Experimentální hodnocení kvality algoritmů

1 SPECIFIKACE ÚLOHY

Cílem tohoto úkolu bylo ohodnotit různé algoritmy pro řešení problému batohu. Předmětem hodnocení je především citlivost výpočetní náročnosti jednotlivých algoritmů na vstupní data (na hodnoty), která jsou generována konfigurovatelným generátorem. Je hodnocena také jejich kvalita. Zpráva obsahuje několik grafů znázorňující různé závislosti na datech a jejich objasnění. Více informací viz odpovídající stránka na Eduxu.

2 ROZBOR MOŽNÝCH VARIANT ŘEŠENÍ

Protože v této úloze již nejde o implementaci, bylo pouze třeba se rozhodnout, jakým způsobem provádět měření. Zvolil jsem před zcela ručním spouštěním testů na generovaných datech raději jednoduchý skript, na jehož výstupu dostávám naměřená data oddělená tabulátorem, odkud je stačilo nakopírovat do Excelu za účelem vytvoření tabulek a grafů.

3 POPIS POSTUPU ŘEŠENÍ A ALGORITMU

Před začátkem měření jsem provedl několik zkušebních testů za účelem zjištění, na jakých hodnotách je vhodné zafixovat proměnné generátoru, které právě netestuji.

Také bylo třeba se rozhodnout, jaké algoritmy na co testovat, protože úplně algoritmy nemá smysl testovat na kvalitu a heuristiky (v mém případě) na čas

Rozhodl jsem se pro následující.

Test doby běhu: Dynamické programování, Větve a hranice.

Test kvality: Heuristika dle poměru cena/váha, heuristika dle ceny (hladová).

Testovat kvalitu u Dynamického programování a Větví a hranic nemá smysl, protože jde o úplné algoritmy, které vždy naleznou optimální řešení, pokud existuje.

Testovat dobu běhu u heuristik nemá smysl, protože nezávisle na datech je vždy O(n *log(n)).

Výpočetní náročnost jsem se rozhodl měřit jednoduše jako doposud – čas běhu. Počítat testované stavy by sice bylo vhodnější, ale vzhledem k tomu, že všechny testy proběhly na stejném stroji, se stejně implementovanými datovými struktury, rozdíl by byl minimální a všechny trendy je možné vidět i takto.

4 NAMĚŘENÉ VÝSLEDKY

4.1 HW / SW KONFIGURACE TESTOVACÍHO SYSTÉMU

- CPU Intel® Core™2 Duo; 2.26 GHz, 2.27 GHz
- 4 GB RAM
- OS Windows 8 64-bit
- Java 7

4.2 ZPŮSOB MĚŘENÍ

Čas byl měřen pomocí třídy **ThreadMXBean** a byl průměrován přes všechny instance velikosti 20, kterou jsem zvolil jako dostatečně velkou při stále poměrně rychlém výpočtu.

Relativní chyby heuristik jsem počítal dle vzorce ε = (C(OPT) - C(APX)) / C(OPT) a byly počítány průběžně, ne na finálním součtu cen.

4.3 VÝSLEDKY

Většina tabulek s daty jsou z důvodu velikosti vynechány. Interpretace výsledků je v závěru.

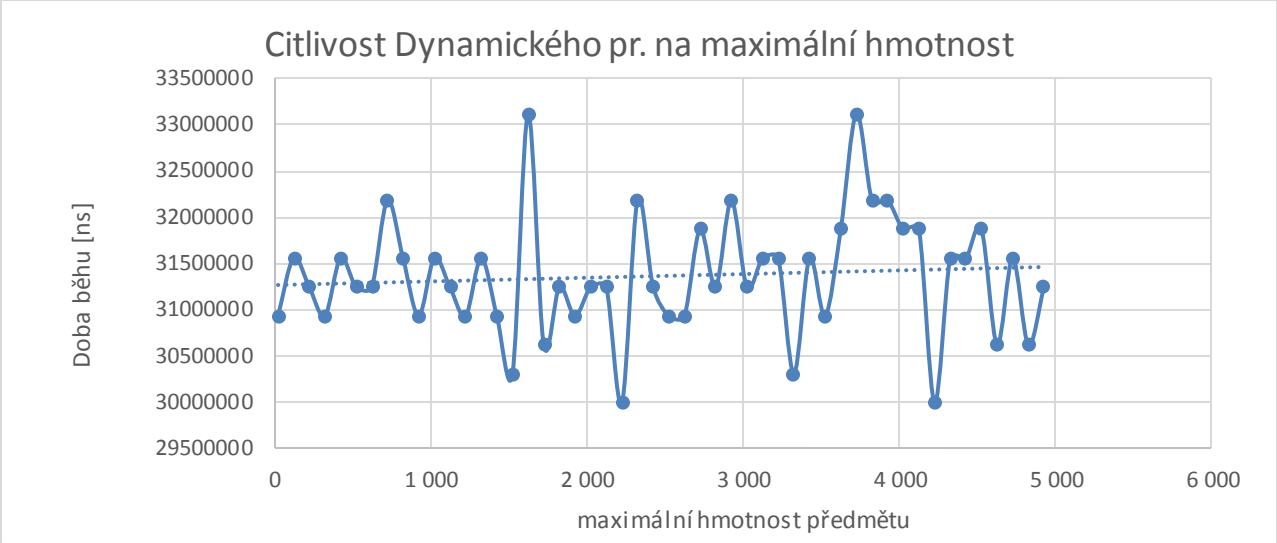
4.3.1 Měření času

4.3.1.1 Vliv maximální hmotnosti

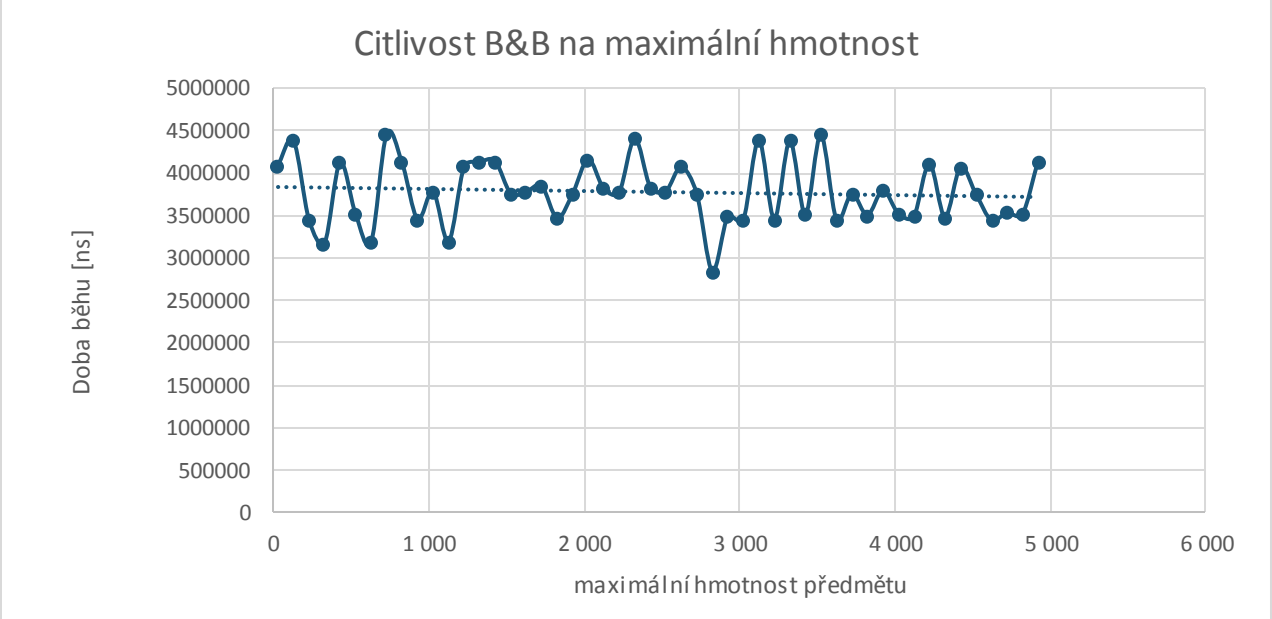
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	**25-5000**
maximální cena věci	200
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.1.1.1 Dynamické programování



4.3.1.1.2 Větve a hranice

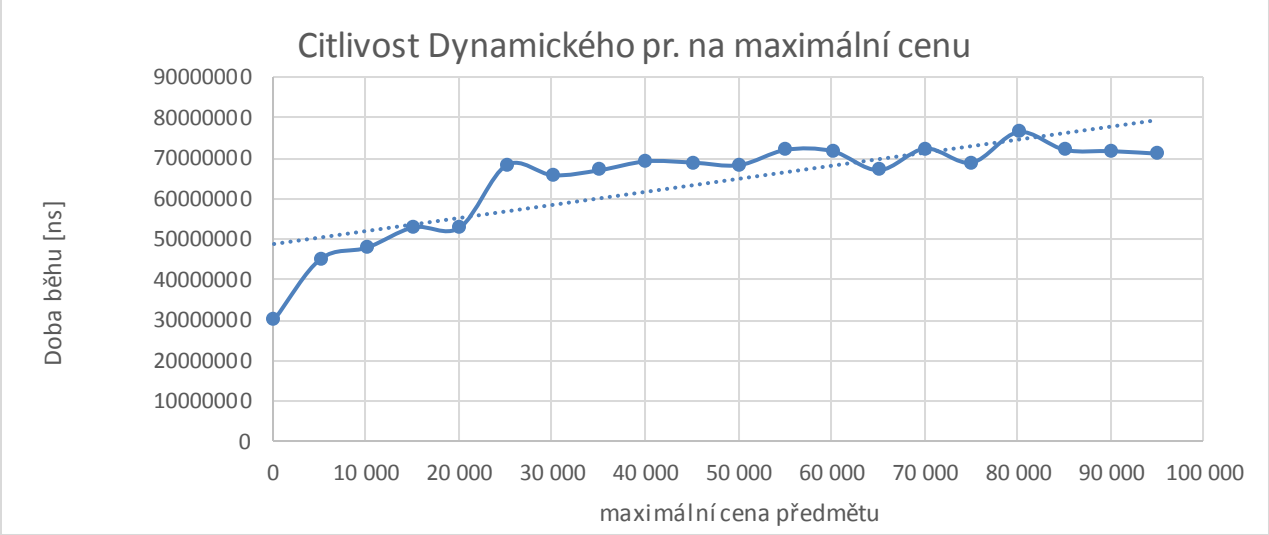


4.3.1.2 Vliv maximální ceny

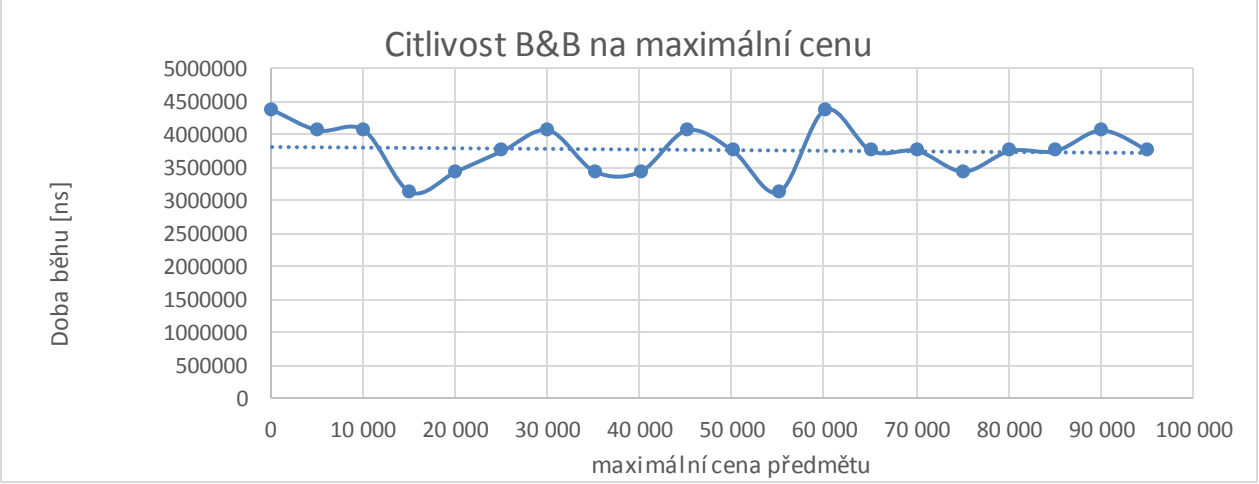
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	1000
maximální cena věci	**100 – 100 000**
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.1.2.1 Dynamické programování



4.3.1.2.2 Větve a hranice

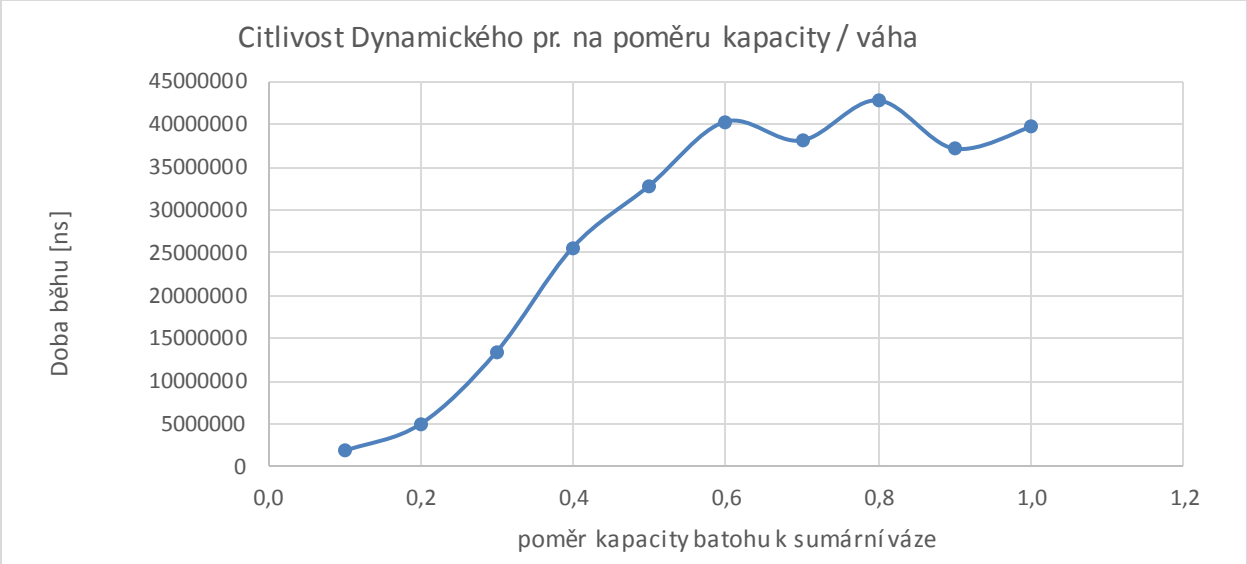


4.3.1.3 Vliv poměru kapacity batohu k sumární váze

Zvolené parametry generátoru:

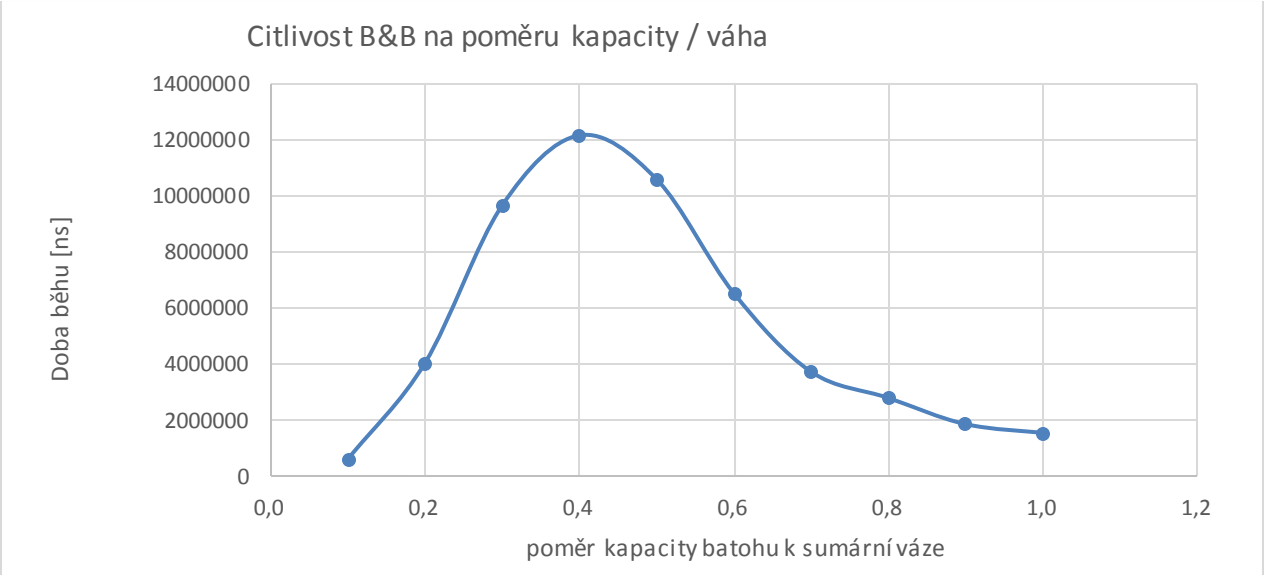
Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	**0.0 – 1.0**
maximální váha věci	1000
maximální cena věci	3000
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.1.3.1 Dynamické programování



4.3.1.3.2 Větve a hranice

poměru kapacity batohu k sumární váze	doba běhu [ns]
0,1	625000
0,2	4062500
0,3	9687500
0,4	12187500
0,5	10625000
0,6	6562500
0,7	3750000
0,8	2812500
0,9	1875000
1,0	1562500

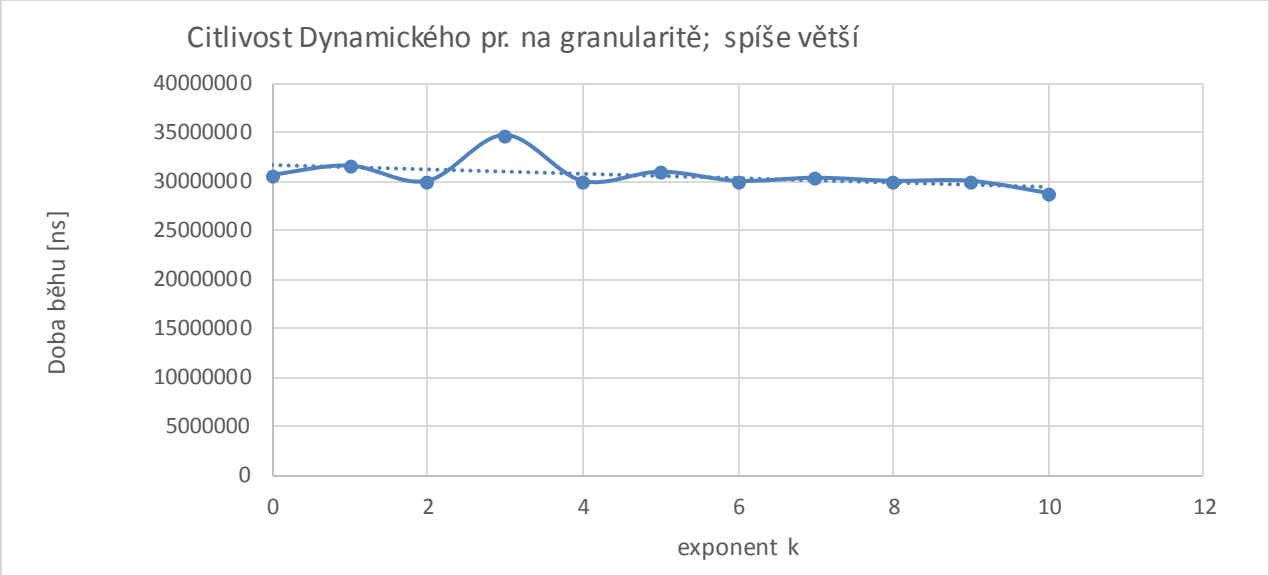


4.3.1.4 Vliv granularity

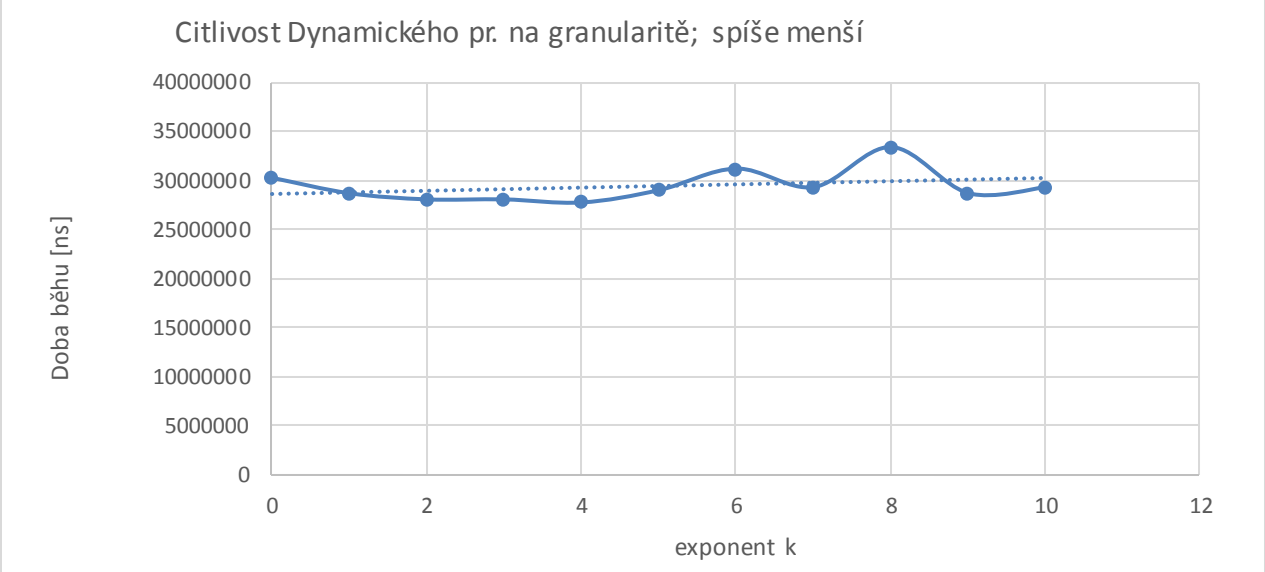
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	30
maximální cena věci	100
exponent k	**0-10**
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	1, -1

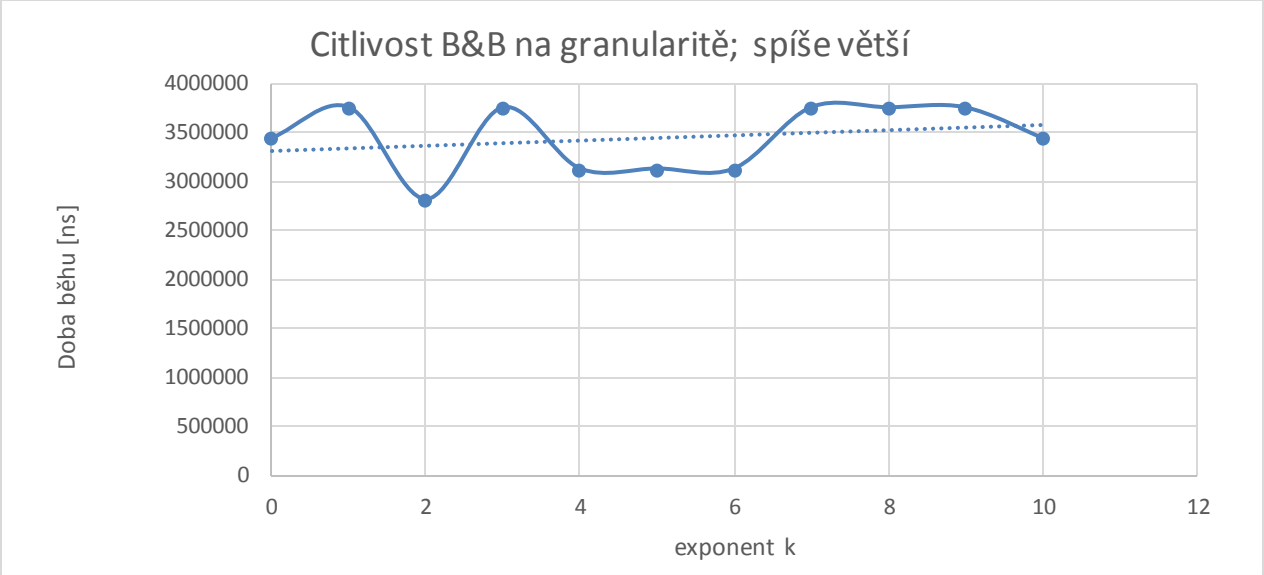
4.3.1.4.1 Dynamické programování pro d=1



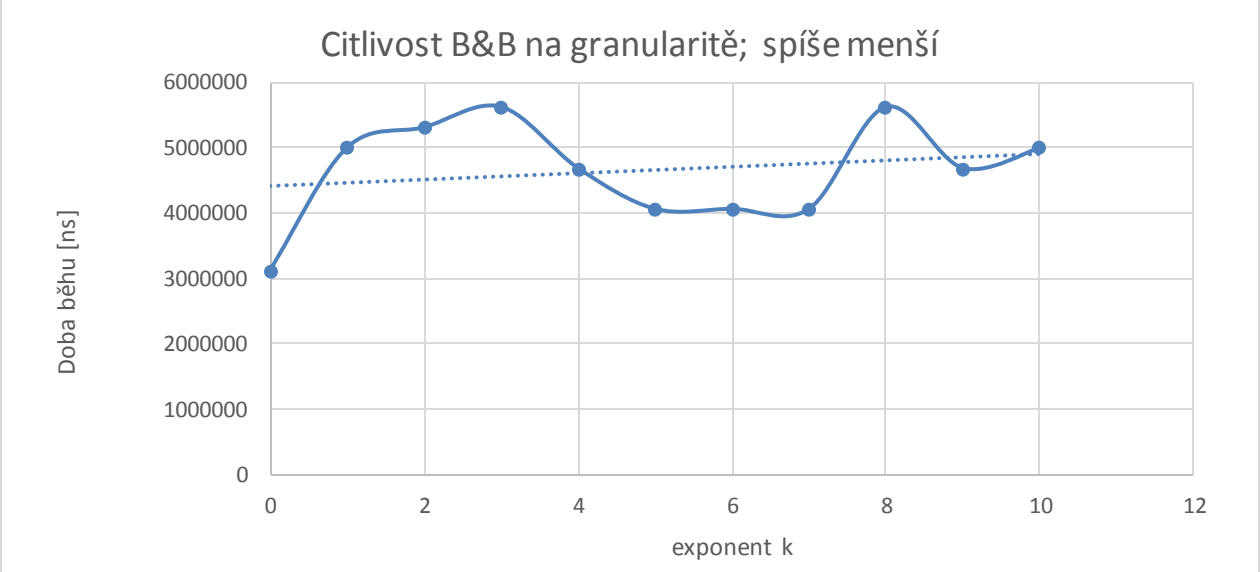
4.3.1.4.2 Dynamické programování pro d=-1



4.3.1.4.3 Větve a hranice pro d=1



4.3.1.4.4 Větve a hranice pro d=-1



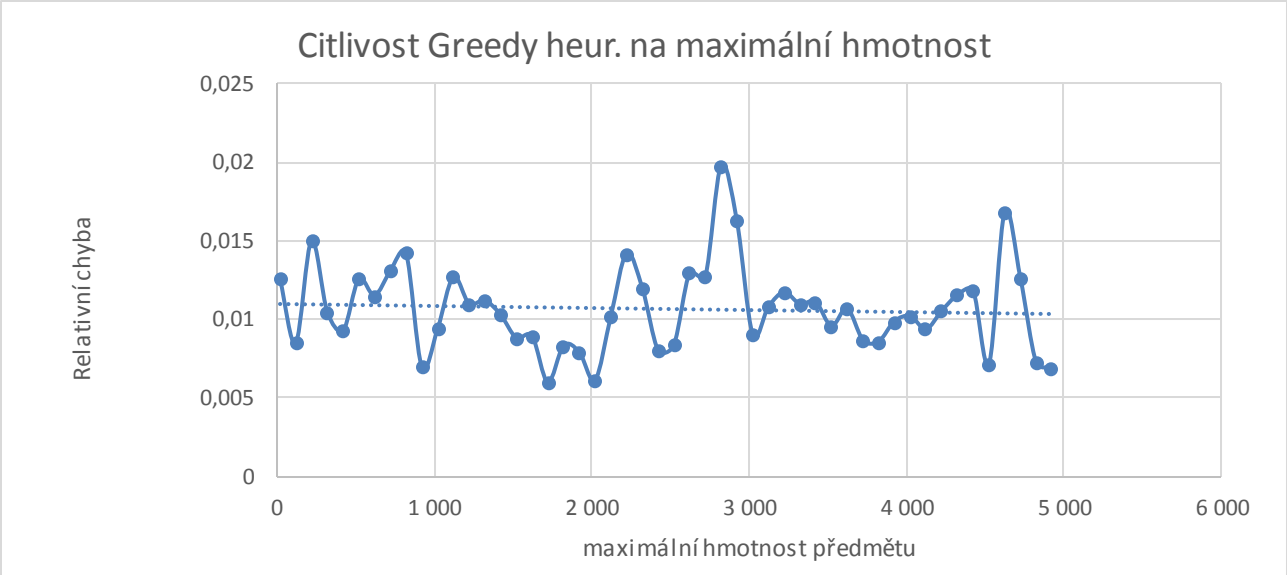
4.3.2 Měření kvality

4.3.2.1 Vliv maximální hmotnosti

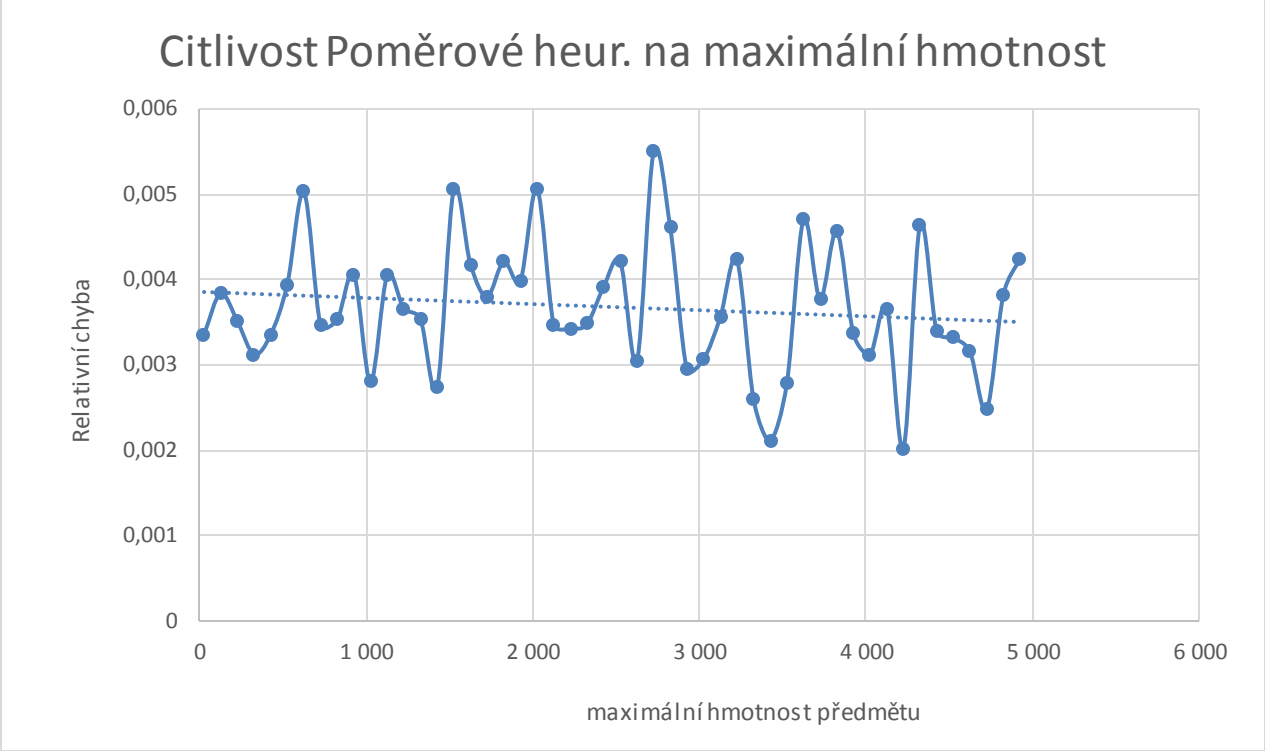
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	**25-5000**
maximální cena věci	200
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.2.1.1 Greedy heuristika



4.3.2.1.2 Heuristika podle poměru cena/váha

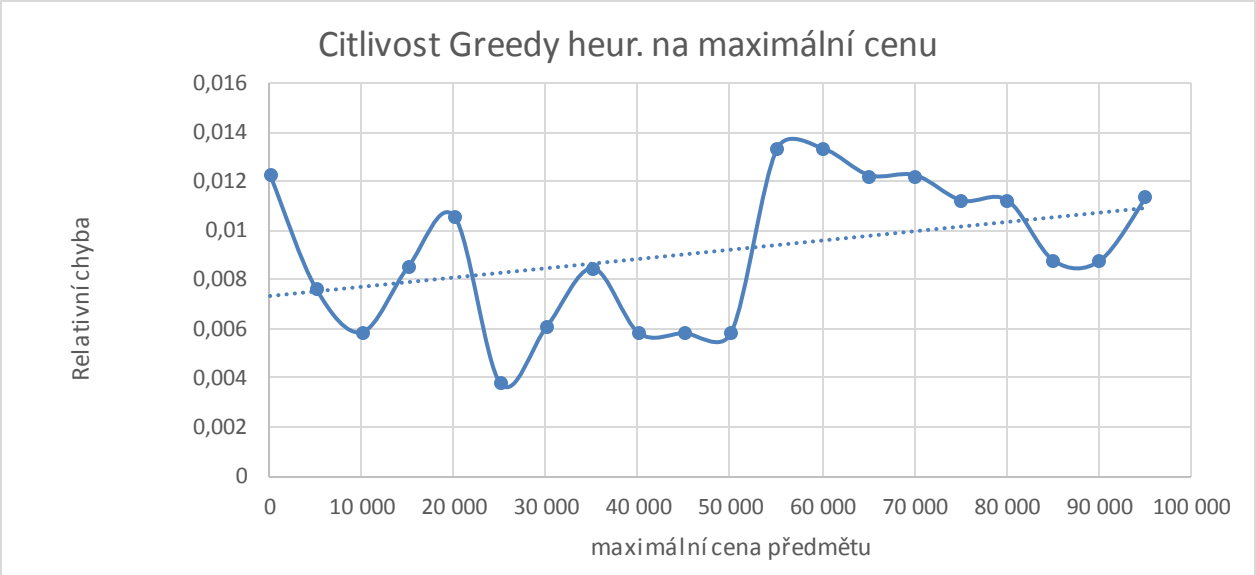


4.3.2.2 Vliv maximální ceny

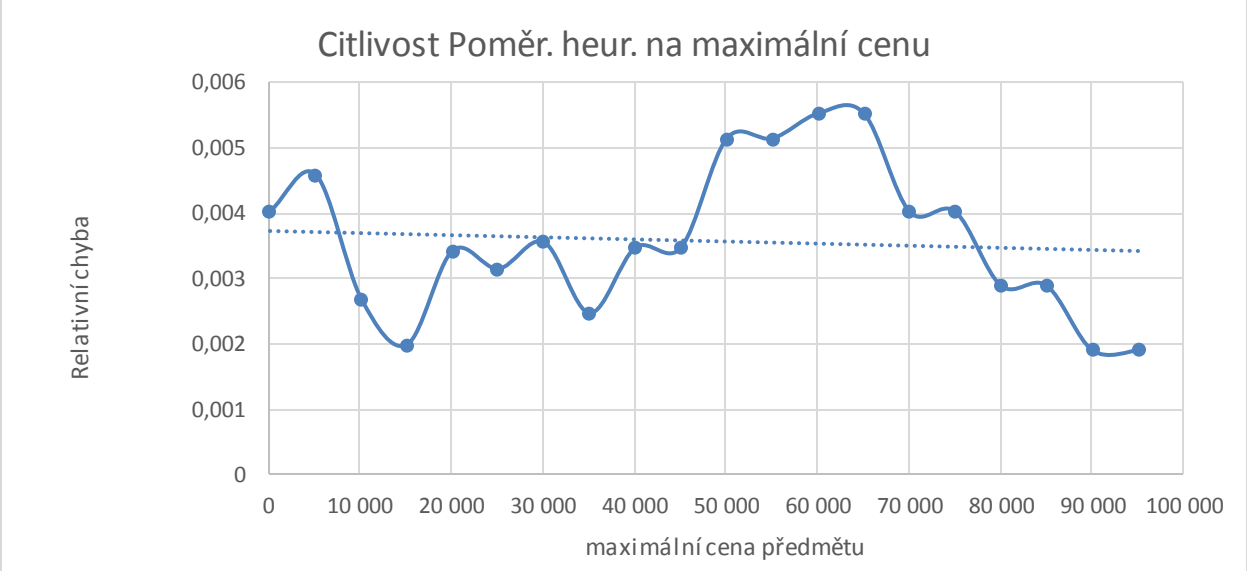
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	1000
maximální cena věci	**100 – 100 000**
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.2.2.1 Greedy heuristika



4.3.2.2.2 Heuristka podle poměru cena/váha

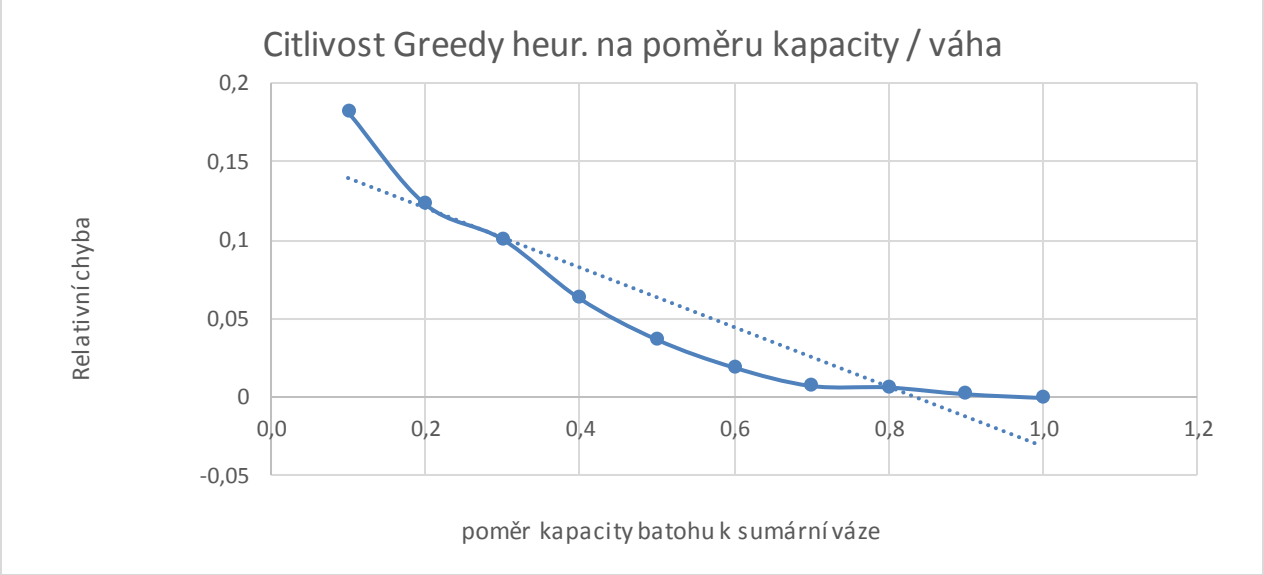


4.3.2.3 Vliv poměru kapacity batohu k sumární váze

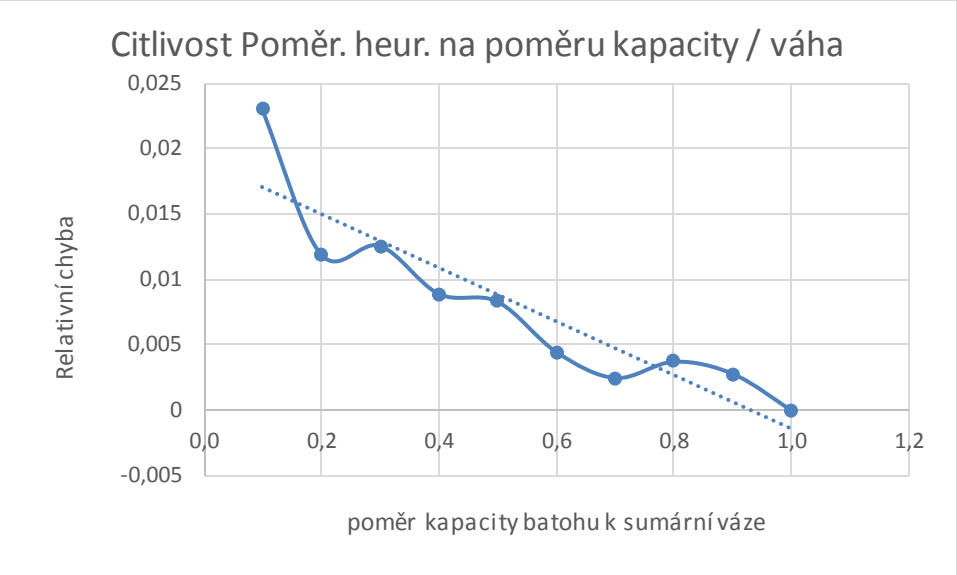
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	**0.0 – 1.0**
maximální váha věci	1000
maximální cena věci	3000
exponent k	1
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	0

4.3.2.3.1 Greedy heuristika



4.3.2.3.2 Heuristka podle poměru cena/váha

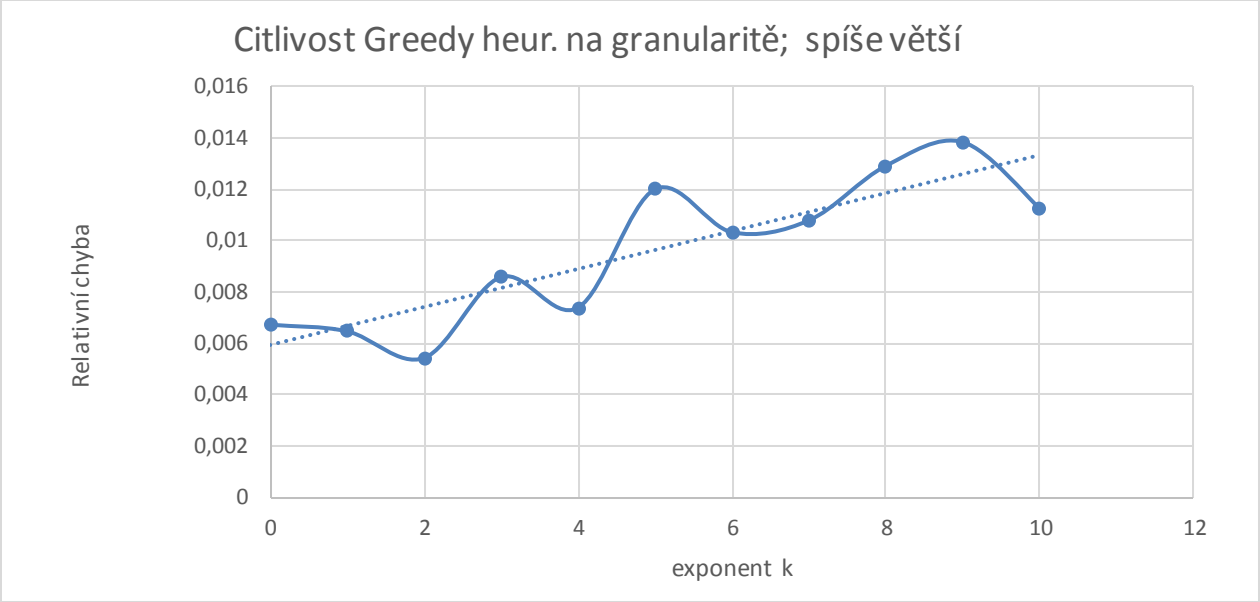


4.3.2.4 Vliv granularity

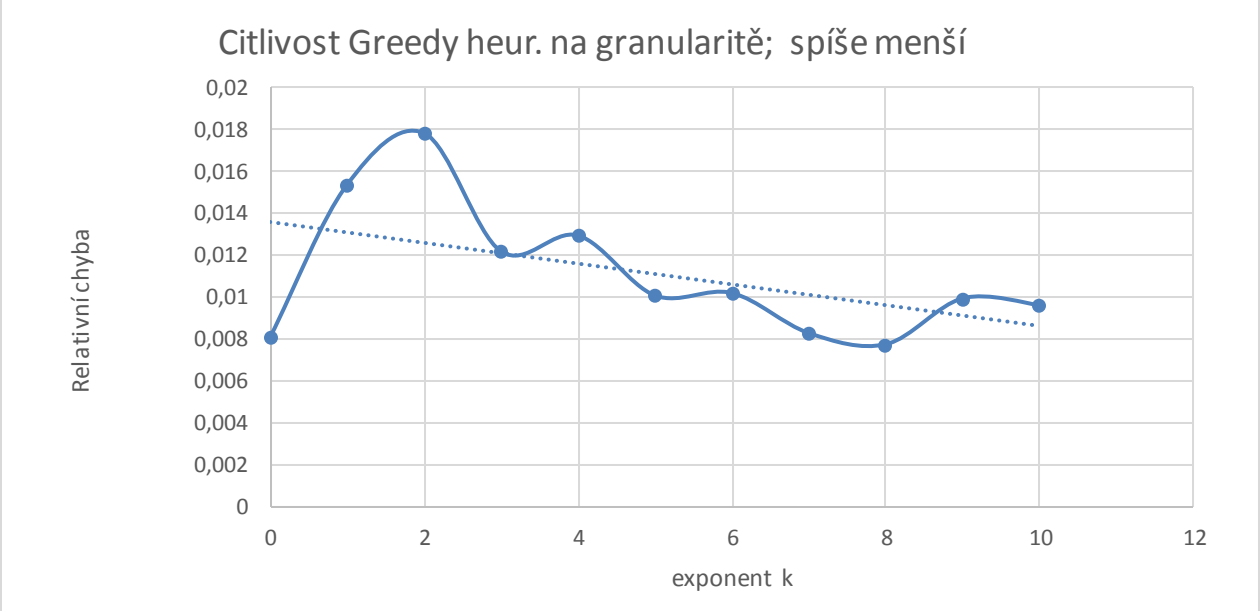
Zvolené parametry generátoru:

Parametr	Hodnota
počet věcí	20
počet instancí	50
poměr kapacity batohu k sumární váze	0,7
maximální váha věci	30
maximální cena věci	100
exponent k	**0-10**
-1..více malých věcí, 1..více velkých věcí, 0..rovnováha	1, -1

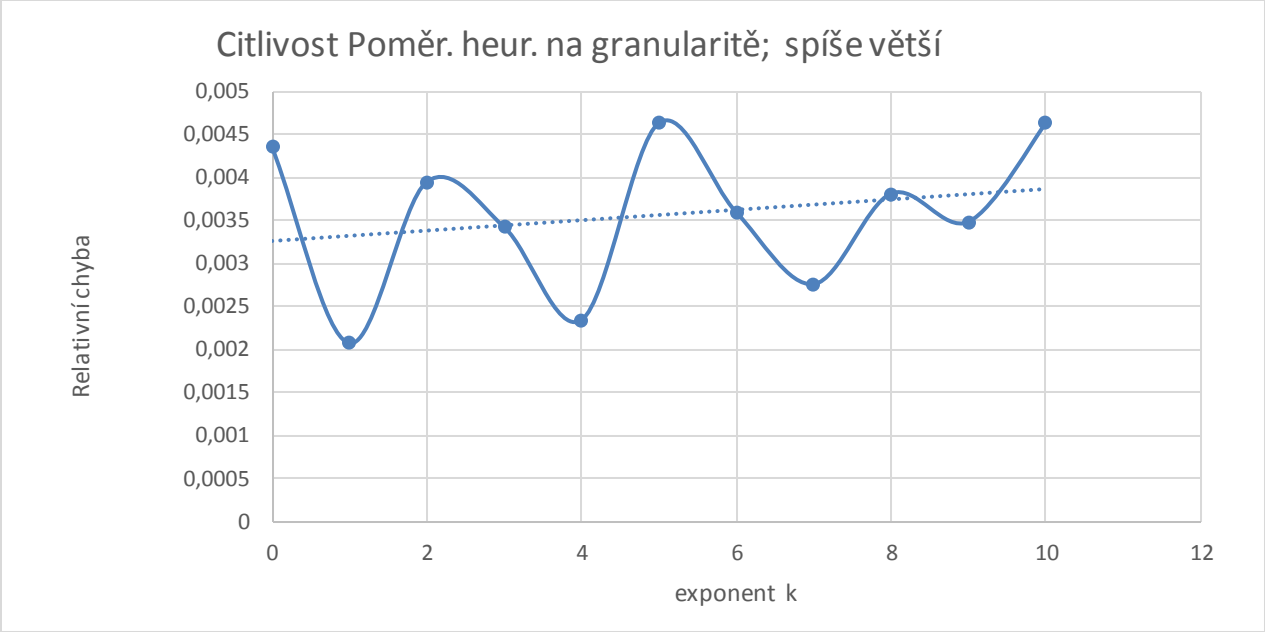
4.3.2.4.1 Greedy heuristika pro d=1



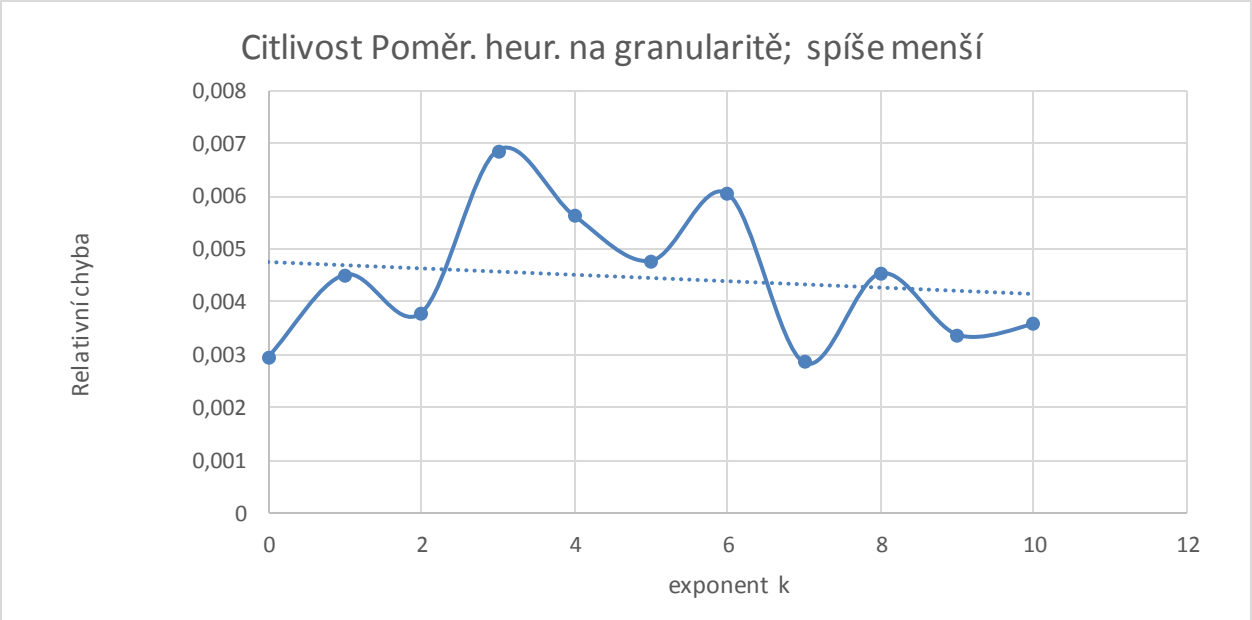
4.3.2.4.2 Greedy heuristika pro d=-1



4.3.2.4.3 Heuristka podle poměru cena/váha pro d=1



4.3.2.4.4 Heuristka podle poměru cena/váha pro d=-1



4.3.2.5 FPTAS

Všechny výše měřené parametry jsem vyzkoušel i na algoritmu FPTAS pro ε = 0,3. Nepodařilo se mi vypozorovat žádný jednoznačný trend či chování algoritmu, kromě těch vycházejících z Dynamického programování.

5 ZÁVĚR

5.1 RYCHLOST

Na základě mých měření ani Dynamické programování ani B&B nevykazuje závislost na maximální hmotnosi předmětů. S přihlédnutím k faktu, jak tyto algoritmy pracují, je toto očekávané chování.

Jak je vidět z grafů, doba běhu **Dynamického programování je závislá na maximální ceně** předmětů. Rozdíl mezi dobou běhu pro předměty o maximální ceně 100 a 100 000 je o něco málo více než **dvojnásobek**. Je třeba zmínit, že tato závislost není horší než lineární a tento jev je očekáváný, protože doba běhu dynamického programování je přímo závislá na velikosti vyplňované tabulky, jejíž jeden rozměr je dán součtem cen všech předmětů.

U algoritmu B&B se neprojevila žádná citlivost na maximální cenu, což je očekávané, protože v implementaci tohoto algoritmu není žádné urychlení, které by mohlo být pouze maximální cenou ovlivněno.

Poměr kapacity batohu k sumární váze předmětů způsobuje růst doby běhu Dynamického programování přibližně lineárně až do stavu, kdy se nám do batohu vejdou předměty všechny. Je to způsobené tím, že při nižších hodnotách tohoto parametru jsou generovány věci větší hmotnosti a je tak možné dříve zamezit vykonávání určité větve rekurze Dynamického programování (má implementace je zdola nahoru).

U algoritmu B&B průběh vypadá jako obrácená parabola. Tedy nejrychleji algoritmus běží při nízkém a velkém poměru. Při nastavení, kdy se do batohu vejde **přibližně polovina předmětů, je nejpomalejší**. Toto je způsobené tím, že při nízkých hodnotách poměru se projevuje **ořezávání vlivem přetížení** a při vysokých zase **ořezávání vlivem ceny hranice**.

Citlivost doby běhu obou algoritmů na granularitu zdá se není. Pouze v případě B&B se zdá, že je o něco rychlejší spíše pro větší předměty. To je ale ovlivněno také hodnotou poměru kapacity batohu k sumární váze, která byla pro účely tohoto měření zafixována na hodnotě 0,7. Obecně toto tvrdit nelze.

Oba algoritmy jsou úplné a mají proto kvalitu 100%. Algoritmus B&B je méně citlivý na vstupní data, je ovlivněný pouze poměrem kapacity batohu k sumární váze.

5.2 KVALITA

Ani Greedy ani Poměr cena/váha heuristika se nezdá být přímo závislá na maximální hmotnosti či ceně předmětů. Implementace algoritmů napovídá, že tyto hodnoty opravdu kvalitu nijak ovlivnit nemohou, pouze posouvají oblast řešení, ale poměry mezi jednotlivými předměty, o které nám jde, nejsou ovlivněny.

Jak heuristika Poměr cena/váha, tak Greedy je však ovlivněna poměrem kapacity batohu k sumární váze. Platí zde, že čím větší je poměr **(čím větší poměr ze zadaných předmětů se nám do batohu vejde), tím lepší je kvalita**. Je tomu tak proto, že se zvětšujícím se tímto poměrem nám klesá počet předmětů, které se průměrně do batohu nevejdou a tím se zmenšuje i oblast, na které lze udělat chybu. Pro poměr 1,0 a více už žádnou chybu udělat nelze.

Násobnost tohoto vlivu je přibližně stejná pro obě heuristiky, nehledě na to, že poměrová heuristika je kvalitnější, než hladová.

Jak heuristika Poměr cena/váha, tak heuristika Greedy (u hladové však především, u poměrové to pouze napovídá trendové proložení) je závislá na granularitě tak, že pokud máme **instance se spíše velkými předměty, klesá nám kvalita** (roste relativní chyba). Komplementárně **při použití spíše menších předmětů nám kvalita roste**. Je to očekávané chování, protože menší předměty nám vždy dávají možnost lépe využít prostor batohu a zužuje se možnost chyby.