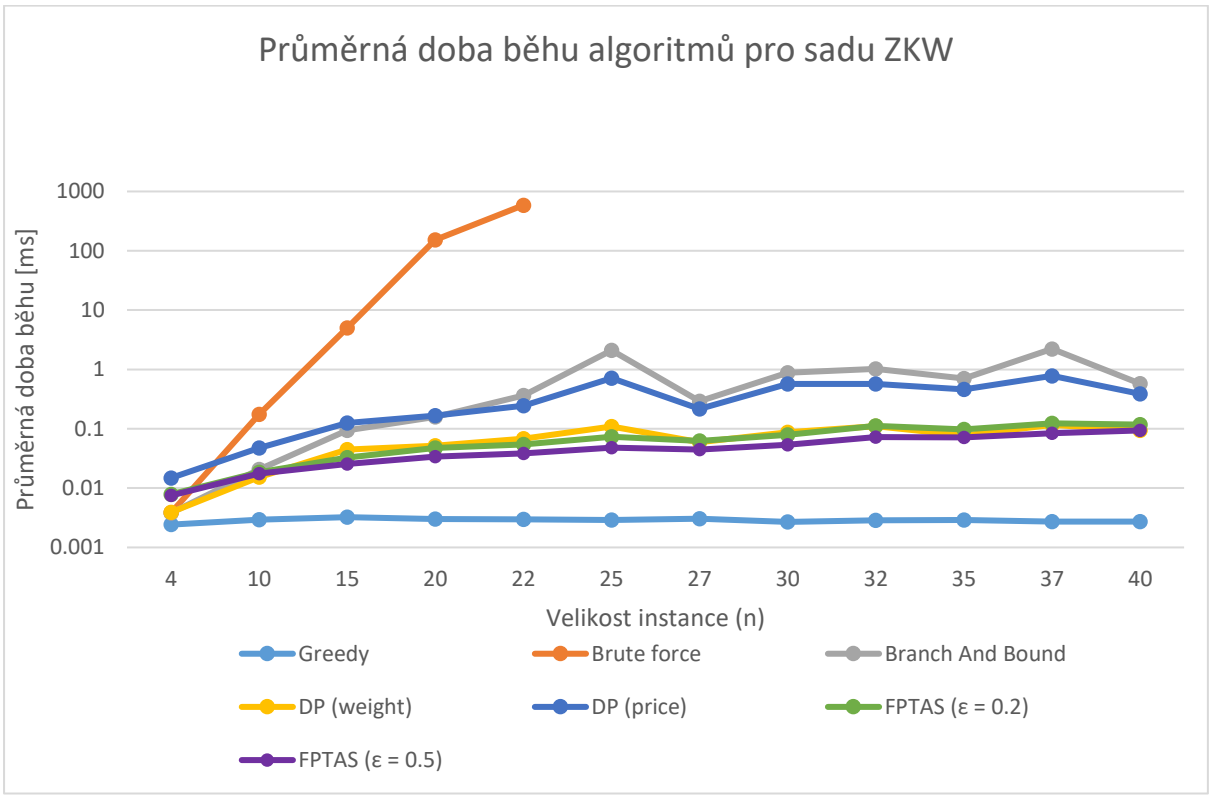
Pro lepší přehlednost byly body v grafu spojeny úsečkou, naměřené hodnoty jsou však samozřejmě diskrétní.

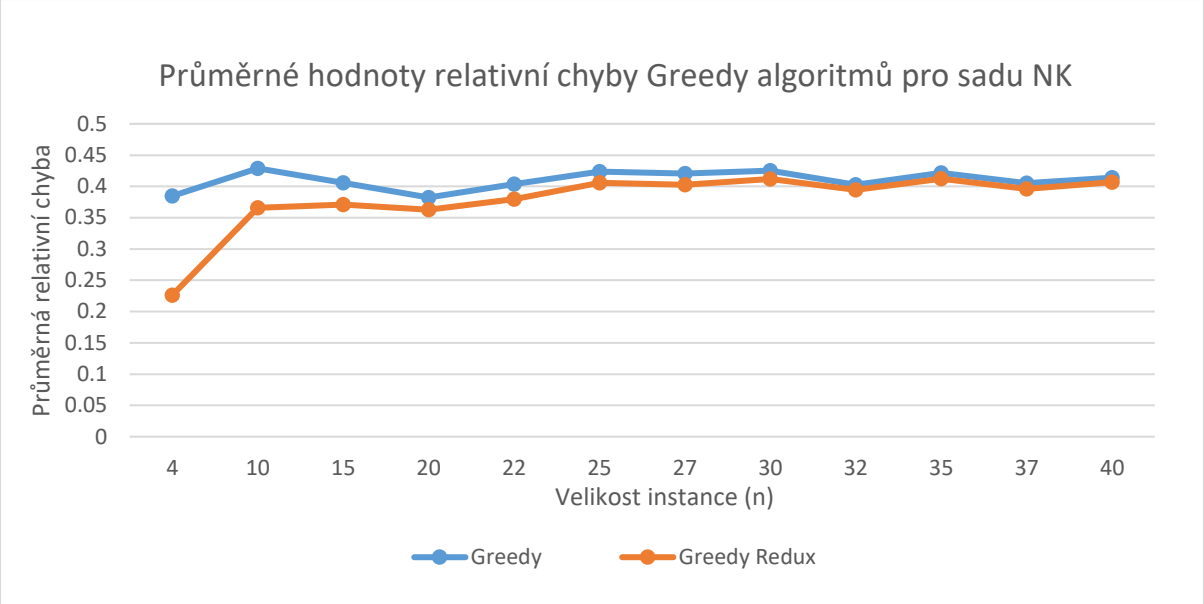
V grafu doby běhu byl vynechán Greedy redux algoritmus, protože je jeho průměrná i maximální doba téměř totožná s dobou běhu Greedy algoritmu. Taktéž byla vynechána doba běhu DP algoritmu, který se rozhodoval mezi cenou a váhou (hodnoty byly téměř totožné s DP dekompozicí podle váhy).

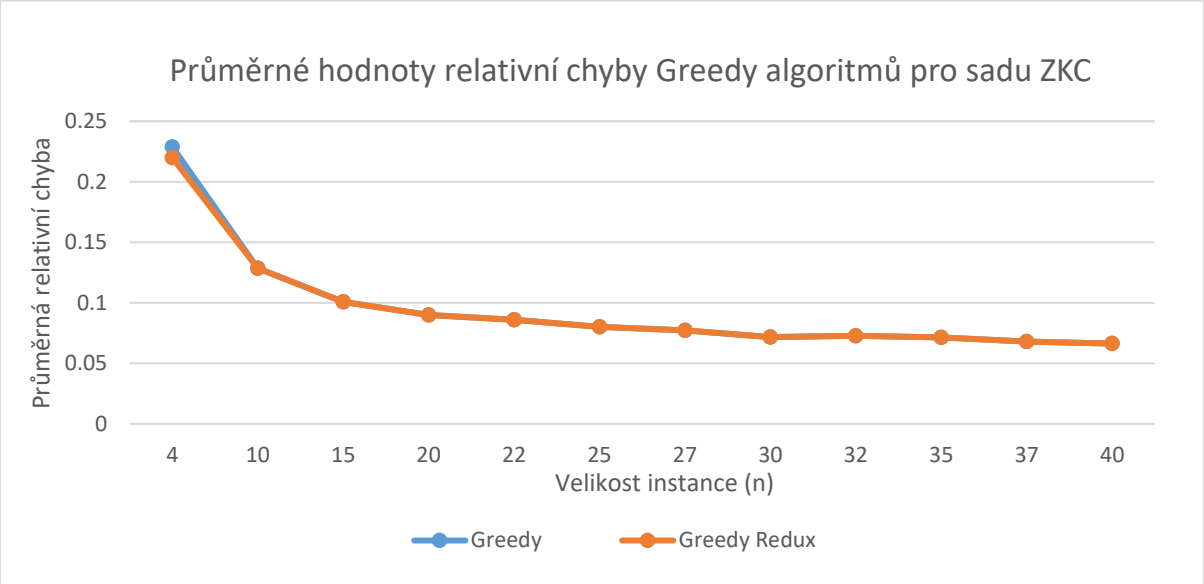
Pro některé velikosti instance byla pro algoritmus FPTAS se vstupní hodnotou $\varepsilon = 0.05$ naměřená maximální chyba 0 (řešení všech poskytnutých instancí dané velikosti bylo naprosto přesné). V grafech jsou však tyto nulové hodnoty namapovány na 1E-08, jelikož jsou tyto grafy v logaritmickém měřítku.

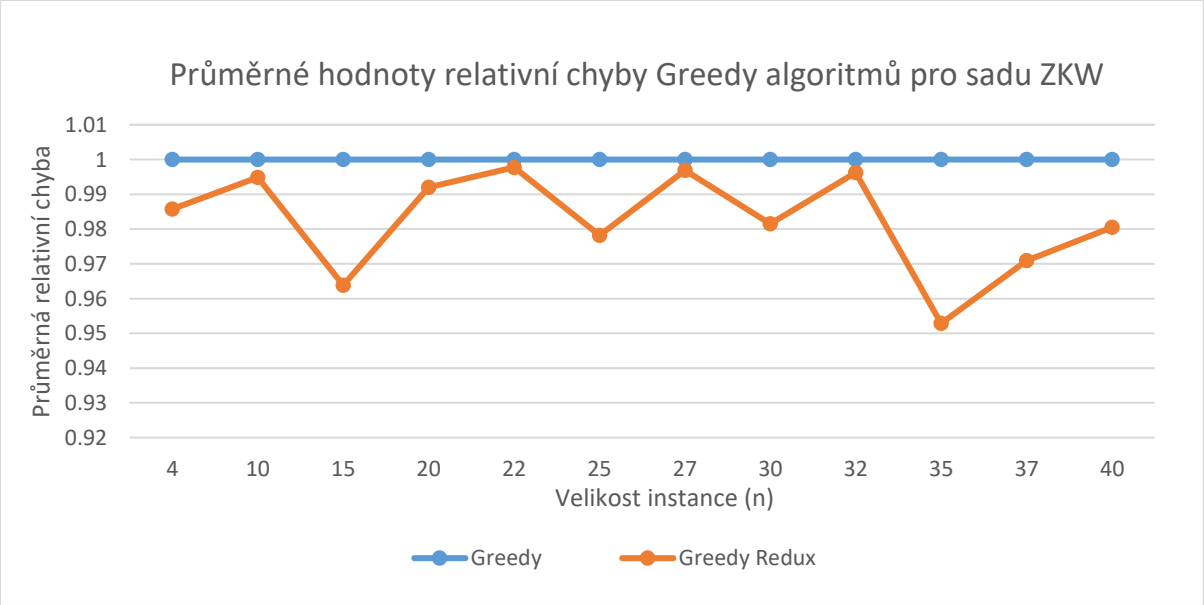


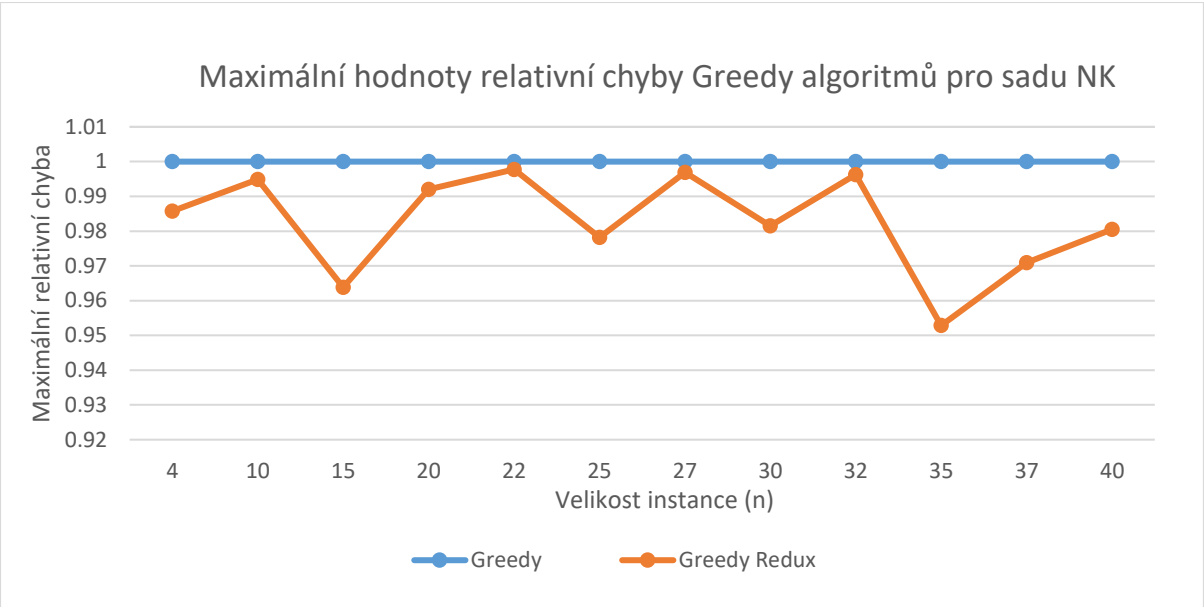

Průměrná doba běhu algoritmů pro sadu ZKC
Průměrná doba běhu [ms]
10000
1000
100
10
1
0.1
0.01
0.001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
Greedy
Brute force
Branch And Bound
DP (weight)
DP (price)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)

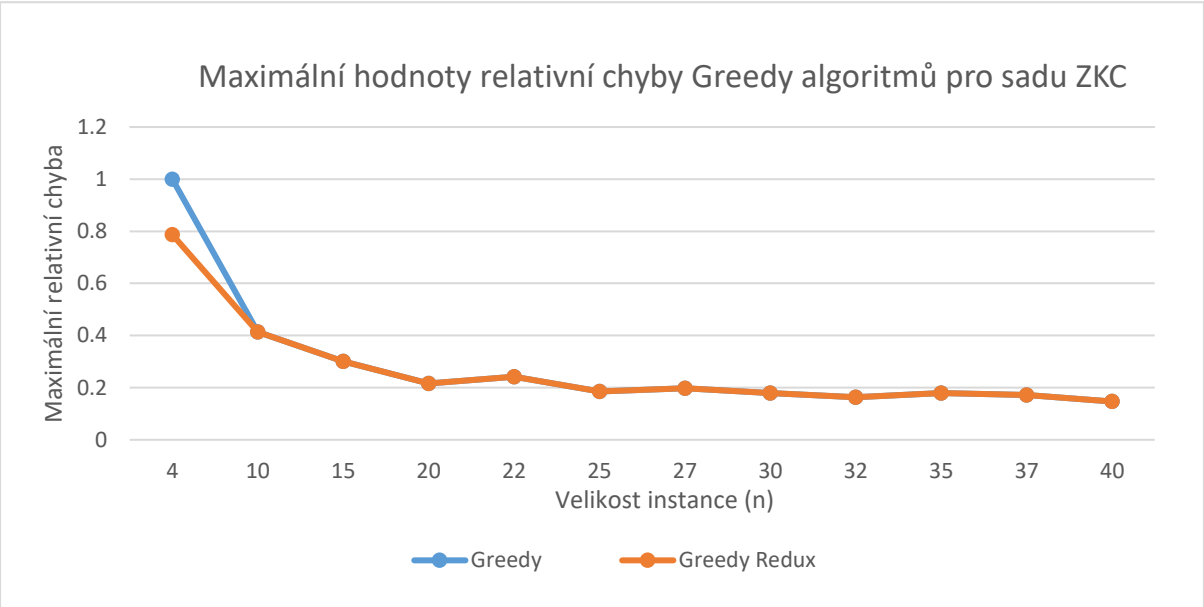

Průměrná doba běhu algoritmů pro sadu ZKW
Průměrná doba běhu [ms]
1000
100
10
1
0.1
0.01
0.001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
Greedy
Brute force
Branch And Bound
DP (weight)
DP (price)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)

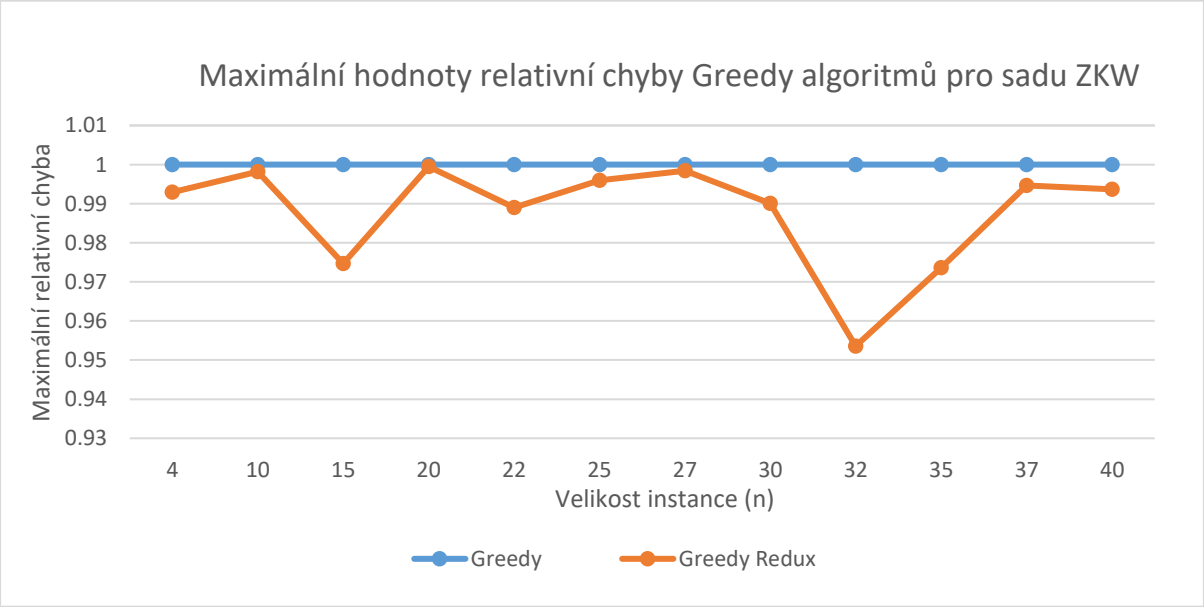

Průměrné hodnoty relativní chyby Greedy algoritmů pro sadu NK
Průměrná relativní chyba
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

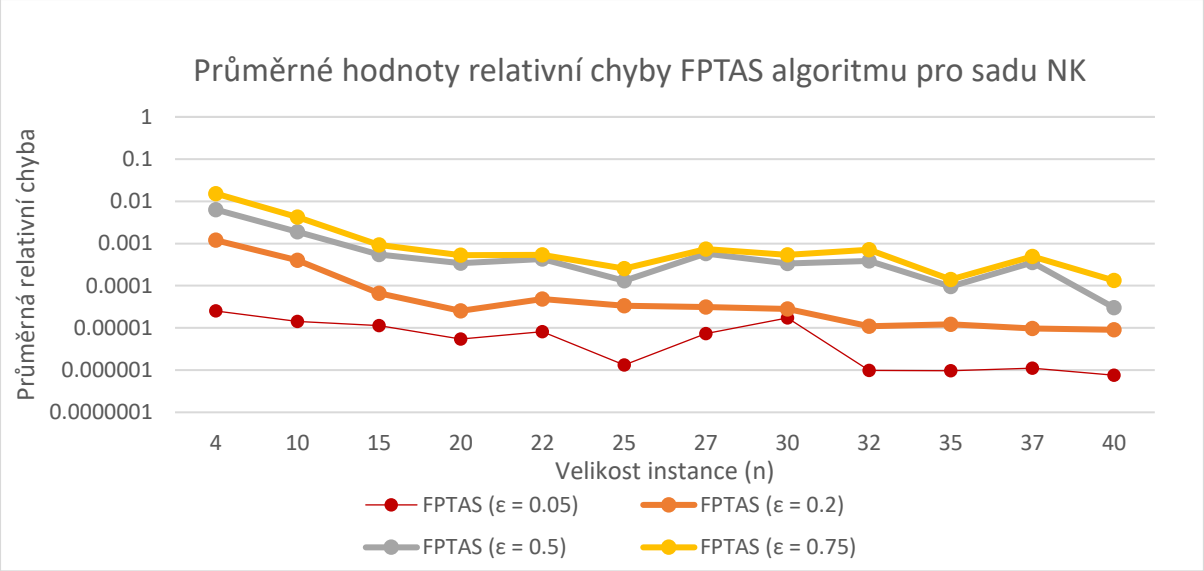

Průměrné hodnoty relativní chyby Greedy algoritmů pro sadu ZKC
Průměrná relativní chyba
0.25
0.2
0.15
0.1
0.05
0
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

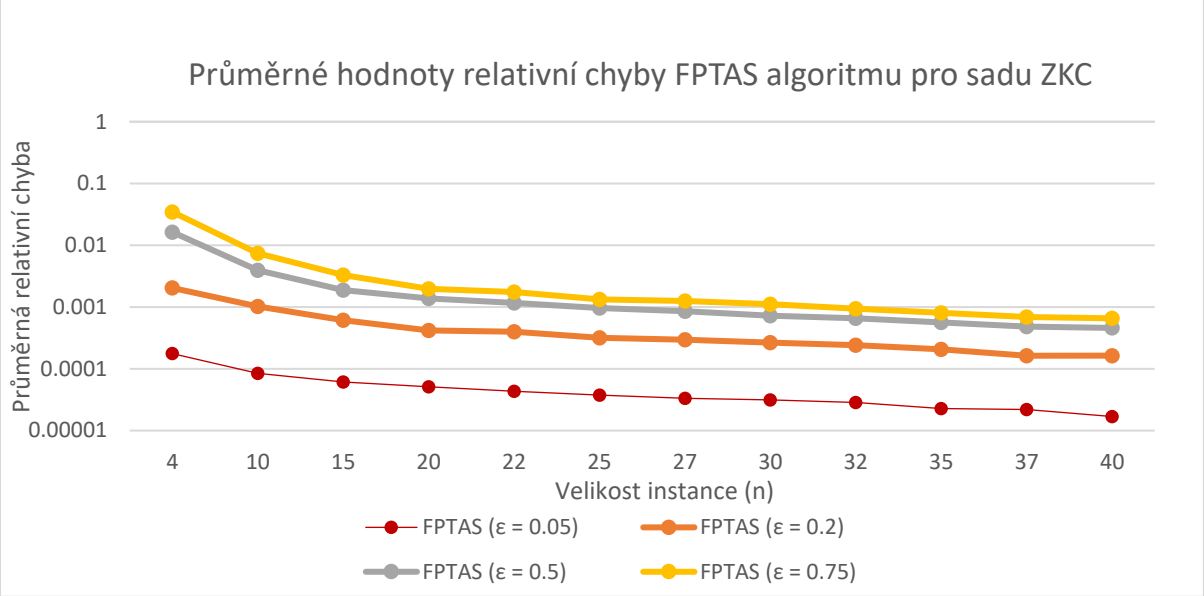

Průměrné hodnoty relativní chyby Greedy algoritmů pro sadu ZKW
Průměrná relativní chyba
1.01
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

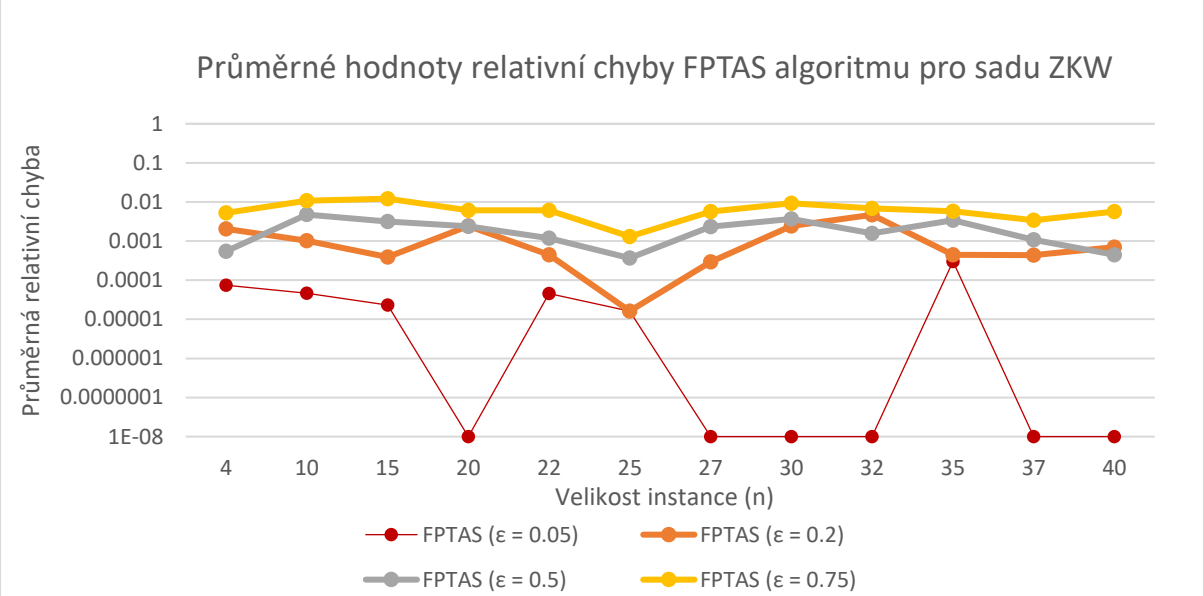

Maximální hodnoty relativní chyby Greedy algoritmů pro sadu NK
Maximální relativní chyba
1.01
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

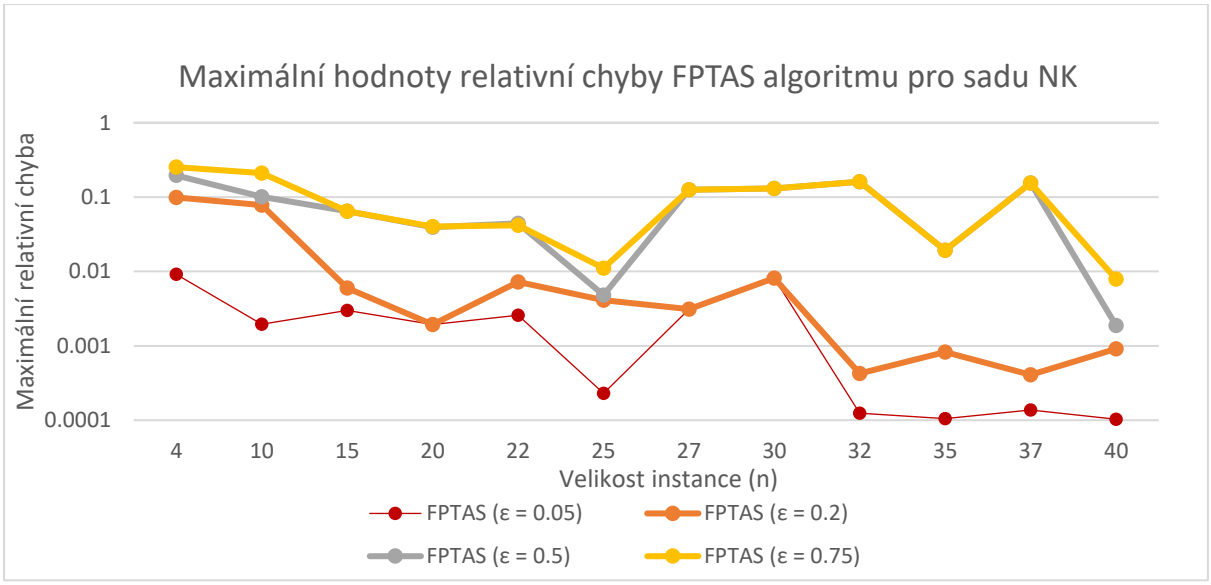

Maximální hodnoty relativní chyby Greedy algoritmů pro sadu ZKC
Maximální relativní chyba
1.2
1
0.8
0.6
0.4
0.2
0
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

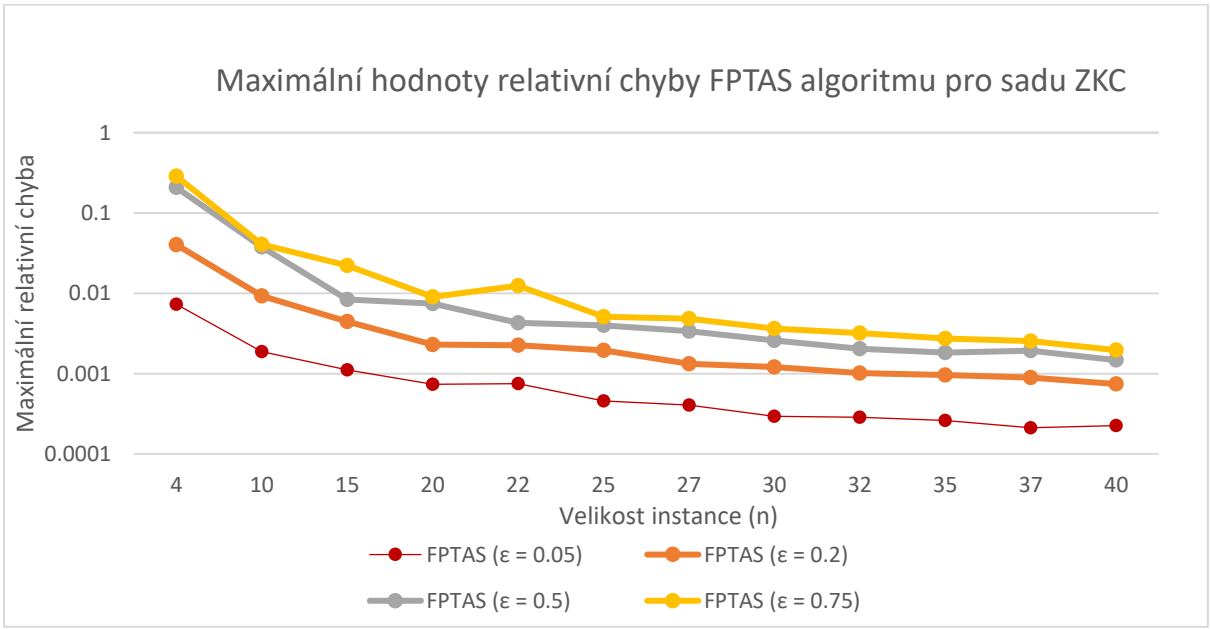

Maximální hodnoty relativní chyby Greedy algoritmů pro sadu ZKW
Maximální relativní chyba
1.01
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
4 10 15 20 22 25 27 30 32 35 37 40
Velikost instance (n)
Greedy
Greedy Redux

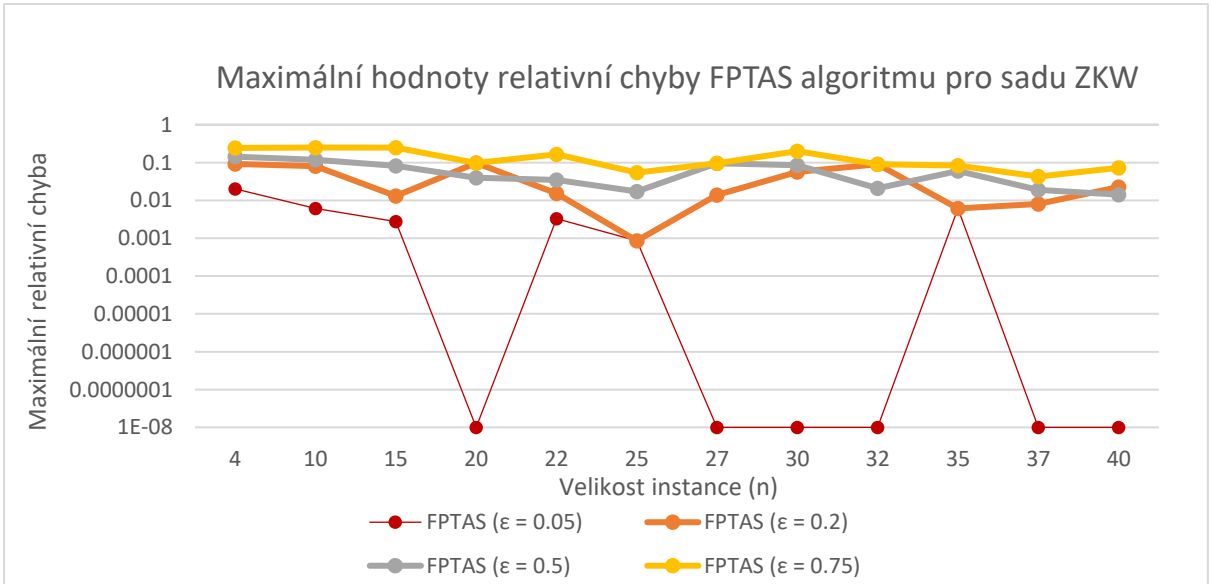

Průměrné hodnoty relativní chyby FPTAS algoritmu pro sadu NK
Průměrná relativní chyba
1
0.1
0.01
0.001
0.0001
0.00001
0.000001
0.0000001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)


Průměrné hodnoty relativní chyby FPTAS algoritmu pro sadu ZKC
Průměrná relativní chyba
1
0.1
0.01
0.001
0.0001
0.00001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)


Průměrné hodnoty relativní chyby FPTAS algoritmu pro sadu ZKW
Průměrná relativní chyba
1
0.1
0.01
0.001
0.0001
0.00001
0.000001
0.0000001
1E-08
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)


Maximální hodnoty relativní chyby FPTAS algoritmu pro sadu NK
Maximální relativní chyba
1
0.1
0.01
0.001
0.0001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)


Maximální hodnoty relativní chyby FPTAS algoritmu pro sadu ZKC
Maximální relativní chyba
1
0.1
0.01
0.001
0.0001
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)


Maximální hodnoty relativní chyby FPTAS algoritmu pro sadu ZKW
Maximální relativní chyba
1
0.1
0.01
0.001
0.0001
0.00001
0.000001
0.0000001
1E-08
4
10
15
20
22
25
27
30
32
35
37
40
Velikost instance (n)
FPTAS (ε = 0.05)
FPTAS (ε = 0.2)
FPTAS (ε = 0.5)
FPTAS (ε = 0.75)

4. Závěr

Stejně jako u předchozího úkolu se projevila závislost podoby vstupních instancí na výpočetní době (a relativní chybovosti u aprox. algoritmů). Nově implementované algoritmy jsou tedy užitečné v závislosti na povaze vstupních dat a nárokům na přesnost výsledků.

Zdaleka nejrychlejším algoritmem je greedy heuristika. Velkou nevýhodou je, že neposkytuje vůbec žádnou záruku o přesnosti výsledků. Pro vstupní sadu ZKC dosáhnul poměrně slušné průměrné i maximální relativní chyby, pro ostatní sady byl však velice nepřesný. Greedy redux chybovost mírně snížil, záruku však žádnou nepřidal.

Zajímavých výsledků dosáhly algoritmy DP. Na naměřených výsledcích se projevilo, že dekompozice dle váhy byla pro všechny poskytnuté vstupní sady efektivnější (nejmenší rozdíl mezi dekompozicemi byl pro sadu ZKW). Také se ukázalo, že pro sadu NR je dekompozice podle ceny efektivnější oproti Branch and Bound až pro instance větší, než 30. Pro sadu ZKW byla doba běhu těchto dvou algoritmů dokonce velmi podobná pro všechny velikosti instancí.

Implementace FPTAS algoritmu se ukázala být relativně přijatelným kompromisem mezi výpočetní dobou a přesností. Navíc se ukázalo, že reálná průměrná i maximální relativní chyba byla pro poskytnuté sady téměř vždy řádově nižší, než teoreticky předpokládaná přesnost. Ze zvolených přesností se ukázala být „nejvhodnější“ přesnost $\varepsilon = 0.5$ – pro nižší ε byla doba běhu velmi podobná dekompozici podle váhy, pro vyšší ε již výpočetní doba příliš neklesala.