Zpráva o řešení domácího úkolu č. 2

**Exaktní a heuristická řešení konstruktivního problému batohu**

<https://moodle-vyuka.cvut.cz/mod/assign/view.php?id=153054>

0/1 problém batohu je NP-těžký optimalizační problém. Výstupem jeho **konstruktivní** verze je binární množina, jejíž prvky určují zastoupení jednotlivých věcí ve výsledném řešení (na rozdíl od **rozhodovací** verze, kde je výstupem *True/False*). Pro **exaktní** řešení tohoto problému existují algoritmy, jako např. *Branch & Bound* nebo *dynamické programování* (dekompozice dle ceny/váhy), které vrací přesný a optimální výsledek. Lze však použít i **heuristické** algoritmy, jako např. *Greedy* nebo přesnější *Redux*, které naleznou řešení mnohem rychleji, avšak na úkor větší chybovosti. V algoritmu *FPTAS*, který využívá výhod dynamického programování, lze navíc chybovost, a tedy i rychlost, regulovat dle potřeby.

Tato zpráva informuje o výsledcích experimentů, jejichž účelem bylo vyhodnotit kvalitu a rychlost zmíněných algoritmů (jmenovitě tedy *Branch & Bound*, *dynamické programování* dle váhy, *Greedy*, *Redux* a *FPTAS*). Pro experiment byly použity tři sady testovacích dat – NK, ZKC, ZKW – s různými velikostmi a počty instancí. Pro všechny testovací sady (a jejich dílčí sady instancí dané velikosti) byly zaznamenávány doby běhu (tj. průměrný čas CPU) a chyby (tj. maximální a průměrná chyba) jednotlivých algoritmů. Tyto údaje jsou zobrazeny v doprovodných grafech a tabulkách dále v tomto dokumentu.

Na rozdíl od předchozí úlohy byla nově použita knihovna *timeit*, která výrazně vylepšila měření času v jazyce Python. Tato zpráva tak poskytuje **přesnější** časové výsledky. Dále, generování přiložených grafů a tabulek bylo plně zautomatizováno a vylepšeno pomocí knihoven *pandas* a *matplotlib*.

**ZVOLENÉ ALGORITMY**

**1**

Označení v experimentech: **B&B**

Popis: Algoritmus prochází všechny konfigurace až do samotných listů rekurze a ukládá nejlepší nalezené řešení. Po cestě kontroluje podmínky na max. váhu a min. cenu. Pokud dosavadní sestava věcí překročí nosnost batohu, algoritmus pokračuje jinou větví rekurze. Obdobně pokud ceny následujících věcí (společně s již dříve přidanými) nepřekročí cenu nejlepšího řešení. Algoritmus tak vždy vrací sestavu věcí s nejlepší cenou.

**2**

Označení v experimentech: **DYNAMIC**

Popis: Výhoda dynamického programování spočívá v ukládání mezivýsledků, které jsou nadále používány pro následující výpočty. Algoritmus takové výsledky (v našem případě) ukládá do 2D pole. V závislosti na tom, kterou dekompozici používáme (dle váhy nebo ceny), do tohoto pole ukládáme průběžné ceny, příp. váhy pro různé sestavy věcí. Algoritmus pak tyto vypočtené ceny/váhy používá při rekurzivním prohledávání, čímž dosahuje zrychlení výpočtu. Pro experimenty uvedené dále v tomto dokumentu byla použita dekompozice **dle váhy**.

**3**

Označení v experimentech: **GREEDY**

Popis: Algoritmus nejprve heuristicky seřadí všechny věci podle klesajícího poměru cena/váha. Z této posloupnosti jsou následně vybírány jednotlivé věci, dokud je batoh unese. Takový algoritmus však nepracuje přesně, vzato s vyšší rychlostí. Tato zpráva dokumentuje experimenty, které předchozí tvrzení později potvrdí.

**4**

Označení v experimentech: **REDUX**

Popis: Jedná se o vylepšení algoritmu *Greedy*. Řešení je navíc porovnáno s nejcennější (a zároveň únosnou) věcí. Hodnotnější z nich je nakonec výsledkem finálním.

**5**

Označení v experimentech: **FPTAS**

Popis: Algoritmus je postavený na základech dynamického programování s dekompozicí dle ceny. Před samotným výpočtem jsou však ceny upraveny a sníženy (2D pole mezivýsledků je tak menší). Tím je dosaženo snížení paměťové a časové náročnosti, která je však kompenzována větší chybovostí. Vstupem algoritmu je mimo jiné také parametr přesnosti, díky kterému lze tuto chybovost regulovat.

**PLATFORMA**

Hardware: Notebook Lenovo, Intel Core i5-8250U 1.6 GHz, 4 GB RAM

Operační systém: Windows 10 Home, 64bit

Programovací jazyk: Python

**PROVEDENÉ EXPERIMENTY**

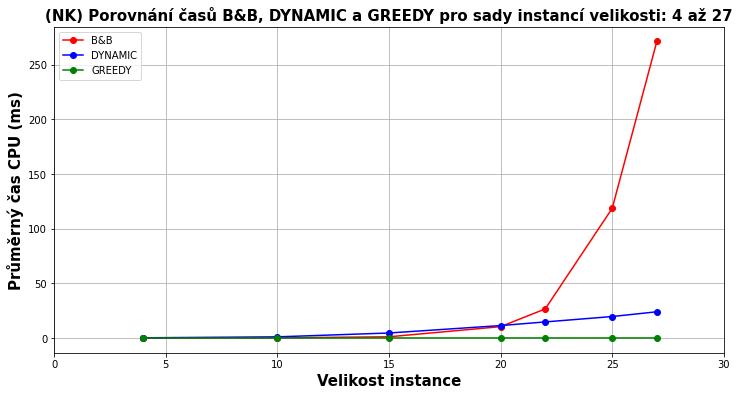
Celkem bylo provedeno 12 experimentů rozdělených do tří bloků podle testovacích sad NK, ZKC a ZKW. Jejich značení je následující: 1A–D, 2A–D, 3A–D

**1A**

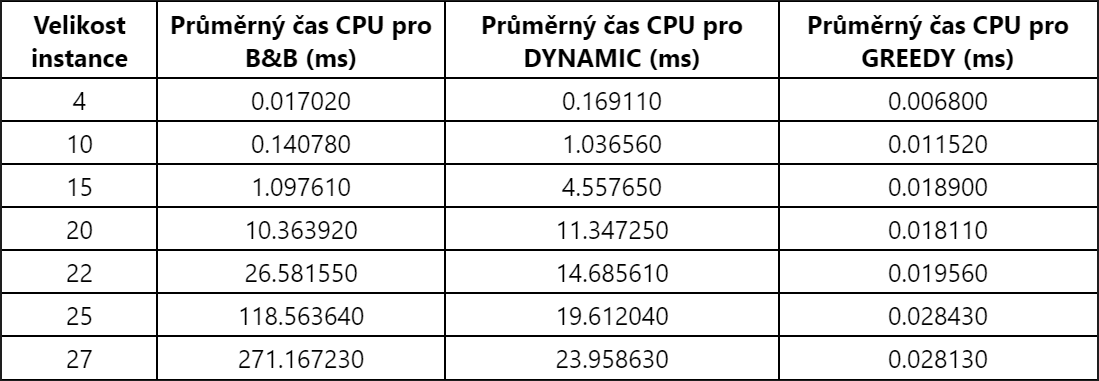
Algoritmy: B&B, DYNAMIC, GREEDY

Testovací sada: NK

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27



Data:



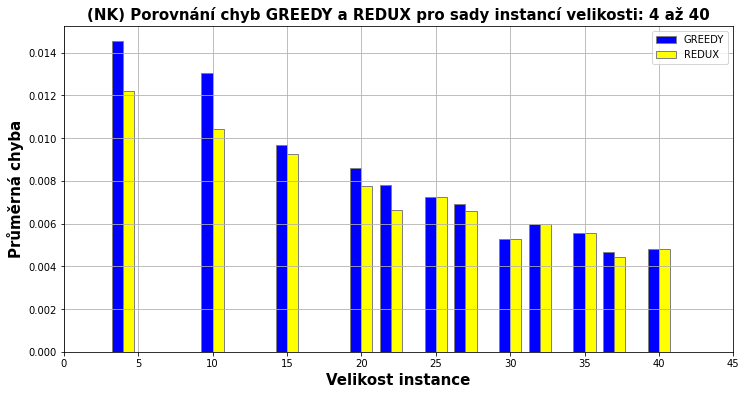
Komentář autora experimentu: V důsledku strmého stoupání křivky B&B nebyly měřeny větší instance, jelikož by postupně způsobily splynutí křivek DYNAMIC a GREEDY. Jednotlivé křivky odpovídají předpokladům ohledně rychlosti zkoumaných algoritmů. Stoupání heuristického GREEDY je díky své rychlosti ve výše uvedeném grafu téměř nepostřehnutelné. Zajímavostí je, že pro menší velikosti instance je rychlost dynamického řešení dokonce horší než B&B.

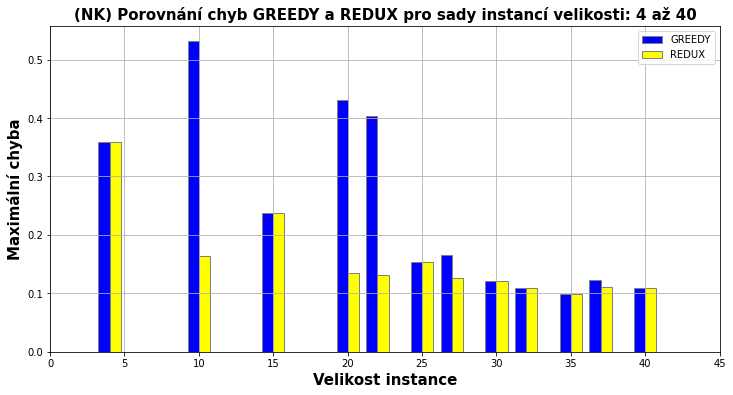
**1B**

Algoritmy: GREEDY, REDUX

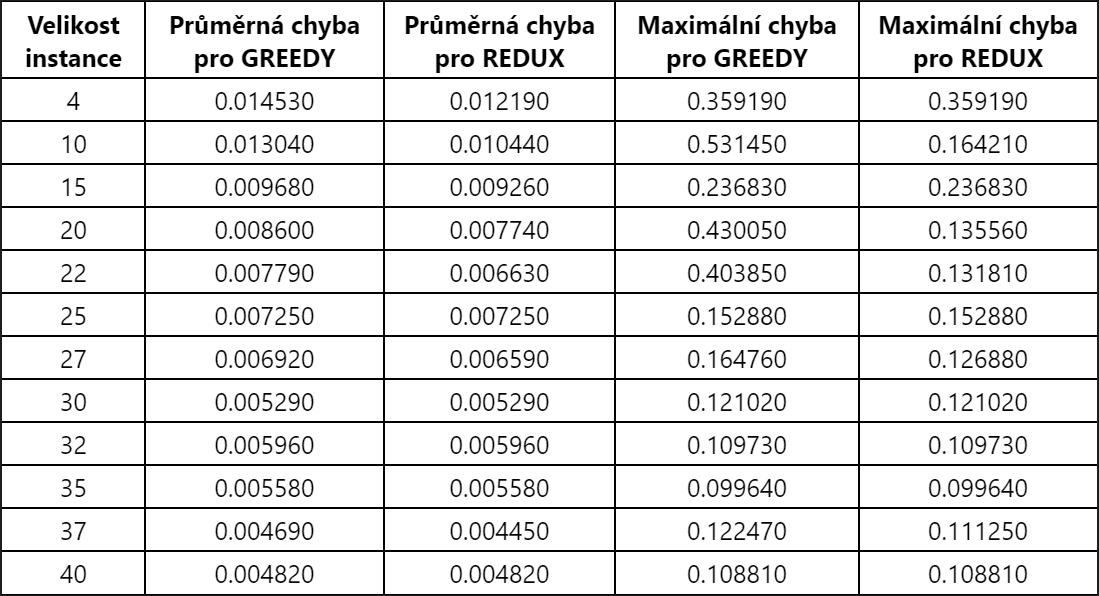
Testovací sada: NK

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40





Data:



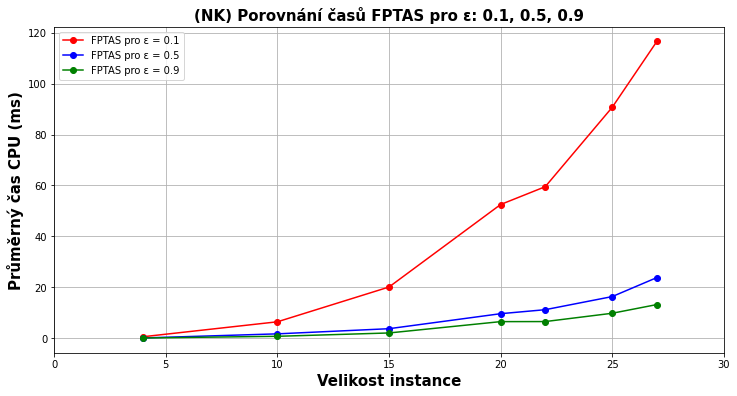
Komentář autora experimentu: Z uvedených výsledků lze vyvodit, že REDUX (jako vylepšený GREEDY) je skutečně buď stejně účinný, nebo dokonce lepší než GREEDY. To lze zpozorovat zejména u maximálních chyb, kdy je u některých velikostí dosaženo opravdu velkých rozdílů.

**1C**

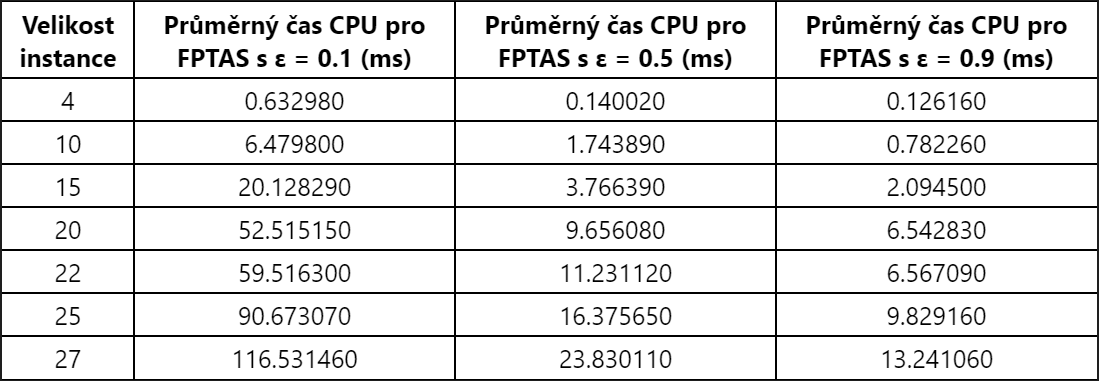
Algoritmus: FPTAS

Testovací sada: NK

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27



Data:



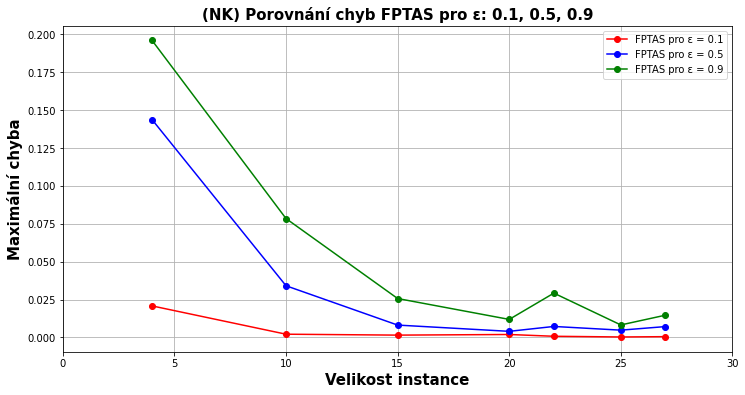
Komentář autora experimentu: Parametr ε označuje přesnost, která je horní hranicí pro maximální chyby. Čím větší je tato přesnost, tím menší je hodnota ε a tím menší jsou maximální chyby. Výsledky také dokumentují, že čím větší je přesnost, tím větší je náročnost a výpočetní výkon algoritmu tak klesá.

**1D**

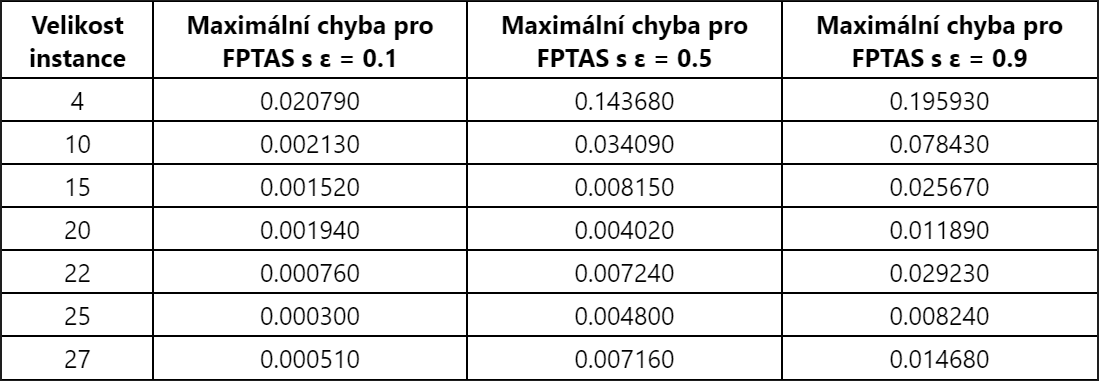
Algoritmus: FPTAS

Testovací sada: NK

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27

****

Data:



Komentář autora experimentu: Přiložené výsledky dokumentují, že s větší přesností je potenciální rozsah maximální chyby menší.

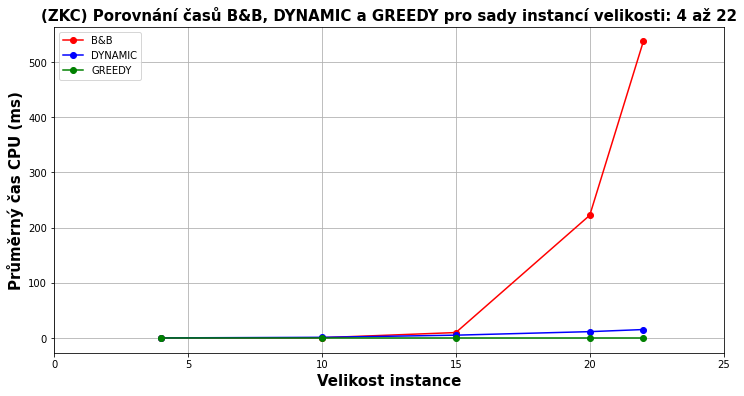
Předchozí sada čtyř experimentů byla opakována i pro testovací sadu ZKC, jak lze číst dále…

**2A**

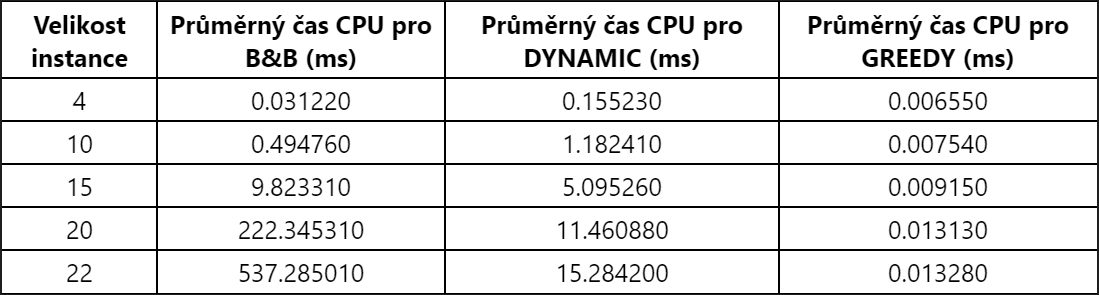
Algoritmy: B&B, DYNAMIC, GREEDY

Testovací sada: ZKC

Velikosti instancí: 4, 10, 15, 20, 22



Data:

****

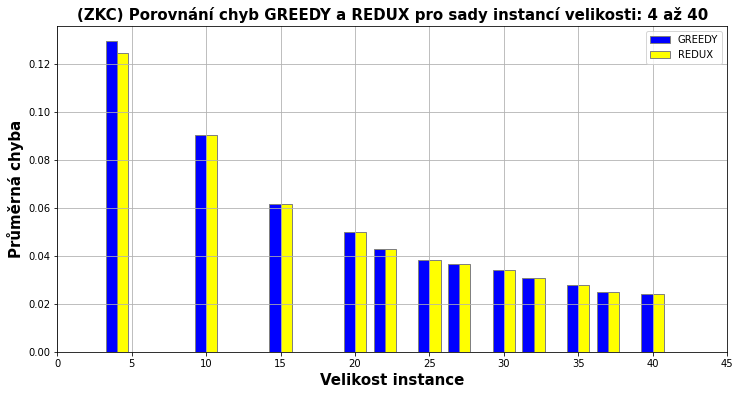
Komentář autora experimentu: Na rozdíl od předchozí sady (tj. NK) lze zpozorovat ztížení výpočtu a zvýšení průměrných časů CPU. Z tohoto důvodu byly měřeny sady instancí velikosti maximálně 22 (oproti 27 v NK). I zde je dynamická verze pro menší velikosti horší než B&B.

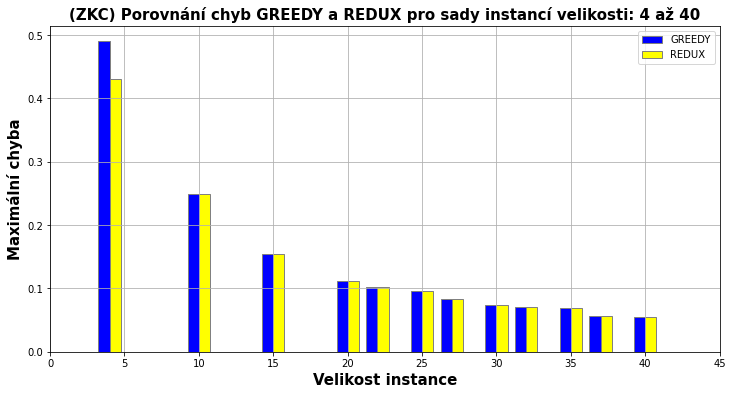
**2B**

Algoritmy: GREEDY, REDUX

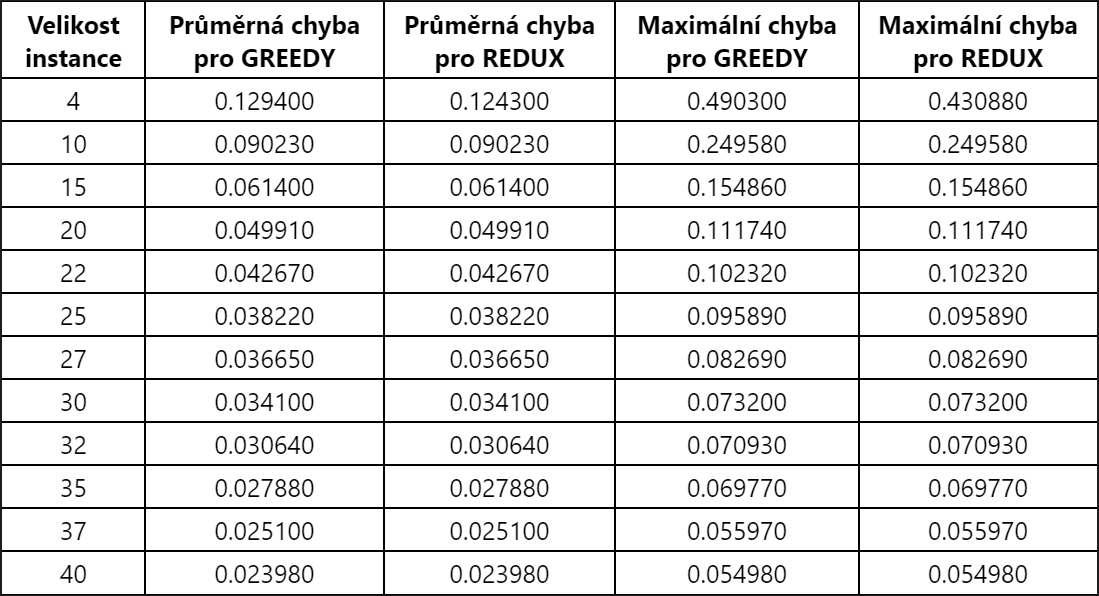
Testovací sada: ZKC

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40

****

****

Data:

****

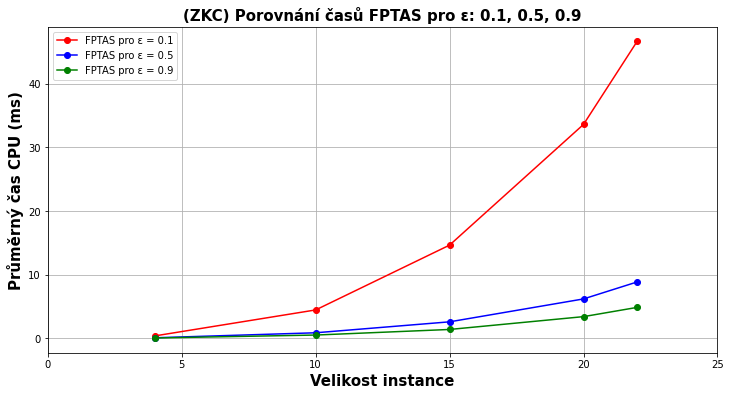
Komentář autora experimentu: Oproti předchozí sadě (tj. NK) algoritmus REDUX zcela ztrácí svůj účinek (tzn. že všechna řešení získaná algoritmem GREEDY jsou hodnotnější než nejcennější věc s přijatelnou váhou v batohu, a tudíž je zde REDUX zcela nevýznamný). Dalším rozdílem je zvýšení průměrné chyby.

**2C**

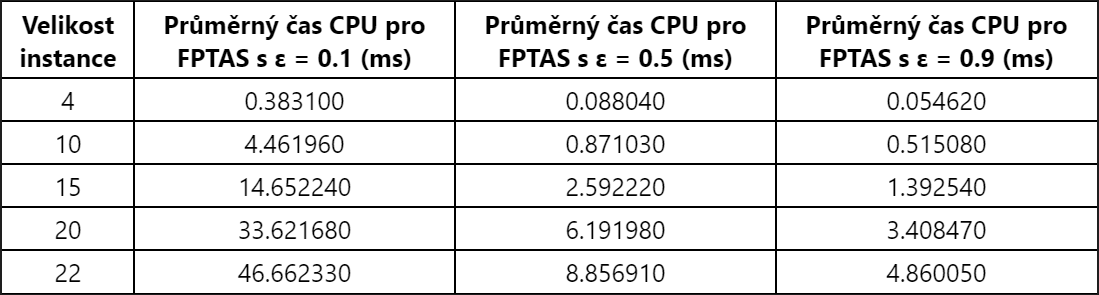
Algoritmus: FPTAS

Testovací sada: ZKC

Velikosti instancí: 4, 10, 15, 20, 22

****

Data:

****

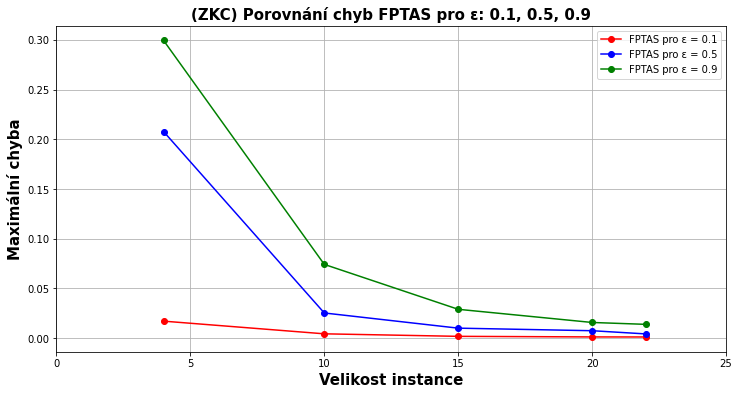
Komentář autora experimentu: Bez komentáře.

**2D**

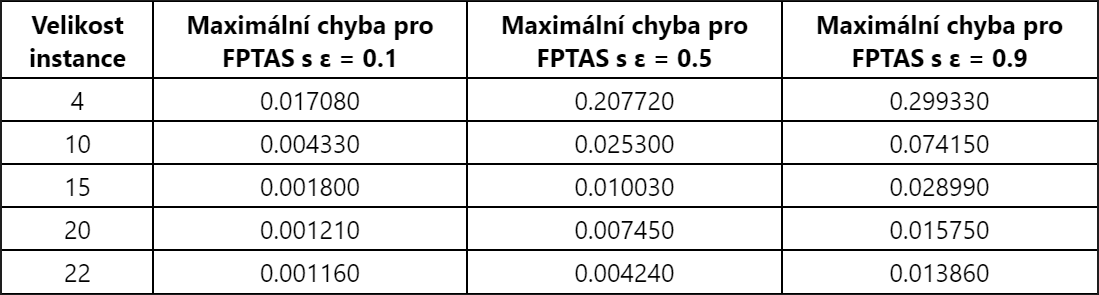
Algoritmus: FPTAS

Testovací sada: ZKC

Velikosti instancí: 4, 10, 15, 20, 22

****

Data:

****

Komentář autora experimentu: Bez komentáře.

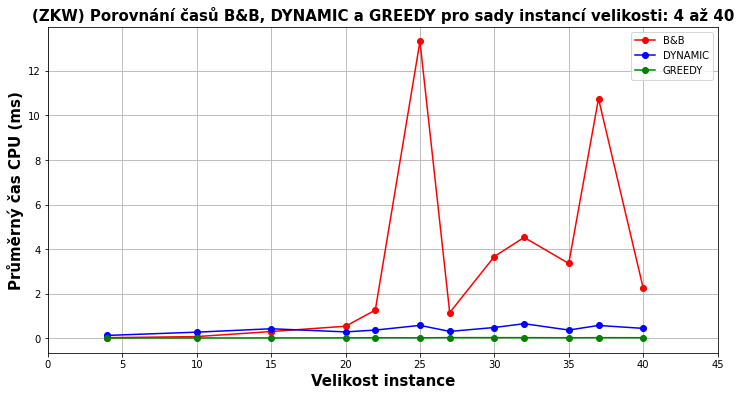
Předchozí sada čtyř experimentů byla opakována i pro testovací sadu ZKW, jak lze číst dále…

**3A**

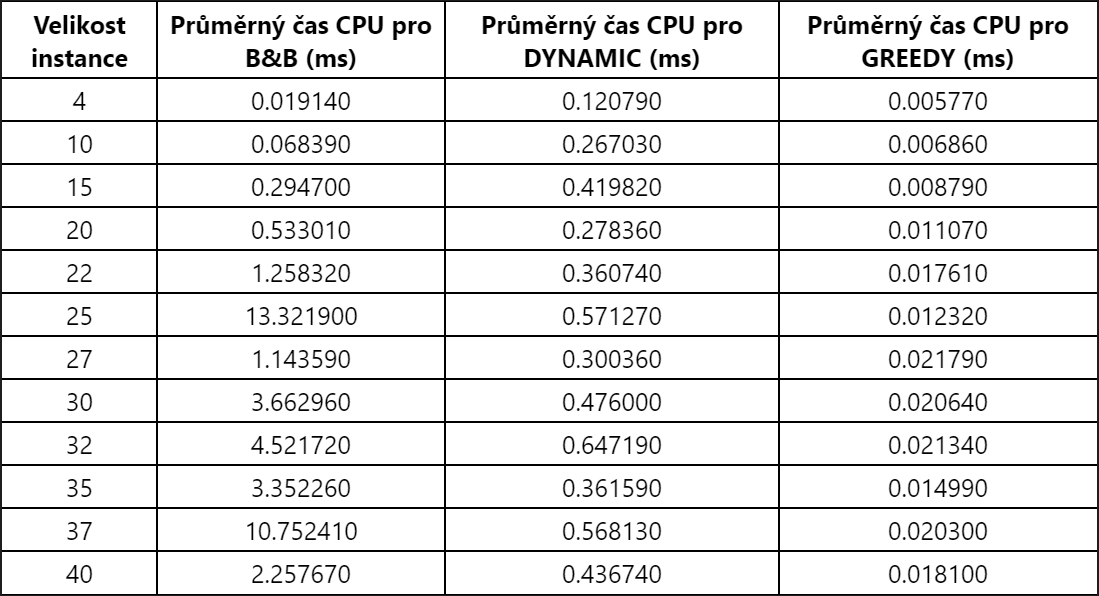
Algoritmy: B&B, DYNAMIC, GREEDY

Testovací sada: ZKW

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40

****

Data:

****

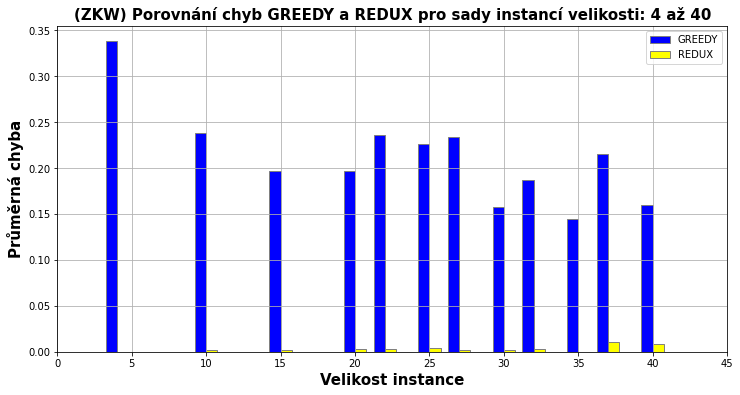
Komentář autora experimentu: Oproti dvěma předchozím sadám (tj. NK a ZKC) je tato sada převratně rychlejší. Uvedený graf a výsledky je ovšem potřeba brát s rezervou. U sad NK a ZKC je pro každou velikost k dispozici právě 500 instancí, přes které se v experimentech počítají průměrné/maximální chyby a rychlosti. V sadě ZKW je pro každou velikost k dispozici vždy jiný počet instancí (většinou menší). U výše uvedeného grafu tudíž nelze brát tvar křivek natolik významně, jako umístění jednotlivých bodů. Ta pak odpovídají předpokladům (např. body B&B mají od velikosti instance 20 (včetně) vždy vyšší hodnotu než body DYNAMIC).

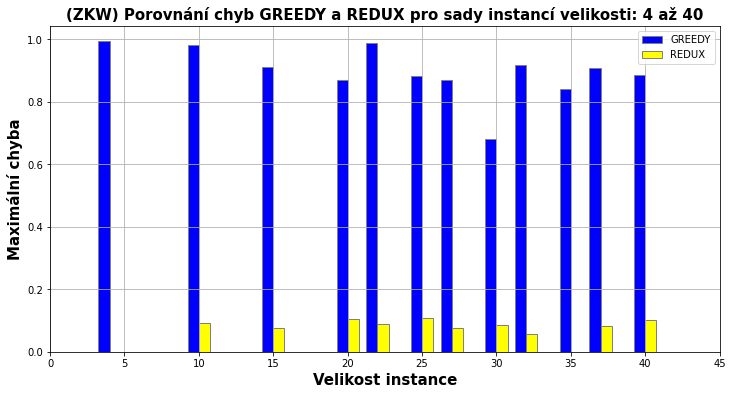
**3B**

Algoritmy: GREEDY, REDUX

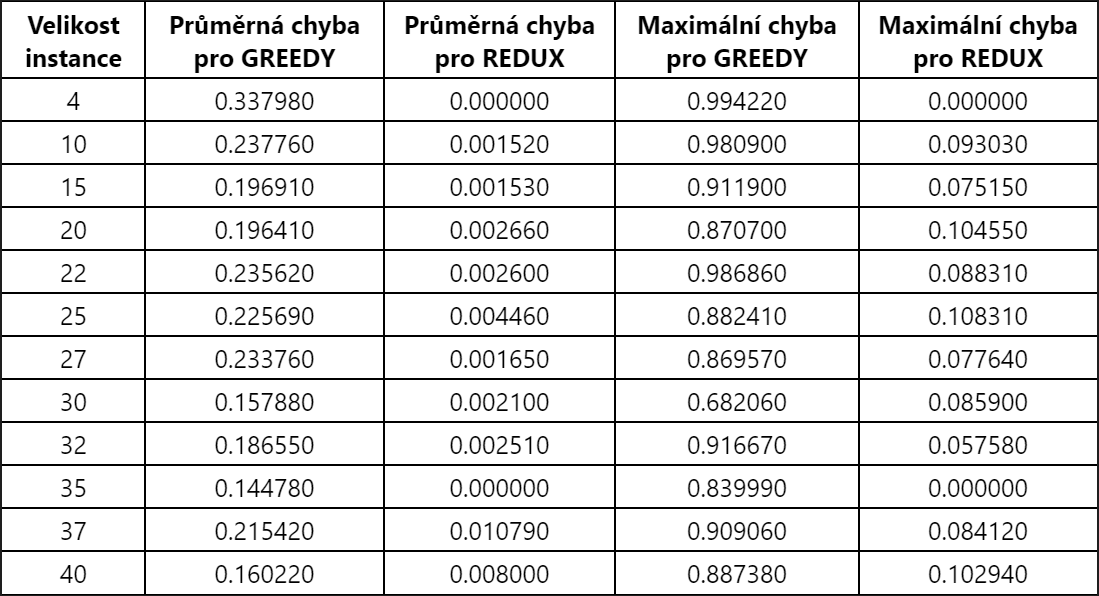
Testovací sada: ZKW

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40

****

****

Data:

****

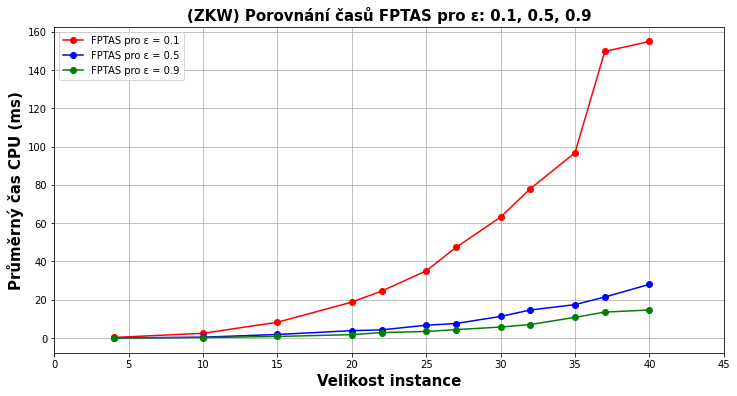
Komentář autora experimentu: Na rozdíl od sady ZKC je zdejší účinek algoritmu REDUX velmi významný a očividný. Řešení, které nalezne algoritmus GREEDY, je tak ve většině případů špatné a REDUX se uplatní jako kvalitnější heuristika.

**3C**

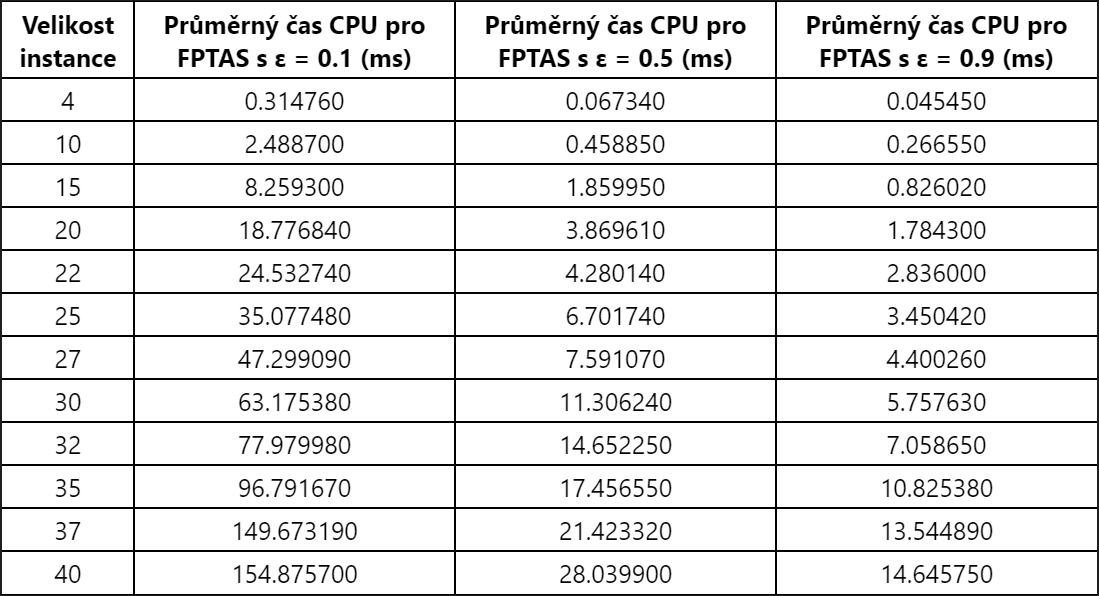
Algoritmus: FPTAS

Testovací sada: ZKW

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40

****

Data:

****

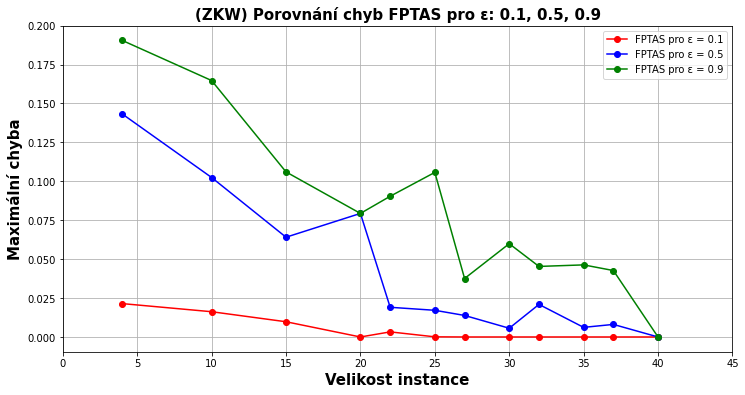
Komentář autora experimentu: Bez komentáře.

**3D**

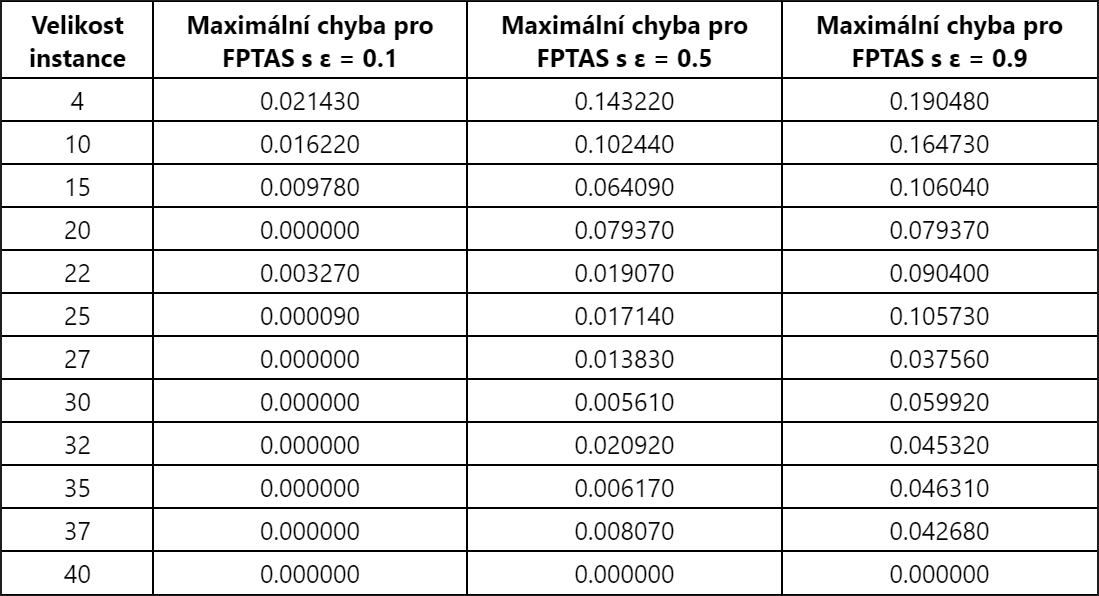
Algoritmus: FPTAS

Testovací sada: ZKW

Velikosti instancí: 4, 10, 15, 20, 22, 25, 27, 30, 32, 35, 37, 40

****

Data:

****

Komentář autora experimentu: Pro některé přesnosti a velikosti instance je výpočet již zcela přesný, a tudíž lze na některých místech v tabulce zpozorovat nulové hodnoty.

**ZÁVĚR**

Z výše uvedených výsledků, grafů, tabulek a komentářů vyplývá, že závislosti času a kvality na velikosti instance odpovídají předpokladům. Výpočetní časy se postupně snižují od B&B přes DYNAMIC (dle váhy) až po GREEDY. GREEDY a REDUX jsou sice rychlé heuristiky, avšak na rozdíl od B&B či DYNAMIC vykazují chybovost. Ta je u REDUX nižší nebo stejná jako u GREEDY. REDUX je tedy lepší v kritériu kvality. Z hlediska jednotlivých sad je ZKC výpočetně náročnější než NK. Projevuje se také větší chybovost, a navíc heuristika REDUX zcela ztrácí význam (kvůli stejné chybovosti jako u heuristiky GREEDY). Sada ZKW je ze všech sad nejméně náročná, což je nejspíše způsobeno různými (a převážně i nižšími) počty instancí pro dané velikosti. Kvalita REDUX je zde zcela opačná a lepší oproti sadě ZKC. Taktéž závislost času a kvality na zvolené přesnosti u algoritmu FPTAS odpovídá předpokladům. S rostoucí přesností se snižuje chybovost, avšak zvyšuje výpočetní náročnost. S klesající přesností se zvyšuje chybovost, avšak snižuje výpočetní náročnost. Maximální chyby jsou vždy shora omezeny stanovenou přesností ε, kterou lze zadat samotnému algoritmu.