Exaktní a heuristická řešení konstruktivního problému batohu

NI-KOP DÚ 2

Tomáš Kalabis

Konstruktivní problém batohu

Je dáno

- celé číslo *n* (počet věcí)
- celé číslo *M* (kapacita batohu)
- celé kladné číslo *B*
- konečná množina $V = \{v_1, v_2, \dots, v_n\}$ (hmotnosti věcí)
- konečná množina $C = \{c_1, c_2, \dots, c_n\}$ (ceny věcí)

Je možné zkonstruovat množinu $X=\{x_1, x_2, \dots, x_n\}$, kde každé x_i je 0 nebo 1, tak, aby platilo

$v_1x_1 + v_2x_2 + \dots + v_nx_n \leq M$ (aby batoh nebyl přetížen).

a výraz

$c_1x_1 + c_2x_2 + \dots + c_nx_n$ byl maximální

Výsledkem tak je množina vybraných věcí v batohu, které jsou lehčí než kapacita batohu (číslo *M*) a hodnota všech věcí je nejcennější.

Technologie a Program

Jako hlavní technologie byl vybrán jazyk *Python* verze 3.9.5 pro svou uživatelskou přívětivost, jednoduchou práci s moduly a jejich bohatou podporu jako je například *matplotlib* (modul pro vykreslování grafů) nebo *numpy* (modul pro matematické výpočty). Pro programování bylo použito prostředí *Jupyter Notebook*. Nevýhodou využití jazyka *Python* je jeho nižší rychlost oproti jazykům jako *C++*.

Program se spouští vyhodnocením všech buněk v Jupyter Notebooku. Aby program správně fungoval, musí být vytvořená složky `results/steps` , `results/time a results/fult` v adresáři s programem a složky `NK/`, `ZKC/` a `ZKW/` by měly být také ve stejné složce jako program (případně upravit volání funkce).

Řešení a metody

Konstruktivní Problém batohu byl řešen několika metodami, které byly následně experimentálně testovány. Tato kapitola popisuje využité metody.

Metoda Větví a hranic (Branch and Bound)

Metoda je implementovaná jako stromová rekurze v jehož každém uzlu se kontroluje, zda podstrom tohoto uzlu je schopen naplnit batoh tak, aby dosavadní maximální cena plnění batohu byla překonána. V negativním případě se podstrom tohoto uzlu vůbec nezkouší, jelikož víme, že nová maximální cena batohu v tomto podstromě nebude nalezena. Časová náročnost této metody je $O(2^n)$.

Greedy Metoda

Tato metoda nejprve před samotným výpočtem seřadí předměty v batohu podle poměru cena/hmotnost v klesajícím pořadí. Nejvýhodnější předměty jsou tedy jako první. Následně se provede výpočet, kde algoritmus vkládá předměty do batohu v tomto pořadí a pokud se do batohu nevejde, předmět se vynechá. Metoda má značnou výhodu v její rychlosti, ale je velmi náchylná na chyby a výsledek tak ve většině případů není optimální, avšak únosný. Časová náročnost výpočtu je $O(n)$, ale náročnost metody jako celek je $O(n \cdot \log n)$ kvůli řazení předmětů na začátku.

Greedy Redux Metoda

Tato metoda opravuje jeden nedostatek Greedy metody. Při jejím použití může být řešení menší než jeden nejhodnotnější předmět, který se do batohu ještě vejde. Implementace této metody je tedy maximum z nejhodnotnějšího předmětu, který se do batohu vejde a řešení Greedy metodou. Časová náročnost je stejná jako u Greedy metody.

Dynamická heuristika podle ceny (DHBP)

Tato metoda využívá dynamického programování, konkrétně ukládání mezivýsledků za účelem zamezení opakování stejného podproblému. Aby však šlo využít dynamické programování podle ceny, je třeba přeformulovat problém.

Označme E(i, c) instanci 0/1 inverzního problému batohu se zadanou cenou c, která vznikne z řešené instance omezením na prvých i věcí. Označme dále W(i, c) sumární hmotnost věcí řešené instance. Pak platí

- $W(0,0) = 0$
- $W(0,c) = \infty$ pro všechna $c > 0$
- $W(i+1, c) = \min(W(i, c), W(i, c-ci+1)+wi+1)$ pro všechna $i > 0$.

Z výsledných řešení W(n, c) vybereme řešení, pro které je W(n, c) < M pro největší c. Je zřejmé, že řešení můžeme omezit na c menší nebo rovné součtu cen všech věcí.

Algoritmus zkouší jednotlivé možné *c* od maximální možné hodnoty (součet cen předmětů v batohu) po nižší hodnoty *c,* a jakmile narazí na řešení které je menší než *M*, je toto řešení zároveň maximální a tedy je řešením konstruktivního problému dané instance batohu.

Metoda FPTAS

Metoda FPTAS nejdříve poměrově zmenší všechny ceny předmětů v batohu podle argumentu $0 < \varepsilon < 1$. Následně se využije metoda dynamické heuristiky podle ceny. Tato metoda je urychlením, avšak řešení ztrácí svou přesnost, tu je však možné korigovat parametrem ε. Zároveň je třeba ošetřit, zda se ceny naopak nezvětší, což je možné v určitých instancích.

Experimenty

Experimenty probíhaly na sadách *NK, ZKC* a *ZKW*, které byly dodány se zadáním. Zkoumali se počty navštívených uzlů při jednotlivých instancích, jednotlivých hodnotách *n* a jednotlivých metodách pospaných výše.

Testování probíhalo na notebooku s následujícími parametry:

- Operační systém: Ubuntu 21
- RAM: 8GB DDR4
- Procesor: i5-6300HQ 2.3GHz

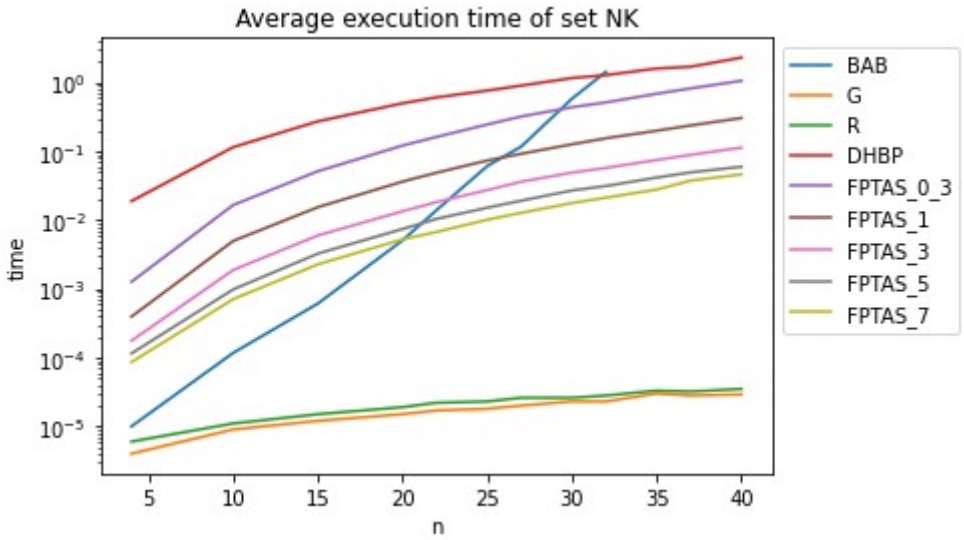
Výsledky

V následující sekci jsou popsané výsledky měření hodnot během experimentování a diskuse k nim.

Poznámka: Všechny grafy jsou vykreslené s logaritmickým měřítkem což je nutno brát v potaz při čtení grafu. Zároveň algoritmus FPTAS s volbou ε rovnu *0,5* je znázorněno v grafech jako *‚FPTAS_5‘, FPTAS s volbou* ε rovnu *0,03* je znázorněno v grafech jako *‚FPTAS_0_3‘* a analogicky s ostatníma.

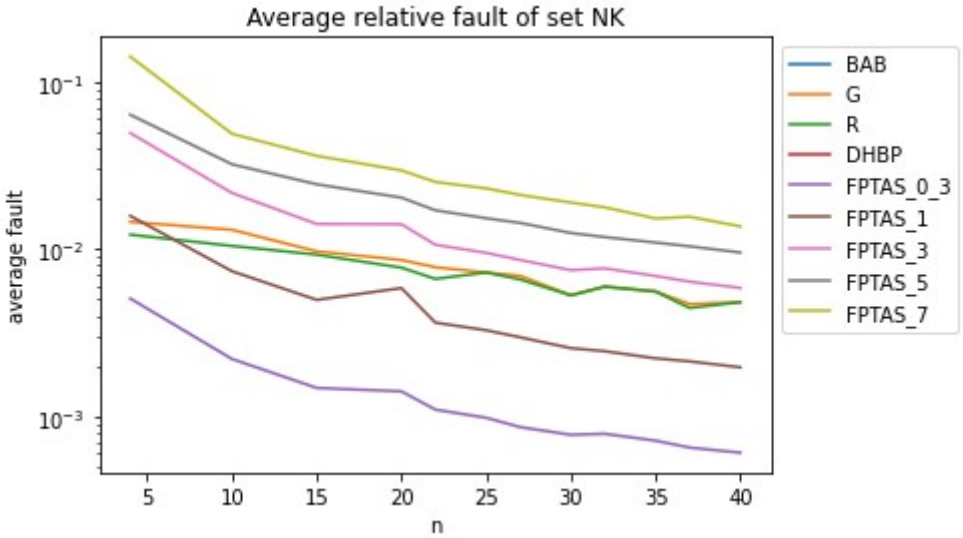
Sada NK

Průměrný čas vykonávání



Z grafu lze vyčíst, že průměrná rychlost je nejnižší u algoritmů *Greedy* a *Redux* a prakticky v rychlosti výpočtu dominují dle očekávání. Algoritmus FPTAS s různými volbami *ε* také téměř kopíruje křivku algoritmu *DHBP,* což je také dle očekávání, akorát se algoritmus zrychluje při vyšší volbě *ε.* U *DHBP* algoritmu si lze všimnout, že u nízkých je nejpomalejší, u vyšších však díky memoizaci je nárůst času pro vyšší *n* mírnější. V této konkrétní sadě je časově výhodnější algoritmus *DHBP* než *B&B* až od *n* rovno 32*.* Algoritmus *Branch and Bound* se jediný vymyká svou téměř exponenciální křivkou (dle očekávané složitosti).

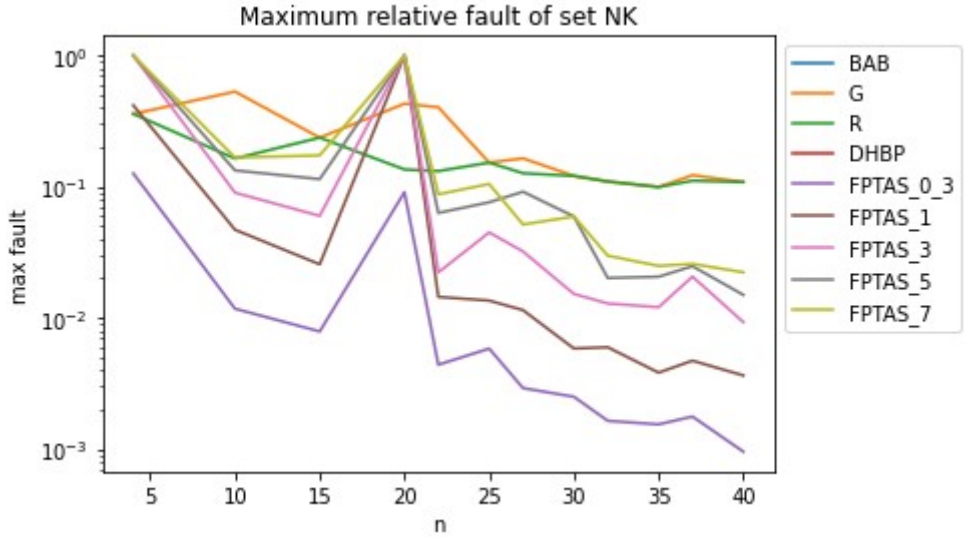
Průměrná relativní chyba



Na začátku je třeba zmínit, že algoritmy *BAB* a *DHBP* nejsou vidět na grafu výše, jelikož jejich průměrná relativní chyba byla pro každé *n* rovno nule. Jsou to tedy výsledkově bezchybné algoritmy a oproti ostatním dominují. Vezmeme li v potaz pouze metody, které

připouští chybu, je patrné, že metoda *FPTAS* s $\varepsilon < \sim 2{,}5$ je v průměrné relativní chybě lepší (u ε=0,3 je to řádově) než *Greedy* metody. Metody *FPTAS* s vyšším *ε* mají poté (nej)vyšší průměrnou relativní chybu.

Maximální relativní chyba

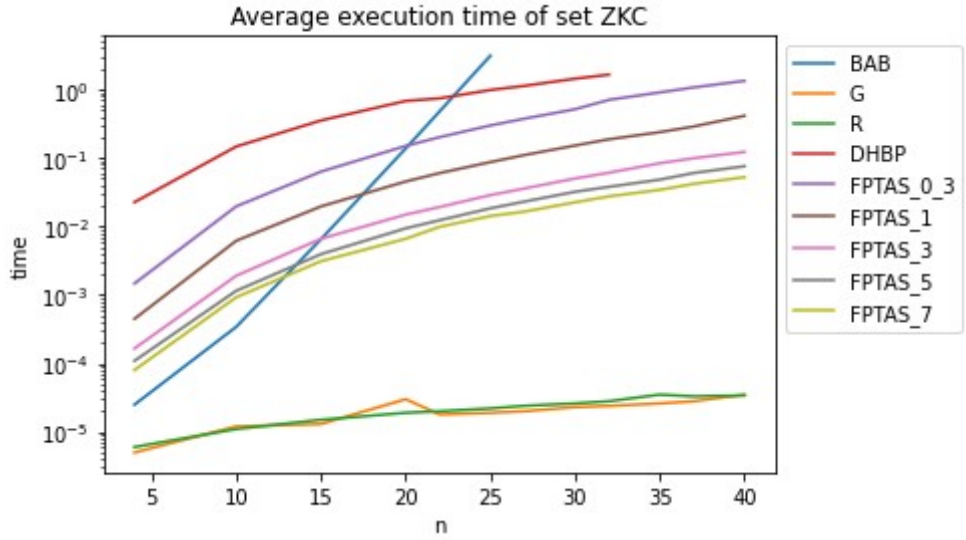


Z grafu je na první pohled patrné, že u metody *FPTAS* je u $n = 20$ vysoká odchylka. Dále lze vyčíst, že maximální relativní chyba je nejvyšší u *Greedy* a *Redux* metod a metoda *FPTAS* má (až dvou řádově – záleží na volbě *ε*) menší maximální relativní chybu. Stojí za zmínku, že ačkoli FPTAS s $\varepsilon > \sim 3$ má vyšší průměrnou relativní chybu (dle předchozího grafu), má (až řádově) menší maximální relativní chybu. Je také patrné, že se zvyšujícím se *n* se zároveň snižuje maximální relativní chyba (zejména u *FPTAS*).

Všechny metody v sadě *NK* odpovídají očekávání, kromě „zubu“ v měření maximální relativní chyby při *n*=20.

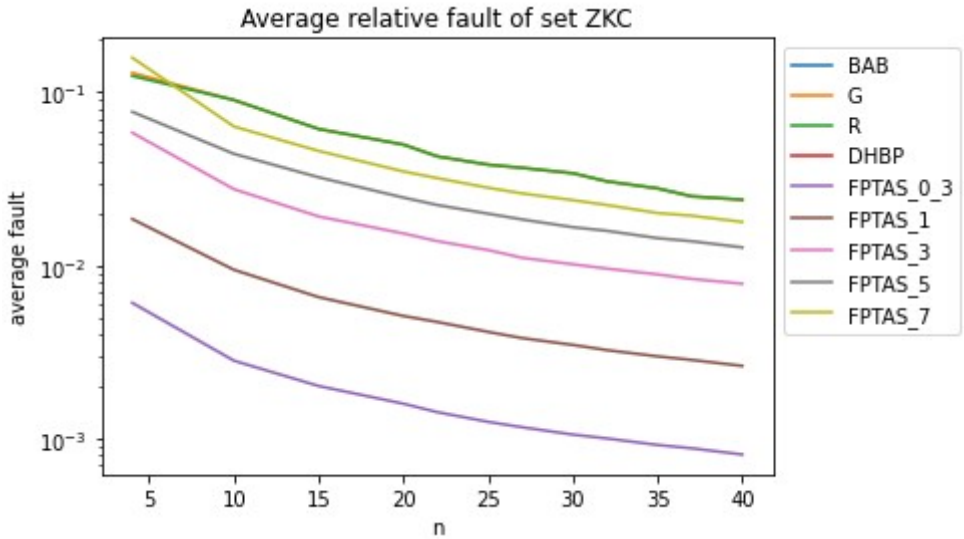
Sada ZKC

Průměrný čas vykonávání



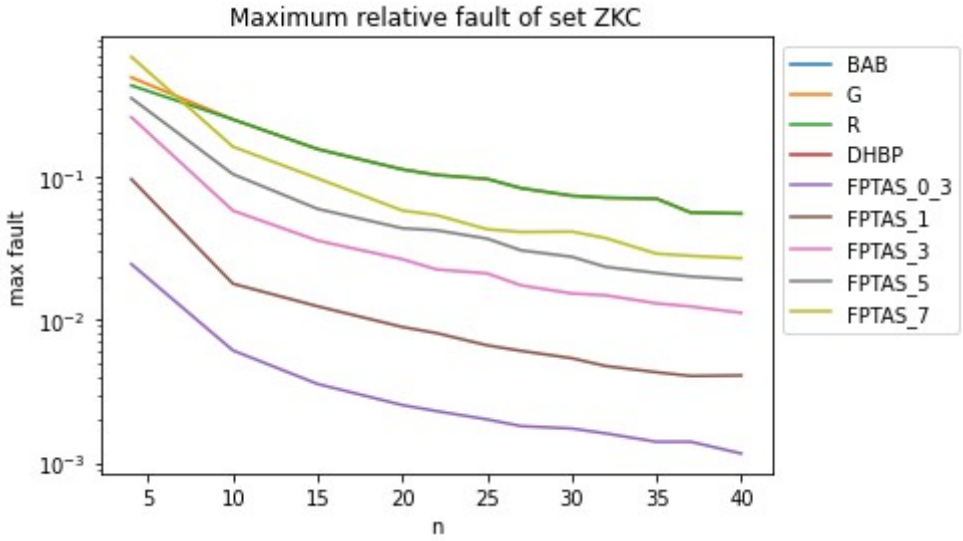
Sada se velmi podobá sadě *NK* ve většině metodách kromě křivky popisující metodu *B&B*, která oproti sadě *NK* má strmější růst. Algoritmus DHBP potažmo FPTAS je také mírně zpomalený (DHBP v NK při n = 20 – 0,50149; DHBP v ZKC při n = 20 – 0,676812).

Průměrná relativní chyba



Opět můžeme o sadě *ZKC* říci, že průměrná relativní chyba metod je podobná sadě *NK*. Nejvýraznější změna oproti sadě *NK* je téměř řádové zvětšení průměrné relativní chyby metod *Greedy* a *Redux*. Z toho lze usuzovat, že tyto metody jsou náchylné dělat větší chyby ve výpočtu při určitých vstupních datech, kdežto *FPTAS* náchylný není. Také křivky popisující hodnoty jsou „hladší“.

Maximální relativní chyba

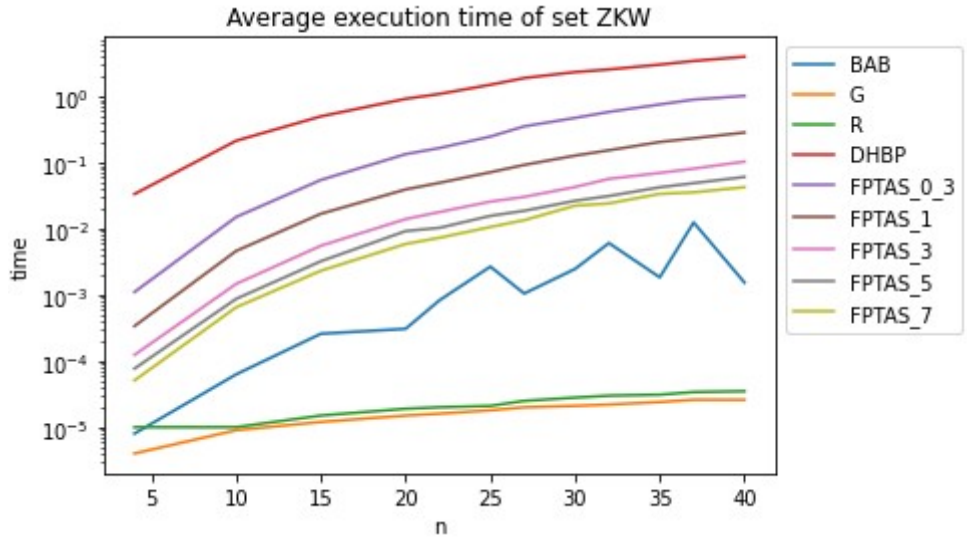


O maximální relativní chybě v sadě *ZKC* lze také říct, že se velmi podobá sadě *NK*. Neobsahuje však odchylku a opět se jeví „hladší“.

Většina metoda naplňuje očekávání, a naopak metody *Branch and Bound* se liší oproti sadě *NK* větší strmostí růstu. Další významná změna je řádové zhoršení chyby (průměrná i maximální) metody *Greedy*/*Redux*.

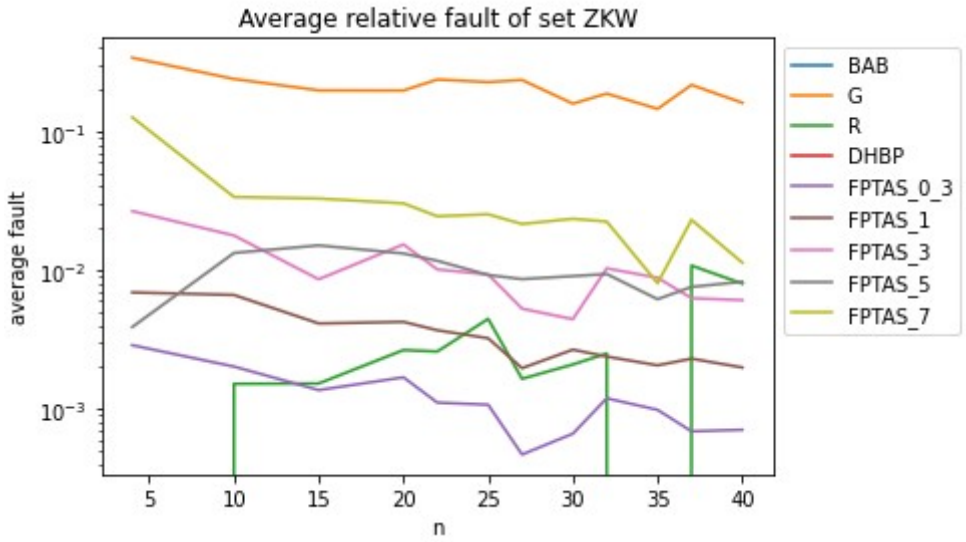
Sada ZKW

Průměrný čas vykonávání



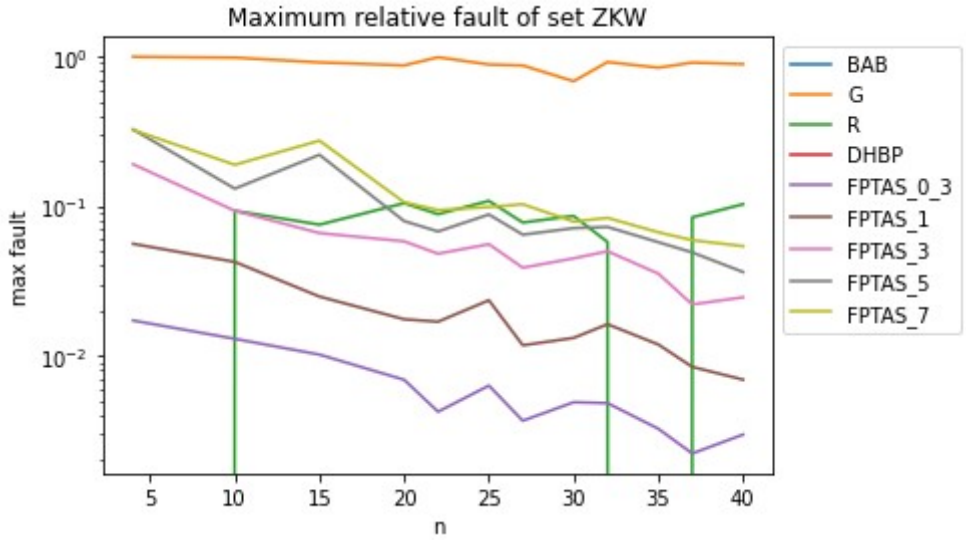
Sada *ZKW* se při měření průměrného času vykonávání liší na první pohled metodou *B&B* a jejím řádovým zlepšením měřených hodnot oproti ostatním sadám, což není očekávané. To může být zapříčiněno speciálními vstupními daty.

Průměrná relativní chyba



Na sadě *ZKW* lze pozorovat výhodu metody *Redux* oproti *Greedy* metodě. Proti očekáváním má při *n=4* a *n=35* metoda *Redux* nulovou chybu. Vstupní data jsou tedy nejspíš příhodná pro tuto situaci. Zároveň lze z grafu vyčíst, že *FPTAS* s $\varepsilon = 0{,}3$ má při některých *n* větší průměrnou relativní chybu než *FPTAS* s $\varepsilon = 0{,}5$, což je proti očekáváním.

Maximální Chyba



Tento graf opět potvrzuje speciální vstupní data pro algoritmy *Greedy* a *Redux* (viz dříve) a opět navazuje svou podobností na sadu *ZKC*.

Sada *ZKW* se vymyká rozdíly v maximální a průměrné relativní chybě mezi *Greedy* a *Redux* heuristikou. Také je metoda *Branch and Bound* řádově rychlejší než sady předešlé a je celkově křivka popisující průměrný čas vykonávání neočekávaná.

Tabulky

Následující tabulky popisují měřené hodnoty ze sady *NK*.

Průměrný čas vykonávání

n	B&B	DHBP	Greedy	Redux	FPTAS_03	FPTAS_1	FPTAS_3	FPTAS_5	FPTAS_7
4	1e-05	0.018941	4e-06	6e-06	0.001269	0.000395	0.000176	0.000115	8.6e-05
10	0.000117	0.115187	9e-06	1.1e-05	0.016589	0.004979	0.001883	0.000982	0.000708
15	0.000612	0.273793	1.2e-05	1.5e-05	0.051522	0.015442	0.005984	0.003259	0.002275
20	0.005089	0.50149	1.5e-05	1.9e-05	0.121717	0.036391	0.013462	0.007434	0.005251
22	0.013949	0.611087	1.7e-05	2.2e-05	0.162352	0.048967	0.018329	0.010429	0.006727
25	0.061392	0.767539	1.8e-05	2.3e-05	0.246607	0.073743	0.027371	0.015181	0.010147
27	0.117486	0.906565	2e-05	2.6e-05	0.32196	0.09233	0.03627	0.01918	0.01275

					8	3	3	8	5
30	0.586125	1.167291	2.3e-05	2.6e-05	0.439163	0.127029	0.049119	0.027054	0.017736
32	1.437378	1.287904	2.3e-05	2.8e-05	0.515822	0.155201	0.057541	0.031388	0.021204
35	--	1.607644	3e-05	3.3e-05	0.690676	0.200226	0.075038	0.041635	0.02787
37	--	1.704578	2.8e-05	3.2e-05	0.825498	0.238516	0.089216	0.049561	0.037572
40	--	2.314415	2.9e-05	3.5e-05	1.064828	0.305981	0.113403	0.059801	0.046247

Průměrná relativní chyba

n	**B&B**	**DHBP**	**Greedy**	**Redux**	**FPTAS_03**	**FPTAS_1**	**FPTAS_3**	**FPTAS_5**	**FPTAS_7**
4	0	0	0.014534	0.012194	0.005059		0.049361		0.141282
10	0	0	0.013041	0.01044	0.002203		0.021635		0.048848
15	0	0	0.009684	0.009263	0.001479		0.014119		0.036053
20	0	0	0.008604	0.007744	0.001411		0.014076		0.029526
22	0	0	0.007785	0.006634	0.001098		0.010609		0.025178
25	0	0	0.007245	0.007245	0.000981		0.00949		0.023061
27	0	0	0.006918	0.006588	0.000863		0.008613		0.021066
30	0	0	0.00529	0.00529	0.000776		0.007476		0.018967
32	0	0	0.005963	0.005963	0.000787		0.007664		0.017761
35	0	0	0.005577	0.005577	0.000716		0.006906		0.015234
37	0	0	0.004691	0.004446	0.000651		0.006386		0.015601
40	0	0	0.004816	0.004816	0.000607		0.005858		0.013666

Maximální relativní chyba

n	**B&B**	**DHBP**	**Greedy**	**Redux**	**FPTAS_03**	**FPTAS_1**	**FPTAS_3**	**FPTAS_5**	**FPTAS_7**

4	0	0	0.129397	0.124305	0.006092		0.058454		0.158338
10	0	0	0.09023	0.09023	0.002825		0.027579		0.063449
15	0	0	0.061398	0.061398	0.002018		0.019203		0.045854
20	0	0	0.049914	0.049914	0.001595		0.015321		0.034944
22	0	0	0.042666	0.042666	0.001426		0.013835		0.031946
25	0	0	0.038217	0.038217	0.001253		0.012289		0.028121
27	--	0	0.036647	0.036647	0.001167		0.011089		0.026085
30	--	0	0.034098	0.034098	0.001058		0.01017		0.023845
32	--	0	0.030635	0.030635	0.001005		0.009606		0.022366
35	--	--	0.027882	0.027882	0.000922		0.008893		0.02009
37	--	--	0.025104	0.025104	0.000882		0.008381		0.0194
40	--	--	0.023983	0.023983	0.000812		0.007853		0.017865

Shrnutí experimentů

Při porovnání jednotlivých sad zjistíme, že sada Z*KW* má nejspíše speciální vstupní data nastavená tak, aby se co nejvíce projevil rozdíl mezi metodami *Greedy* a *Redux* a zároveň taková data, že metoda *Branch and Bound* je výrazně rychlejší (je téměř řádově rychlejší než *FPTAS* s *ε=0,7*). Sada *ZKC* má pak nejspíše méně přívětivá vstupní data, jelikož všechny metody mají časové zhoršení oproti sadě *NK*.

Závěr

Z experimentů lze zesumarizovat, že algoritmy *B&B* a *DHBP* mají přesné řešení oproti ostatním metodám, které se dopouštějí při výpočtu chyby. Nejnižší průměrné a maximální relativní chyby se dopouští metoda *FPTAS* s nízkou volbou *ε* (např. 0,1). Nejvyšší průměrné relativní chyby se dopouští metoda *FPTAS* s vysokou volbou *ε* (např. 0,7) s náhodnou sadou dat. Avšak s již speciálně upravenou sadou dat se dopouští největší relativní chyby *Greedy*/*Redux* metody a maximální relativní chyby se dopouštějí opět metody *Greedy*/*Redux* nezávisle na vstupních datech. Co se týče rychlosti výpočtu, jsou dominantní metody *Greedy*/*Redux*. Řádově pomalejší je poté *FPTAS* u kterého platí čím nižší volba *ε*, tím pomalejší je, a to téměř nezávisle na vstupu. Nejpomalejší jsou pak metody *DHBP* a *Branch and Bound*, která je od jistého *n* (v sadě *NK* n=32) pomalejší než *DHBP* a je téměř exponenciální.