3. Experimentální hodnocení kvality algoritmů

Zadání úlohy:

- Prozkoumejte citlivost metod řešení problému batohu na parametry instancí generovaných generátorem náhodných instancí.
- Na základě zjištění navrhněte a proveďte experimentální vyhodnocení kvality řešení a výpočetní náročnosti
- Zkoumejte zejména následující metody:
  - ↘ hrubá síla (pokud z implementace není evidentní úplná necitlivost na vlastnosti instancí)
  - ↘ metoda větví a hranic, případně ve více variantách
  - ↘ dynamické programování. FPTAS algoritmus není nutné testovat, pouze pokud by bylo podezření na jiné chování, než DP
  - ↘ heuristika - poměr cena/váha
- Pozorujte zejména závislosti výpočetního času (případně počtu testovaných stavů) a rel. chyby na:
  - ↘ maximální váze věcí
  - ↘ maximální ceně věcí
  - ↘ poměru kapacity batohu k sumární váze
  - ↘ granularitě

Hrubá síla:

Algoritmus hrubé síly jsem z experimentu vynechal (jak povoluje zadání), protože je zcela nezávislý na instanci problému. Vždy vyzkouší všechny možné kombinace a na konci vybere tu nejlepší. Výpočetní čas je proto exponenciálně závislý pouze na počtu předmětů na vstupu.

Heuristická metoda:

Heuristická metoda musí být ze své podstaty datově citlivá. Odhaduji, že jejích průběh ovlivní zejména poměr celková váha a kapacity batohu. Případně granularita.

Metoda větví a hranic:

Dá se očekávat, že metoda bude nějakým způsobem závislá na poměru sumární váhy a kapacity batohu. Očekávám, že s rostoucím poměrem poroste i nutný výpočetní čas, kvůli nefunkční horní mezi.

Dynamické programování:

Odhadujeme, že dynamické programování bude závislé na maximální (průměrné) ceně předmětu, což vyplývá z principu algoritmu.

Experiment:

Pro každý parametr jsem vygeneroval pomocí přiloženého generátoru vstupní data a následně jsem nad nimi pustil všechny tři algoritmy (metodu větví a hranic, dynamické programování a heuristiku).

Pro zajištění kvalitního měření času jsem pro menší instance algoritmy pustil v cyklu a jejich čas je proto tvořen průměrem těchto časů. Proto i hodnoty, které jsou obyčejně na hranici měřitelnosti, dávají smysl.

Relativní chyba vůči optimálnímu řešení pro heuristické řešení je počítána podle doporučeného vzorce z eduxu.

$$\varepsilon_{rel} = \frac{C(OPT) - C(APX)}{C(OPT)}$$

Použité přepínače kompilátoru GCC:

-Ofast -flto -Wno-sign-compare -std=c++11 -fopenmp -fstrict-aliasing -mfpmath=sse -mavx -mpc32 -funsafe-math-optimizations -ffast-math

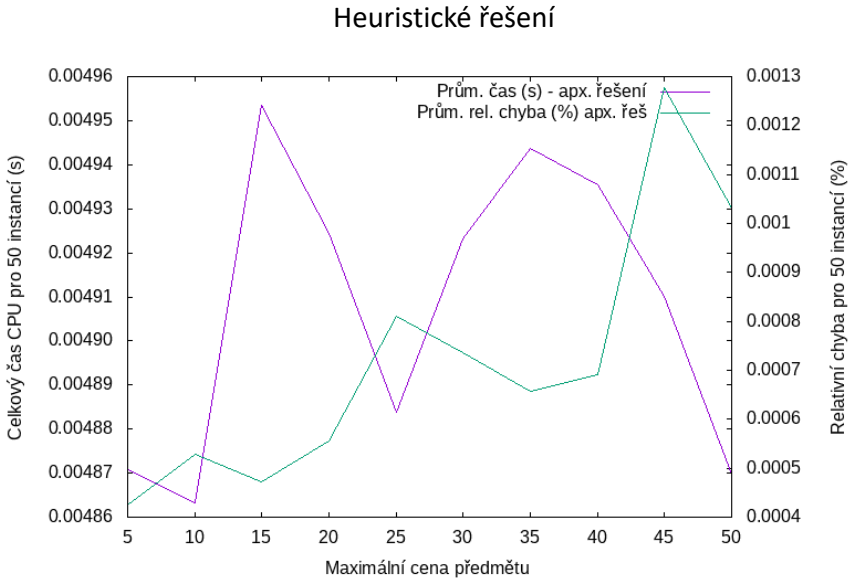
Testovací soustava:

Intel(R) Core(TM) i5-3570K CPU @ 3.40GHz + 4 cores + Hyper Threading

Úlohy byly spouštěny v režimu jednoho vlákna a na PC neběžela žádná náročná úloha. K měření času jsem použil standardní knihovnu, se kterou mám dobré zkušenosti.

Závislost na maximální ceně věcí:

Pro parametr maximální ceny předmětů jsem zvolil rozsah <5, 50> s krokem 5. Zbývající parametry jsem zafixoval. Počet předmětů 50, poměr kapacity k sumární váze: 0.8, maximální váha: 100, rovnováha malých a velkých věcí.



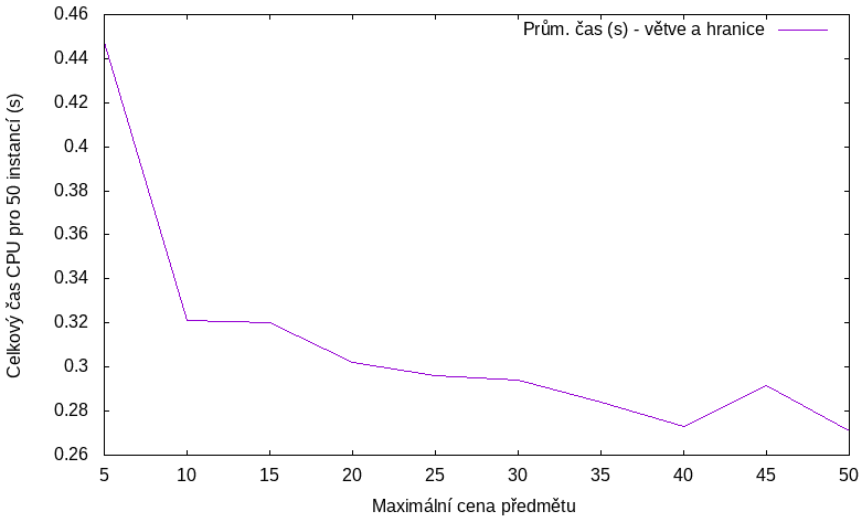
Graf [1] závislosti celkového času CPU a prům. rel. chyby v závislosti na max. ceně předmětu.

Z grafu [1] je čitelné, že pro heuristické řešení je závislost času na maximální ceně předmětu minimální a případné rozdíly jsou spíše způsobeny spíše rozptylem měření. I když jsem měření prováděl pro velký počet instancí i opakování. To odpovídá lineární složitosti algoritmu, která závisí pouze na počtu předmětů.

Relativní chyba roste se zvyšující se cenou předmětu, to může souviset s tím, že roste rozptyl (resp. průměr) ceny. Proto jsou pravděpodobnější instance, které obsahují těžký a drahý objekt, který by v optimálním řešení neměl být, jednodušeji roste i rozptyl heuristického řešení.

Přesto jde pouze o statistická data a případná závislost nemusí být vždy vidět

Metoda větví a hranic

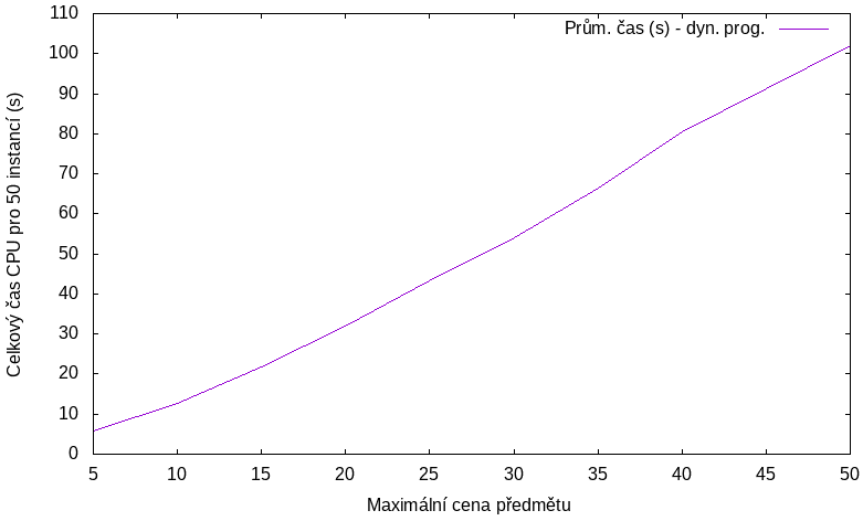


Graf [2] závislosti celkového času CPU v závislosti na max. ceně předmětu.

Graf [2] zobrazuje závislost času CPU na maximální zvolené ceně předmětu pro metodu větví a hranic. Je čitelné, že s rostoucí maximální cenou klesá časová složitost procesu poměrně podstatnou měrou. Téměř dvojnásobné zrychlení stojí za rozbor.

Toto chování může být způsobeno tím, že cena věcí je s rostoucí max. cenou více diferencována. To usnadňuje práci heuristického odhadu nejlepšího řešení, lze pak snadněji označit slibné a zbytečné větve stromu a tím urychlit výpočet.

Dynamické programování (rozklad podle ceny)

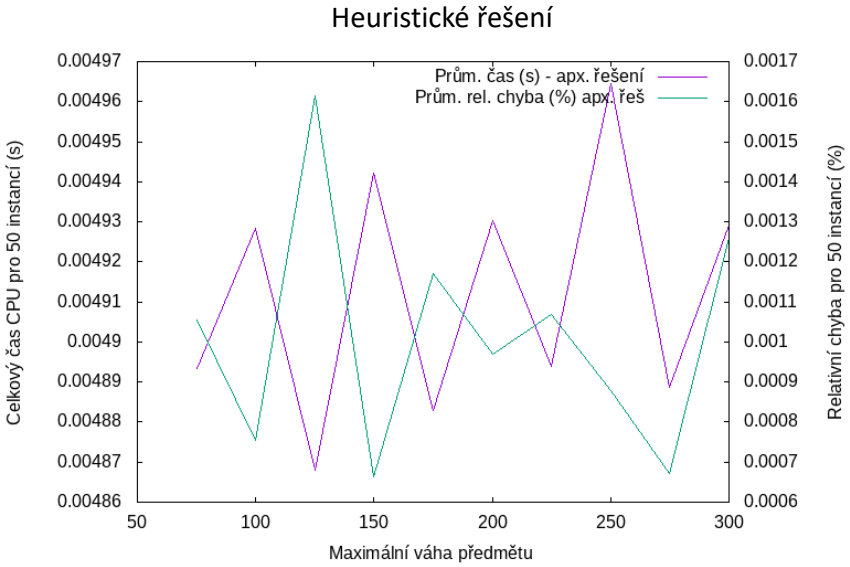


Graf [3] závislosti celkového času CPU v závislosti na max. ceně předmětu.

Graf [3] zobrazuje závislost času CPU na maximální zvolené ceně předmětu pro dynamický rozklad podle ceny. Jak jsme předpokládali, spotřebovaný čas roste s maximální cenou předmětu. Platí totiž, že dynamické programování alokuje a pracuje s $O(m * n)$ {m – celková cena, n – počet položek} jednotkami paměti. Graf odpovídá odhadům, že složitost poroste polynomiálně s rostoucí (tím pádem průměrnou) cenou předmětu.

Závislost na maximální váze předmětu

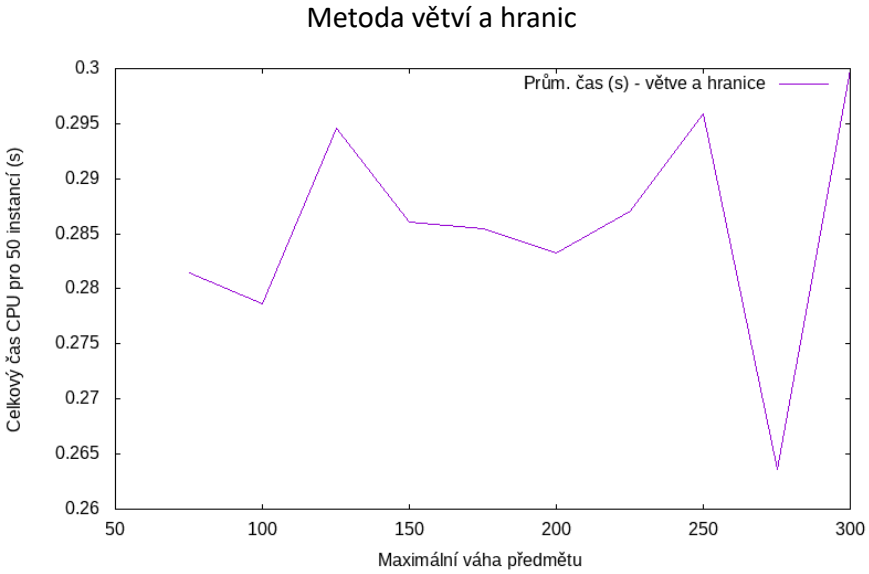
Pro parametr maximální ceny předmětů jsem zvolil rozsah <75, 300> s krokem 25. Zbývající parametry jsem zafixoval. Počet předmětů 50, poměr kapacity k sumární váze: 0.8, maximální cena: 100, rovnováha malých a velkých věcí.



Graf [4] závislosti celkového času CPU a relativní chyby v závislosti na max. váze předmětu.

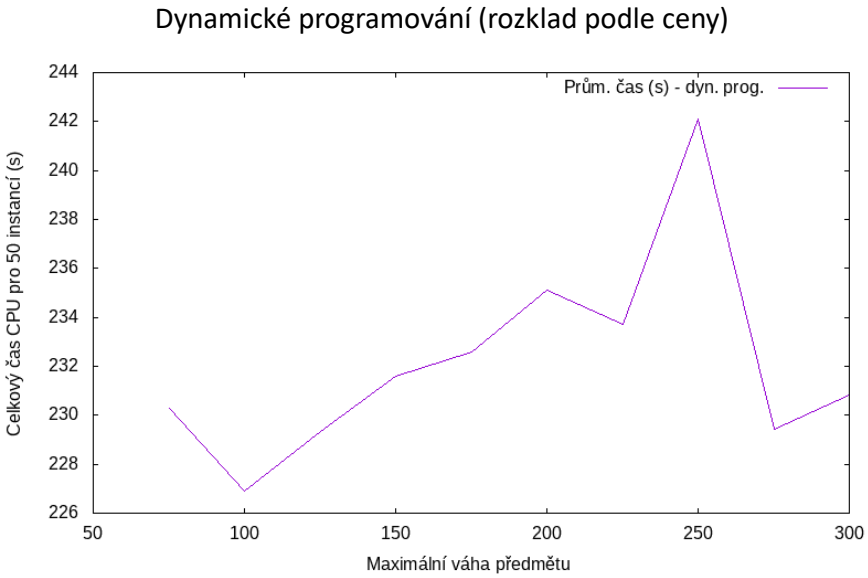
Z grafu [4] je čitelné, že pro heuristické řešení je závislost času i relativní chyby na maximální váze předmětu minimální a případné rozdíly jsou spíše způsobeny spíše rozptylem měření. I když jsem měření prováděl pro velký počet instancí i opakování. Z naměřených dat přesto nevyplývá citlivost heuristického řešení na maximální váze věci.

Přesto jde pouze o statistická data a případná závislost nemusí být vždy vidět



Graf [5] závislosti celkového času CPU v závislosti na max. váze předmětu.

Graf [5] zobrazuje závislost celkového času CPU v závislosti na maximální váze předmětu pro metodu větví a hranic. Celkový čas, i přes to, že počet opakování byl velmi vysoký, je poměrně ustálený a výkyvy jsou relativně malé. Metoda větví a hran se zdá být poměrně necitlivou na změnu maximální váhy předmětu.

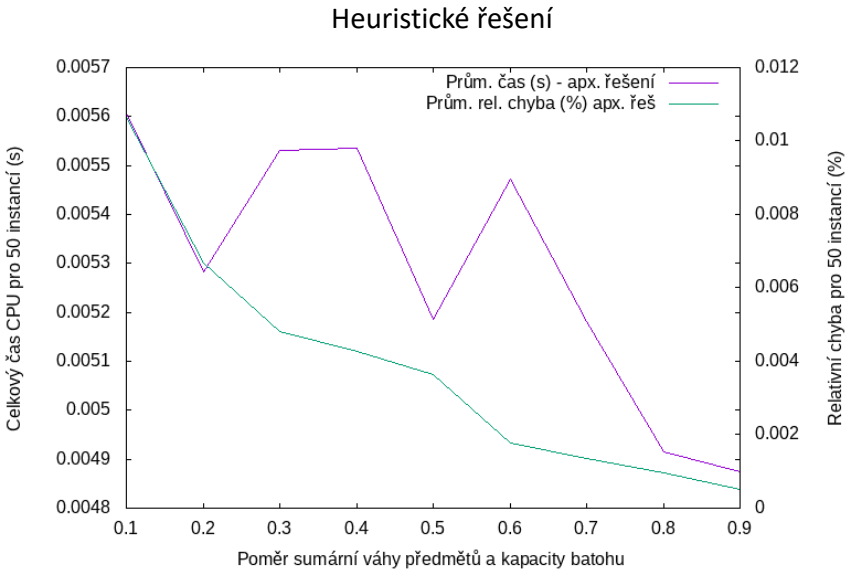


Graf [6] závislosti celkového času CPU v závislosti na max. ceně předmětu.

Graf [6] zobrazuje závislost času CPU na maximální zvolené váze předmětu pro dynamický rozklad podle ceny. Zdá se, že doba výpočtu mírně roste se zvyšující se maximální vahou předmětu. V tomto případě se může jedna o nahodilost a rozptyl generovaných instancí. Čas se totiž nezvyšuje nijak kriticky, i když mírná závislost se může objevit.

Závislost na poměru sumární váhy a kapacity

Pro parametr poměru sumární váhy a kapacity jsem zvolil rozsah <0.1, 0.9> s krokem 0.1. Zbývající parametry jsem zafixoval. Počet předmětů 50, poměr kapacity k sumární váze: 0.8, maximální cena: 100, maximální váha: 200, rovnováha malých a velkých věcí.



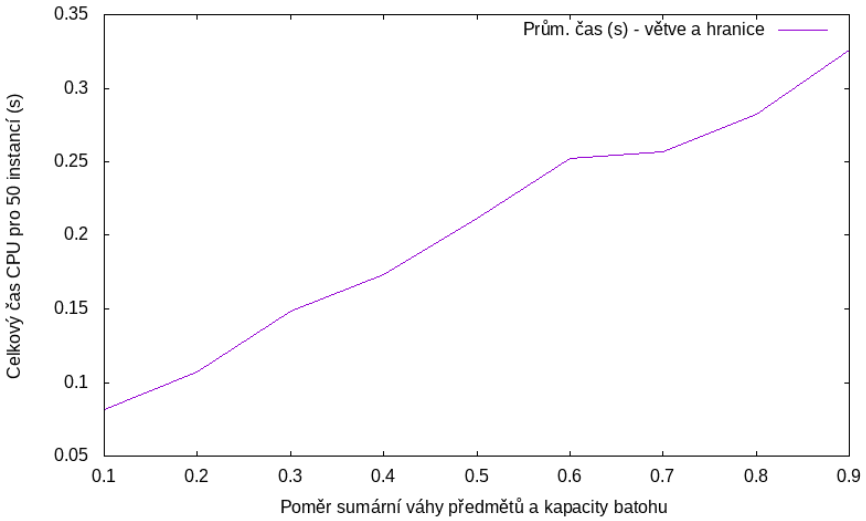
Graf [7] závislosti celkového času CPU a relativní chyby v závislosti na poměru sum. váhy a kapacity.

Graf [7] zobrazuje závislost času CPU a relativní chyby na poměru sum. Váhy a kapacity batohu. Podle předpokladu by se se zvyšujícím se poměrem měla klesat chyba, protože klesá pravděpodobnost, že některý předmět není součástí optimálního řešení (pro poměr 1.0 se optimálního řešení účastní všechny předměty).

Graf podporuje nás předpoklad závislosti, se zvyšujícím se poměrem rapidně klesá relativní chyba. Z funkcionality algoritmy je jasné, že pro poměr 1.0 bude chyba vždy nulová.

Celkový spotřebovaný čas se mění pouze pozvolna a po prozkoumání algoritmy vyplývá, že závislost by měla být minimální.

Metoda větví a hranic



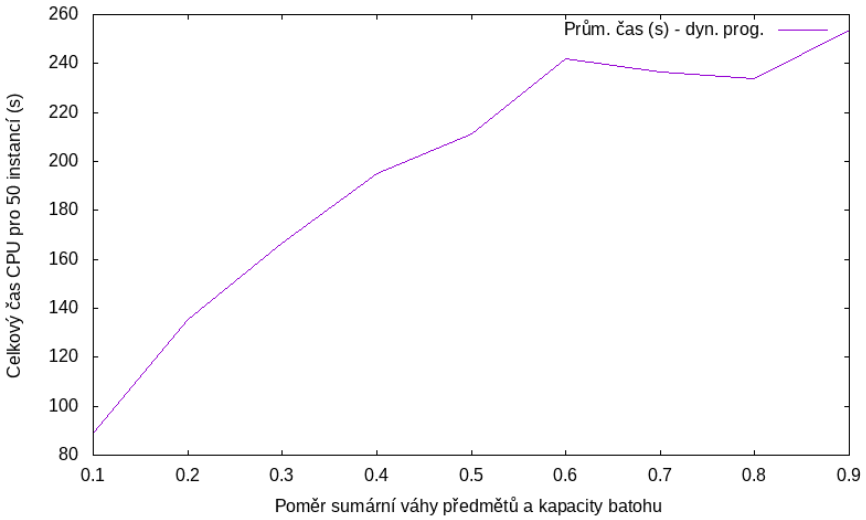
Graf [8] závislosti celkového času CPU v závislosti na poměru sum. váhy a kapacity.

Graf [8] zobrazuje závislost celkového času CPU v závislosti na poměru sum. váhy a kapacity pro metodu větví a hranic. Můžeme snadno odvodit, že s rostoucím poměrem zahrnutých předmětů v optimálním řešení stoupá i čas potřebný pro výpočet metodou větví a hranic.

Heuristika použitá metodou větví a hranic přestává být výhodná. To může být zapříčiněno tím, že přestává fungovat horní mez, protože s každým dalším předmětem je pravděpodobnost, že zvýšíme nejlepší nalezené řešení, poměrně vysoká. Protože víme, že zahnutím dalšího předmětu zlepšíme řešení. Užitečnosti heuristiky proto závisí na poměru sumární váhy a celková kapacity batohu.

Tento výsledek potvrzuje náš odhad.

Dynamické programování (rozklad podle ceny)



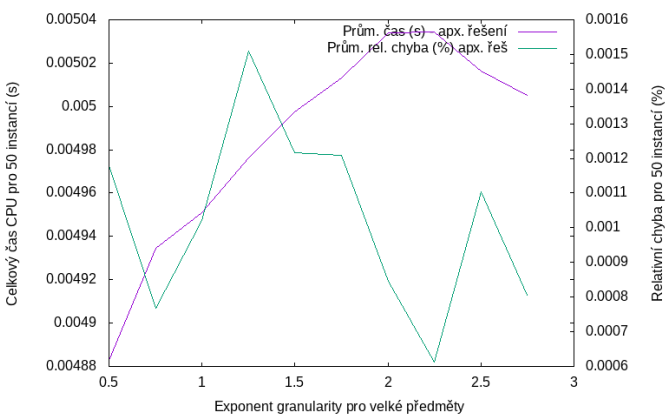
Graf [9] závislosti celkového času CPU v závislosti na poměru sum. váhy a kapacity.

Z grafu [9] je čitelné, že pro dynamické programování roste potřebný čas pro zvyšující se poměr sumární váhy a kapacity batohu. To může být způsobeno tím, že horní mez (kapacity batohu) problému přestává účinně ořezávat možné průchody a zvětšuje se proto hloubka rekurzivního průchodu.

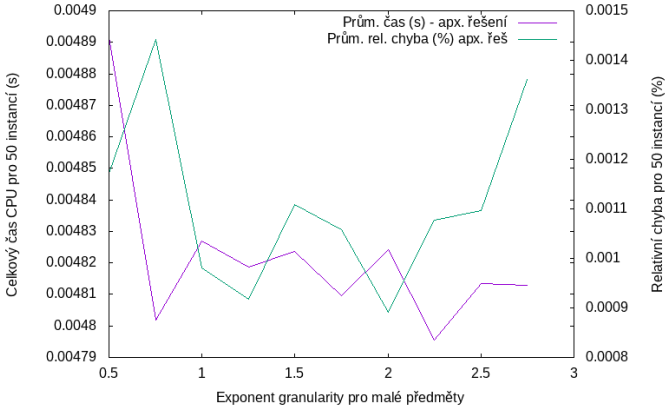
Závislost na granularitě instance

Pro parametr granularity jsem zvolil rozsah <0.5, 2.75> s krokem 0.25, jednou pro převahu malých předmětů a jednou pro převahu velkých předmětů. Zbývající parametry jsem zafixoval. Počet předmětů 50, poměr kapacity k sumární váze: 0.8, maximální cena: 100, poměr sum. váhy a kap: 0.8

Heuristické řešení



Graf [10] závislosti celkového času CPU a relativní chyby v závislosti na gran. pro převahu velk. předm.



Graf [11] závislosti celkového času CPU a relativní chyby v závislosti na gran. pro převahu mal. předm.

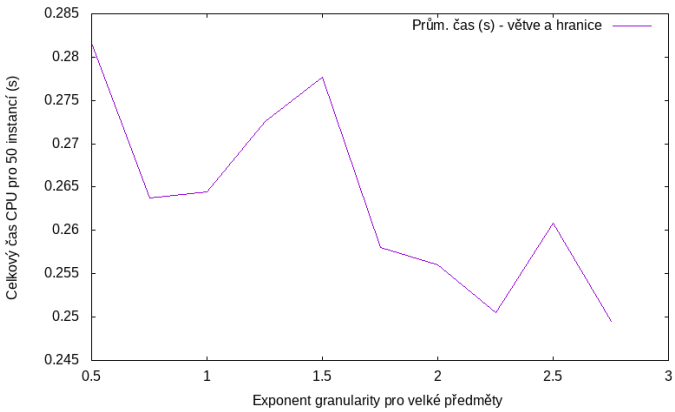
Grafy [10] a [11] zobrazují závislosti celkového času CPU a relativní chyby na granularitě předmětů pro převahu velkých, respektive malých předmětů. V grafech je vidět, že rozptyl naměřených hodnot je poměrně vysoký, přesto je zřetelné, že závislost v datech může být. Pro velké předměty se se zvyšující se granularitou zvyšuje i doba výpočtu, pro převahu malých předmětů vypadá výsledek opačně. Výsledný čas, je tak citlivý na indexu granularity.

Pro velké předměty je obecně pravděpodobnost, že se účastní optimálního řešení stejná jako pro malé. Pokud ale změníme granularitě, změníme tím i průměrné složení optimálního řešení.

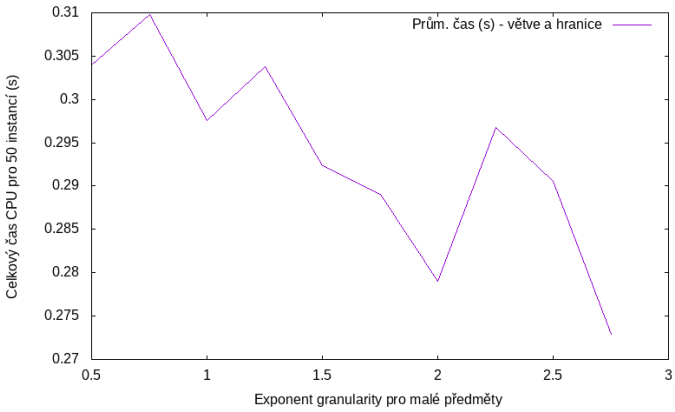
Vysvětlení časové závislosti pro velké předměty (pro malé předmětu funguje popis obráceně): Algoritmus si předměty seřadí podle poměru výhodnosti. Pokud zároveň roste pravděpodobnost velkého předmětu, je očekávané, že na prvních místech bude nějaký velký předmět téměř pokrývající kapacitu batohu. Proto musí algoritmus projít větší část pole, aby našel menší předmět, kterým zaplní zbývající místo. To je ale s rostoucí granularitou (pravděpodobností) velkého předmětu obtížnější (musíme projít celé pole). Tímto způsobem si vysvětluji zmíněnou časovou závislost.

Z dat se zdá, že relativní chyba nezávisí na poměru převahy velkých, resp. malých předmětů.

Metoda větví a hranic



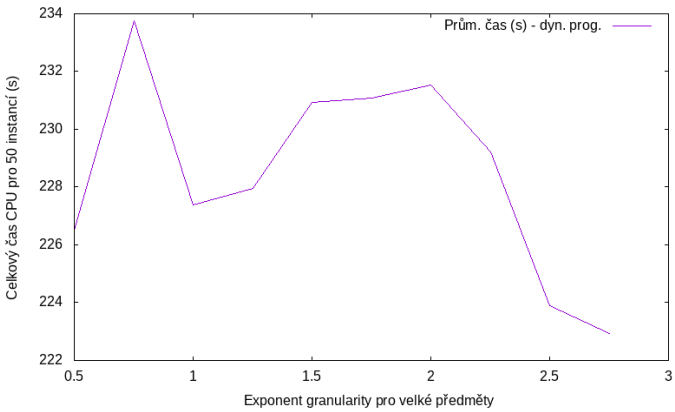
Graf [12] závislosti celkového času CPU v závislosti na gran. pro převahu velk. předm.



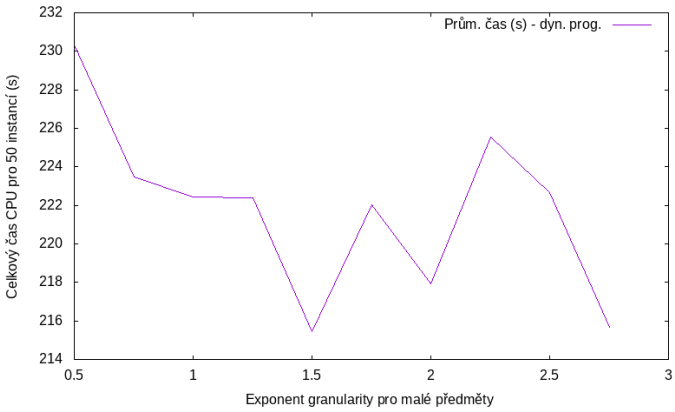
Graf [13] závislosti celkového času CPU v závislosti na gran. pro převahu malých předm.

Grafy [12] a [13] zobrazují závislosti celkového času na granularitě předmětů pro převahu velkých, respektive malých předmětů. Lze vypozorovat, že s rostoucí granularitou klesá výpočetní čas. Může to být způsobeno tím, že s rostoucím poměrem granularity je množina předmětů vychýlená ve smyslu poměr cena / váha proto je pro heuristiku snadnější odhadnout, jestli je předmět součástí zlepšujícího řešení.

Dynamické programování (rozklad podle ceny)



Graf [14] závislosti celkového času CPU v závislosti na gran. pro převahu velk. předm.



Graf [15] závislosti celkového času CPU v závislosti na gran. pro převahu malých předm.

Grafy [14] a [15] zobrazují závislosti celkového času na granularitě předmětů pro převahu velkých, respektive malých předmětů. Lze vypozorovat, že s rostoucí granularitou klesá výpočetní čas. Může to být způsobeno tím, že s rostoucím poměrem granularity je množina předmětů vychýlená ve smyslu poměr cena / váha proto je pro rekurzivní funkci snadnější odříznou řešení, která porušují maximální nebo minimální

Závěr:

Během experimentu jsme vyhodnotili několik algoritmů řešících problém batohu. Programy jsme spustili nad různými instancemi dat a pozorovali jsme změny v časové složitosti, případně relativních chybě v závislosti na změně parametrů instancí.

Podle předpokladů se zejména algoritmus dynamického programování při rozkladu podle ceny projevil silně datově citlivý na zvolené maximální ceně.

Metoda větví a hranic se ukázala být datově závislou na maximální ceně předmětů.

Podle odhadu se heuristické řešení projevilo jako citlivé na poměr sumy vah a celkové kapacita batohu zejména z pohledu relativní chyby. Stejně tak i metoda větví a hranic a dynamické programování se zdá být citlivá na tento poměr, pro ně roste čas se zvyšujícím se poměrem.

Ve všech třech algoritmech se projevuje i koeficient granularity, ale pouze mírně.

Je nutné říct, že naměřené hodnoty slouží pouze jako podpůrné argumenty pro naše odhady a samy o sobě nic nedokazují. Jedná se tak o statistiku, kdy si můžeme být našimi tvrzeními jisti s nějakou přesností, která závisí na celém způsobu provádění experimentu.