Jan Klicpera

NI-KOP – Domácí Úkol 1

1. Specifikace úlohy

Úkolem je naprogramovat řešič *rozhodovací 0/1 verze problému batohu*, který umožní problém řešit hrubou silou, nebo metodou větví a hranic. Dalším úkolem je experimentálně vyhodnotit závislost výpočetní složitosti na velikosti instance za pomocí poskytnutých datových sad.

Pro rozhodovací 0/1 verzi problému batohu jsou zadány následující hodnoty:

- $n \in \mathbb{N}$ (počet věcí)
- $M \in \mathbb{N}_0$ (kapacita batohu)
- $B \in \mathbb{N}_0$ (minimální vyžadovaná cena věcí v batohu)
- $V = \{v_1, v_2, \dots, v_n\}$ (hmotnosti věcí)
- $C = \{c_1, c_2, \dots, c_n\}$ (ceny věcí)

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

- $v_1x_1 + v_2x_2 + \cdots + v_nx_n \leq M$ (aby batoh nebyl přetížen)
- $c_1x_1 + c_2x_2 + \cdots + c_nx_n \geq B$ (aby cena věcí v batohu byla větší, nebo rovna než minimální vyžadovaná cena)

2. Implementace řešení

Pro implementaci byl zvolen jazyk C#, výstupem je konzolová aplikace. Aplikace je plně konfigurovatelná za pomocí přepínačů, chybné vstupy jsou ošetřeny. Aplikace obsahuje dvě povinné verze algoritmu pro řešení a tři experimentální.

Samotné algoritmy byly implementovány rekurzivně, metoda řešení problému bude proto v následujícím textu označována jako procházení binárního stromu – v každém uzlu se rozhodujeme, zda další předmět v pořadí do batohu přidáme, či nepřidáme. Listy stromu tedy reprezentují možné kombinace složení batohu. Počet uzlů stromu je závislý na *počtu věcí* (n). Pro velikost vstupu n lze sestavit binární strom s $n + 1$ hladinami, celkový počet uzlů stromu je tedy

$$2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$$

Metoda brute force

Tento postup nijak neoptimalizuje procházení stromu. pokud nebylo doposud nalezeno řešení, tak je rekurzivní výpočet zastaven pouze v listech stromu. Pokud je kdykoliv během výpočtu nalezena konfigurace, která splňuje zadané podmínky na maximální váhu a minimální cenu, pak je běh algoritmu ukončen s kladným výsledkem. Takto nalezená konfigurace nemusí být spočtena úplně (není nutné procházet až do listu), jelikož je z této částečné konfigurace vždy možné vytvořit úplnou validní konfiguraci (vynecháním všech zbylých předmětů). V případě, kdy řešení neexistuje je nutné projít celý strom, tedy $2^{n+1} - 1$ uzlů.

Metoda větví a hranic

Tato metoda umožňuje zredukovat počet navštívených uzlů stromů pomocí vhodného zastavení rekurzivního výpočtu v případech, kdy již není možné sestrojit konfiguraci, která by splňovala podmínky. Tyto případy jsou:

- Váha věcí v batohu přesáhla maximální kapacitu batohu.
- Aktuální cena věcí v batohu + součet cen dosud „nenavštívených“ věcí je menší než minimální požadovaná cena batohu.

První případ lze odhalit jednoduše porovnáním váhy přidaných věcí s kapacitou batohu. Druhý případ lze odhalit také jednoduše, pokud si budeme udržovat pomocnou proměnnou, která nám bude určovat informaci o maximální ceně, kterou můžeme ještě z dané částečné konfigurace dosáhnout. Pokud při rekurzivním zanoření předmět nepřidáme, pak od této proměnné odečteme cenu nepřidaného předmětu. Pokud hodnota proměnné klesne pod minimální vyžadovanou cenu věcí v batohu, pak nemá smysl dále pokračovat ve výpočtu podstromu.

Experimentální metoda větví a hranic se seřazením předmětů

Velikost „ořezaných“ větví procházeného stromu lze optimalizovat pomocí seřazení vstupní instance předmětů podle jejich hodnot (váhy a/nebo ceny). Pokud předměty s největšími hodnotami budou zpracovány jako první, tak budou neplatné konfigurace odhaleny „výše“ ve stromě a bude tím maximalizován počet vrcholů, které není zapotřebí navštěvovat. Jelikož je složitost seřazení pole o velikosti n prvků $O(n\,log(n))$, tak byl počet operací potřebný na seřazení označen jako zanedbatelný a nebyl započítán do výsledného počtu operací. Byly vyzkoušeny tři různé způsoby řazení:

- Řazení podle ceny
- Řazení podle váhy
- Řazení podle ceny a váhy

Řazení dle kombinace ceny a váhy lze realizovat více metodami, bylo jich několik vyzkoušeno, ve výsledcích je prezentována pouze jedna metoda. Tato metoda nejprve převede cenu a váhu předmětů na procentuální hodnotu, která vyjadřuje, do jaké míry tyto hodnoty přispějí k naplnění limitů kladených na váhu a cenu. Tyto hodnoty je zapotřebí vypočítat, aby se cena a váha převedly na „společné“ jednotky (velikost hodnoty vůči limitu kladeného na danou hodnotu).

Pro váhu se tato hodnota spočítá vydělením váhy předmětu kapacitou batohu. Pro i-tý předmět tedy platí:

$$\mathrm{v}ratio_i = \frac{\mathrm{v}_i}{M}$$

Pro cenu je nejprve zapotřebí spočítat maximální součet cen nepřidaných předmětů tak, aby stále byla splněna podmínka kladená na cenu předmětů. Touto hodnotou vydělíme cenu předmětu. Pro i-tý předmět tedy platí:

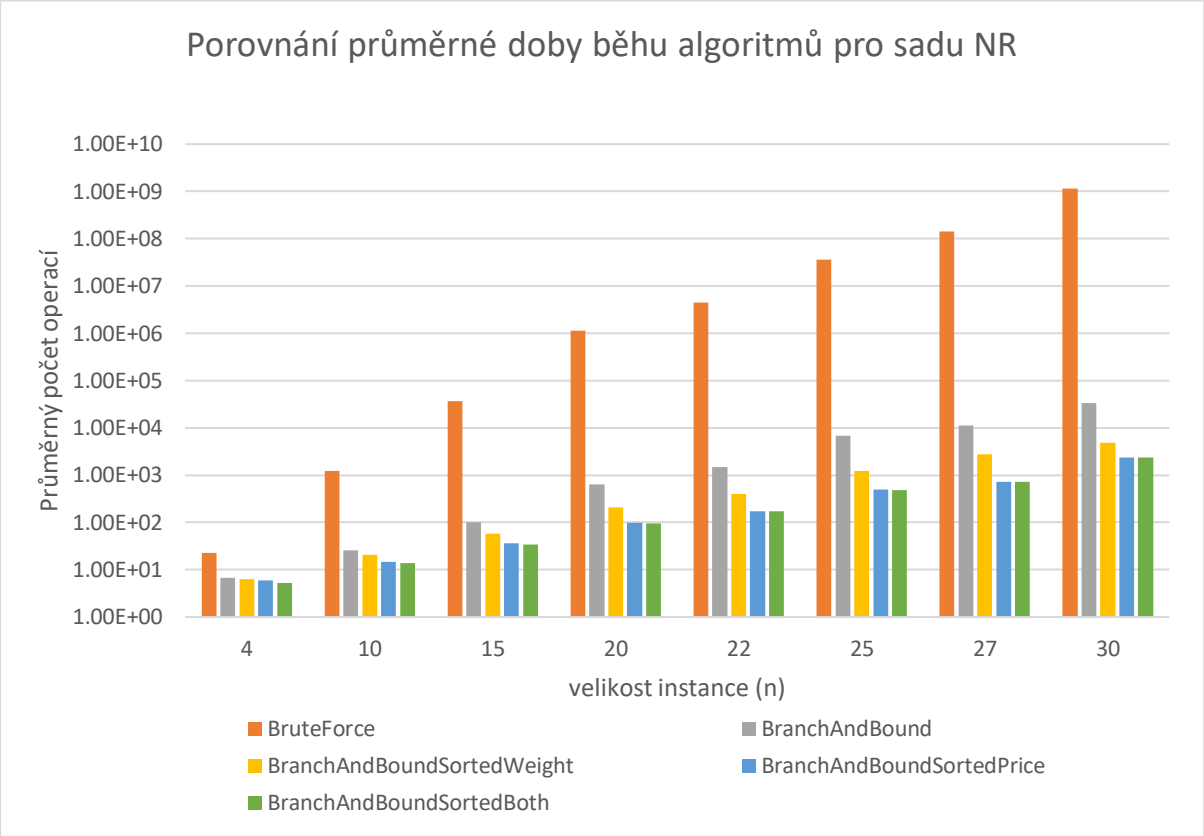
$$cratio_i = \frac{c_i}{\left(\sum_{j=1}^{n} c_\mathrm{j}\right) - B}$$

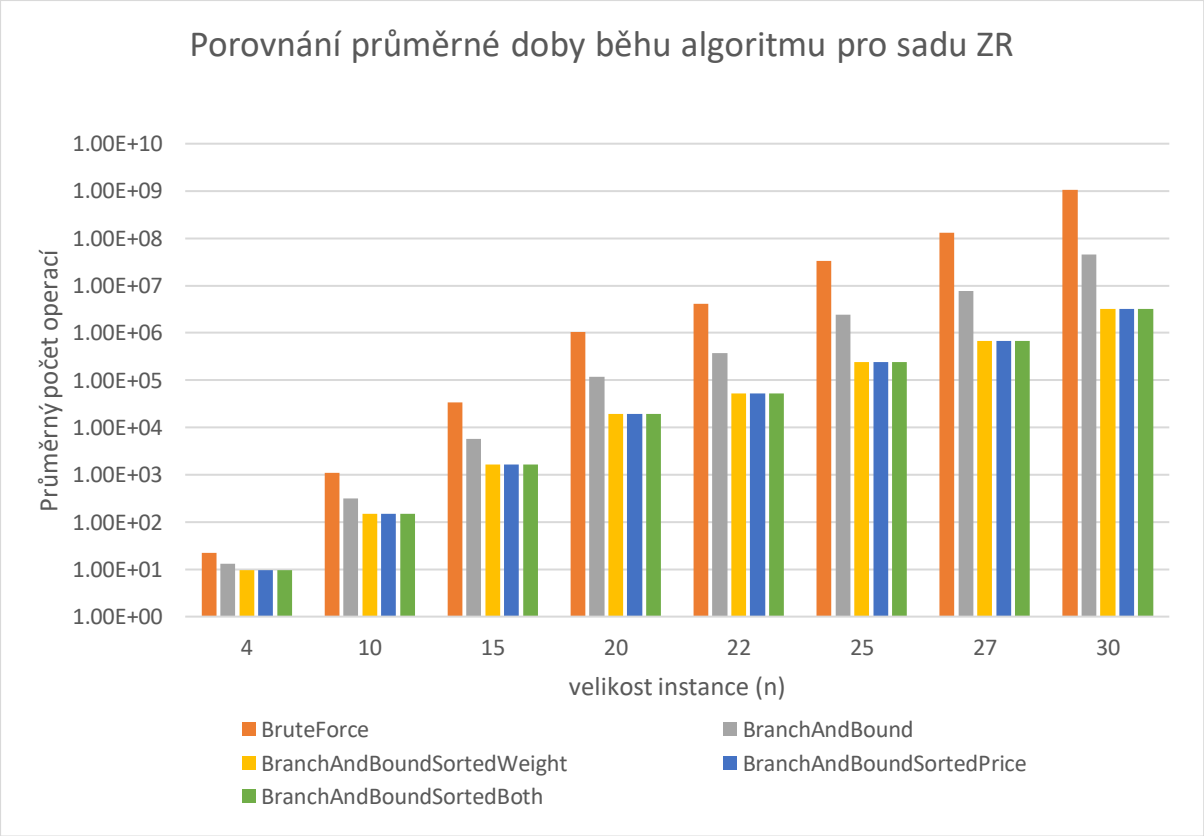
Pozn: Není vyloučeno, aby byl dělenec záporný, či dokonce nulový. V případě, že je záporný, neexistuje žádné řešení. V případě, že je nulový, tak může existovat pouze jedno řešení, kdy do batohu všechny předměty přidáme (pokud nebude porušena podmínka kapacity batohu).

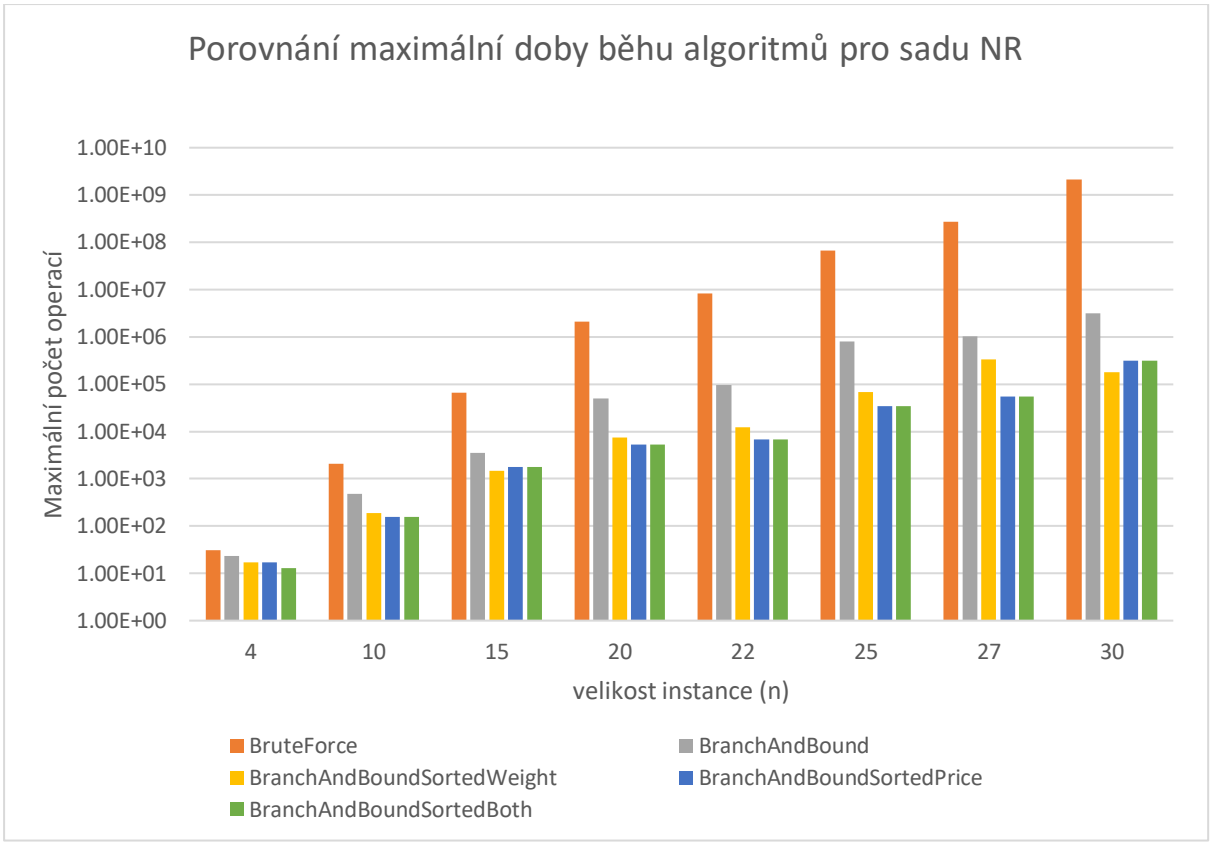
Předměty jsou výsledně seřazeny podle největší hodnoty z každé dvojice $vratio_i$ a $cratio_i$.

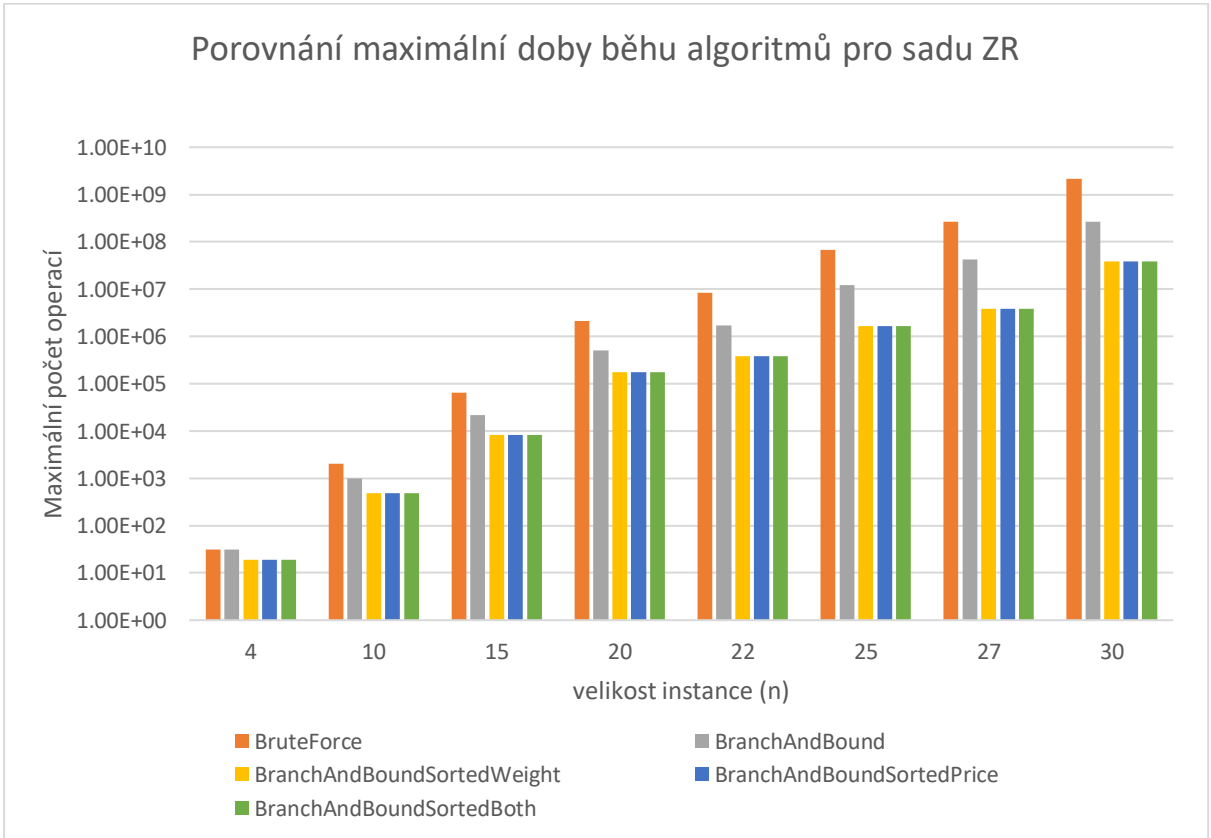
3. Naměřené výsledky

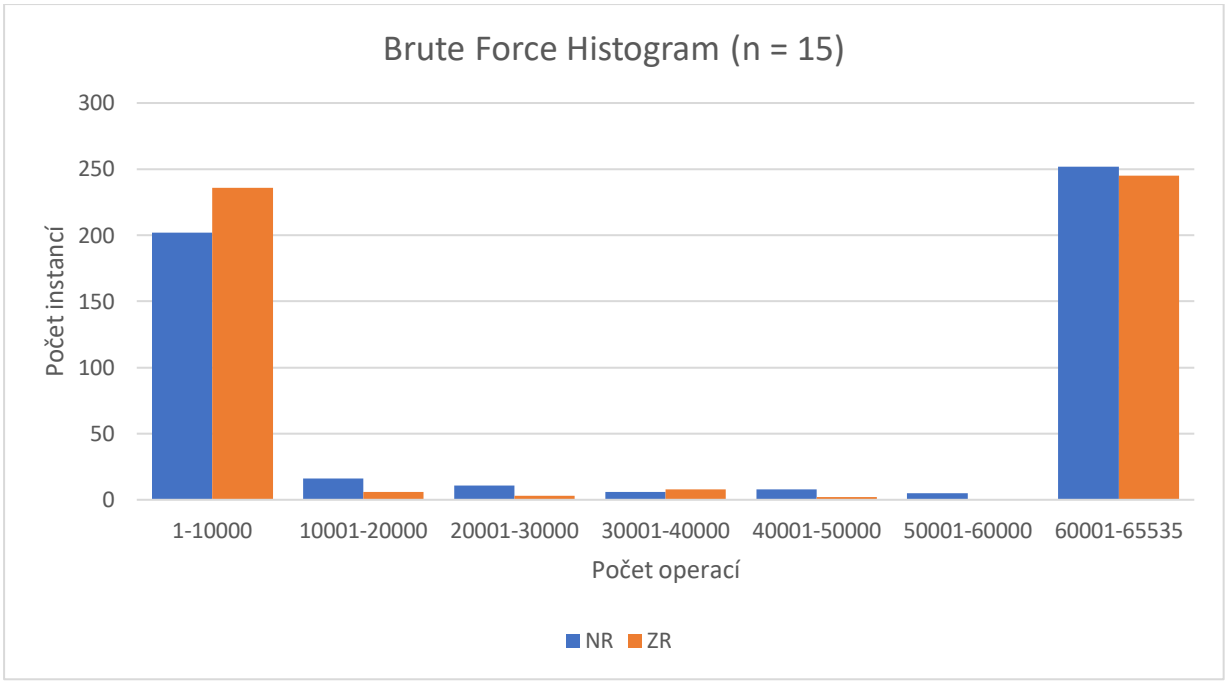
Čas běhu algoritmu byl měřen jako počet volání rekurzivní funkce.

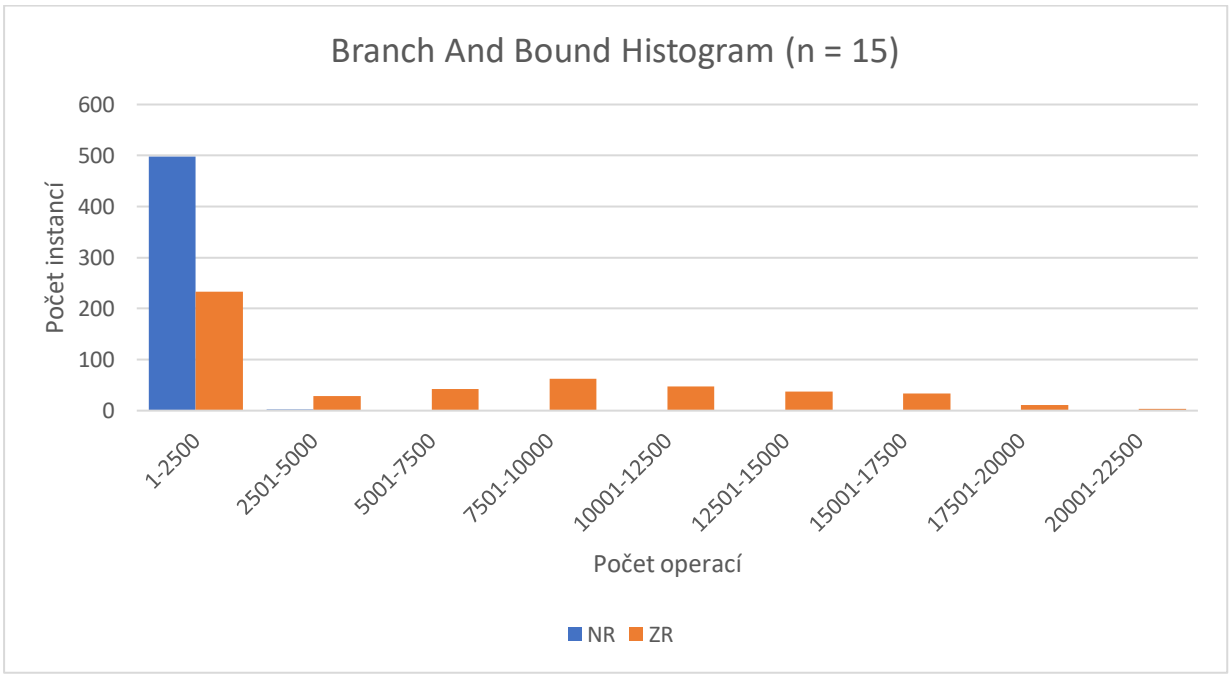





Porovnání maximální doby běhu algoritmů pro sadu NR
Maximální počet operací
1.00E+10
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
4
10
15
20
22
25
27
30
velikost instance (n)
BruteForce
BranchAndBound
BranchAndBoundSortedWeight
BranchAndBoundSortedPrice
BranchAndBoundSortedBoth


Porovnání maximální doby běhu algoritmů pro sadu ZR
Maximální počet operací
1.00E+10
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
4
10
15
20
22
25
27
30
velikost instance (n)
BruteForce
BranchAndBound
BranchAndBoundSortedWeight
BranchAndBoundSortedPrice
BranchAndBoundSortedBoth


Brute Force Histogram (n = 15)
300
250
200
150
100
50
0
Počet instancí
1-10000
10001-20000
20001-30000
30001-40000
40001-50000
50001-60000
60001-65535
Počet operací
NR
ZR


Branch And Bound Histogram (n = 15)
600
500
400
300
200
100
0
Počet instancí
1-2500
2501-5000
5001-7500
7501-10000
10001-12500
12501-15000
15001-17500
17501-20000
20001-22500
Počet operací
NR
ZR

4. Závěr

Z naměřených výsledků je patrné, že metoda větví a hranic umožňuje výrazně snížit nutný počet operací k získání výsledku. Přesto je asymptotická složitost i s využitím této heuristiky stále exponenciální. Navíc mohou existovat instance (viz výsledky sady ZR), které není heuristika schopna příliš dobře optimalizovat (hodnoty zadaných věcí nepřesahují limity v tak velké míře).

Zajímavé byly výsledky při seřazení vstupní instance. Na testovacích sadách bylo téměř vždy efektivnější seřazení podle ceny než seřazení podle váhy. Seřazení podle obou hodnot bylo bohužel pouze velmi nepatrně lepší než seřazení pouze podle ceny.

Na vytvořených histogramech je taktéž patrná nestabilita metody větví a hranic v závislosti na vstupních instancích. Zatímco brute force metoda dosahuje pro obě sady téměř stejné četnosti doby běhu, metoda větví a hranic je postihnuta co se týče doby běhu pro sadu ZR.

Pozn.: doba běhu metody větví a hranic měla pravděpodobně dopadnout ještě hůře pro sadu ZR, během psaní této zprávy jsem si však uvědomil, že předměty zpracovávám v obráceném pořadí, což pravděpodobně naruší některé „zlomyslně" poskytnuté instance.