Jan Klicpera

NI-KOP – Domácí Úkol 3

1. Specifikace úlohy

Detailní specifikace je dostupná na https://moodle-vyuka.cvut.cz/mod/assign/view.php?id=89700.

Úkolem je experimentálně vyhodnotit kvalitu algoritmů implementovaných v předchozím úkolu. Sledovány jsou dva parametry algoritmů:

- Výpočetní náročnost
- Kvalita řešení, pokud se jedná o heuristiku

K dispozici je generátor instancí batohu, který je parametrizován a umožňuje poskytnout instance batohu s požadovanými vlastnostmi.

2. Implementace algoritmů

Pro implementaci byl zvolen jazyk C#, výstupem je konzolová aplikace. Aplikace je plně konfigurovatelná za pomocí přepínačů, chybné vstupy jsou ošetřeny.

Čas běhu byl měřen pomocí systémové třídy *Stopwatch*, která umožňuje velmi přesně měřit výpočetní čas. Při měření času je každá instance spouštěna pětkrát, výsledným časem je průměr těchto pěti běhů.

Hrubá síla

Algoritmus řešící problém pomocí hrubé síly vždy projde všechny možné konfigurace, jeho běh tedy není nijak omezován i v případě, kdy již nelze dosáhnout platné konfigurace.

Metoda větví a hranic

Algoritmus ukončí výpočet pro danou větev konfigurací, pokud je zřejmé, že z dané částečné konfigurace již nemůže vzniknout řešení. Tedy v případech kdy:

- Váha věcí v batohu přesáhne kapacitu
- Nelze překonat dosavadní nejlepší nalezený výsledek ani při přidání všech zbývajících předmětů do batohu.

Předměty jsou zpracovány v takovém pořadí, v jakém byly uloženy ve vstupním seznamu

Metoda dynamického programování

Detailní popis implementace byl sepsán v předchozí zprávě, shrnu tedy pouze důležité vlastnosti algoritmů:

- Implementovány byly dva způsoby dekompozice – podle váhy a podle ceny.
- Pro memoizaci bylo použito 2D pole.
- Oba způsoby jsou implementovány pomocí for cyklů a začínají s prázdným batohem.
- Předměty jsou zpracovány v takovém pořadí, v jakém byly uloženy ve vstupním seznamu.
- Konfigurace, které přesáhnou kapacitu batohu nejsou dále zpracovávány.

Greedy redux heuristika

Pro řazení byla použita Linq metoda OrderBy, která je implementována dle dokumentace jako quicksort. Pro získání hodnoty kvality algoritmu se vstupní instance vyřeší i pomocí metody větví a hranic (výpočetní čas pro tuto kontrolu není započítán ve výsledcích).

3. Naměřené výsledky

Výsledky byly měřeny v OS Windows 10 64bit s procesorem AMD Ryzen 5 2600 six-core (3.4 GHz). Běh nebyl paralelizován, bylo tedy vždy využíváno pouze jedno jádro procesoru.

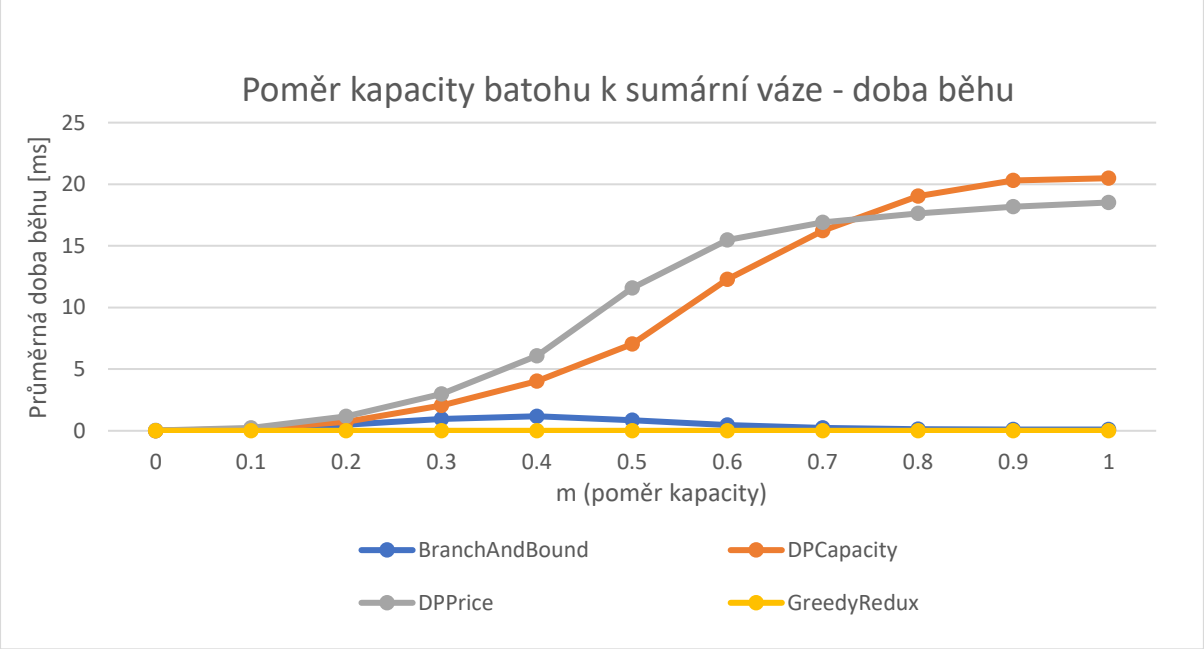
Pokud není uvedeno jinak, pak byly měřené instance generovány s následujícími parametry:

- n = 18 (počet věcí)
- N = 500 (počet instancí)
- m = 0.8 (poměr kapacity batohu k sumární váze)
- W = 5000 (maximální váha věci)
- w = bal (převaha lehkých/těžkých věcí)
- C = 5000 (maximální cena věci)
- c = uni (korelace s váhou)
- k = 1 (exponent granularity)

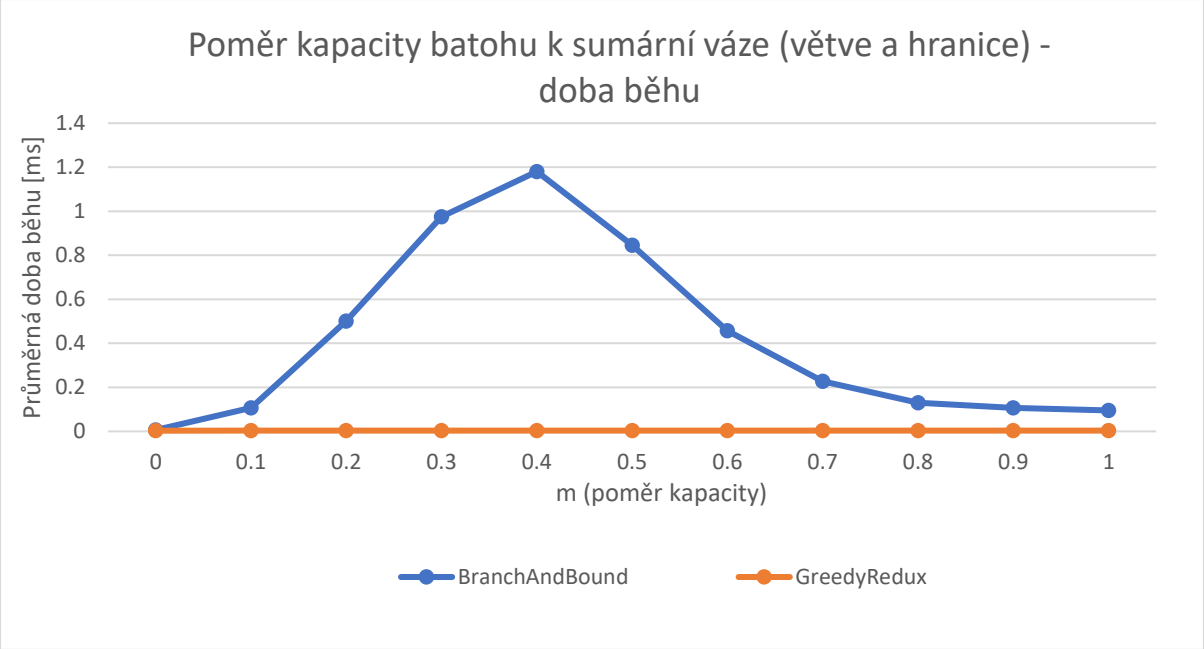
Jelikož implementovaná metoda hrubou silou projde vždy všechny možné konfigurace, lze bezpečně předpokládat, že žádný z parametrů (kromě počtu věcí) nijak neovlivní výpočetní dobu. V grafech tedy není výpočet hrubou silou uváděn. Pro představu se výpočetní čas pro jednu instanci pohyboval okolo 40ms pro n=18.

Poměr kapacity batohu k sumární váze

Před samotným měřením byla předpokládána závislost výpočetní doby obou dekompozic na tomto parametru. Obě dekompozice „zahazují" konfigurace, které překročí maximální kapacitu batohu, lze tedy předpokládat, že čím menší bude kapacitu batohu poměrově k sumární váze, tím více konfigurací se „zahodí", což zrychlí běh algoritmu.



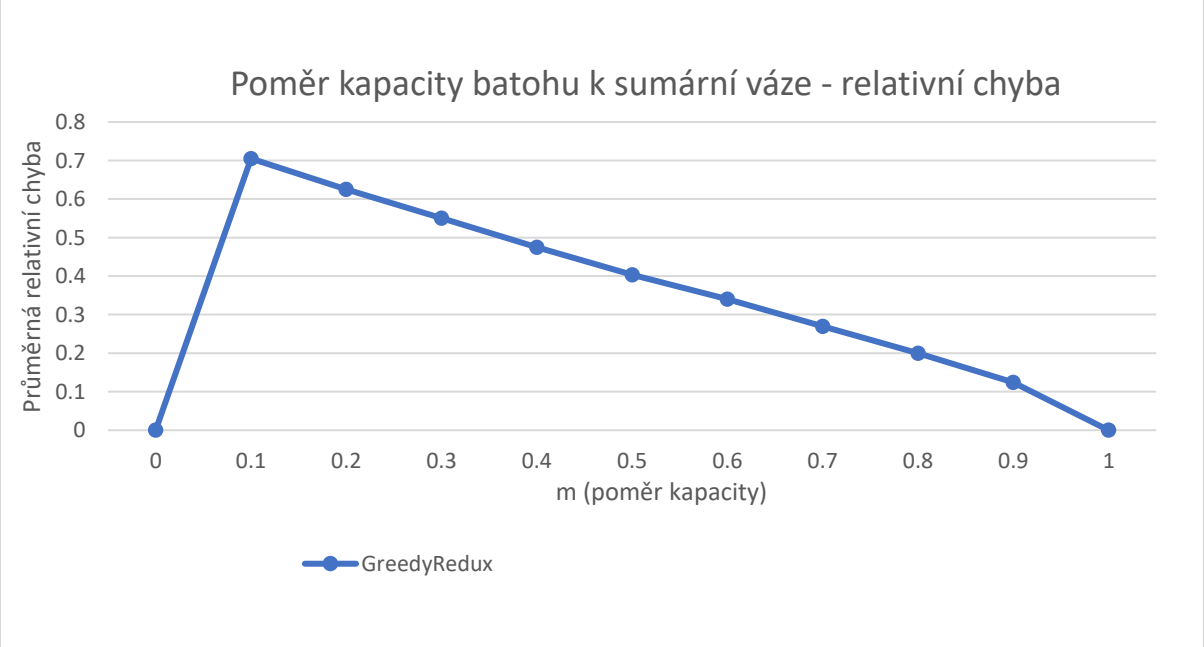
Tento předpoklad se na základě naměřených výsledků potvrdil. Z toho grafu však není patrný průběh metody větví a hranic, zde je průběh izolován od dekompozic:



Tento průběh je vcelku zajímavý a rozhodně nebyl zcela očekávaný. Z grafu je patrné, že algoritmu se nejhůře daří ořezávat, když se poměr kapacity batohu k sumární váze pohybuje okolo hodnoty 0.4. Toto je pravděpodobně způsobeno tím, že při tomto poměru se sice ořeže určité množství větví na základě překročení váhy, zkomplikuje to však nalezení „dobré“ dočasné optimální ceny pro ořezávání na základě ceny zbývajících předmětů.

Předpoklad je tedy ten, že pro hodnoty parametru m nižší, než 0.4 (zhruba) je zvýhodněno ořezávání dle váhy, pro hodnoty parametru m vyšší, než 0.4 je zvýhodněno ořezávání dle ceny.

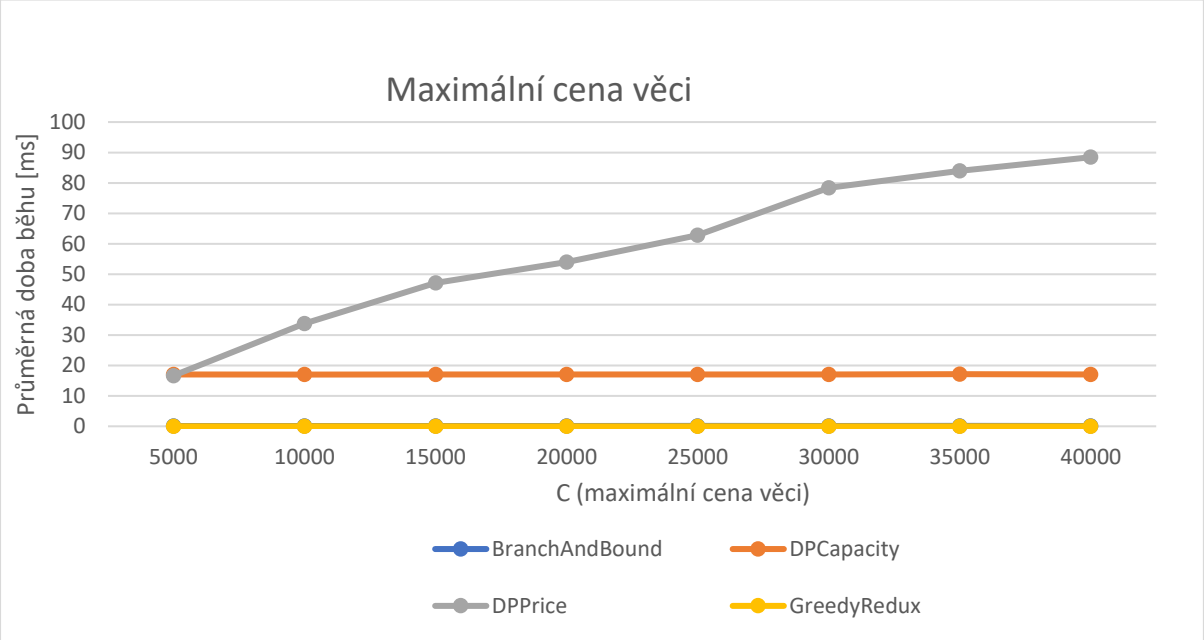
Dále byla testována průměrná relativní chyba greedy redux heuristiky. Zde jsem žádnou závislost nepředpokládal.



Na základě naměřených výsledků, je však závislost patrná. Odhadoval bych, že hlavním důvodem je ten, že čím větší je poměr, tím menší „prostor“ má algoritmus se zmýlit, protože může do batohu typicky přidat větší procento předmětů.

Maximální cena věci

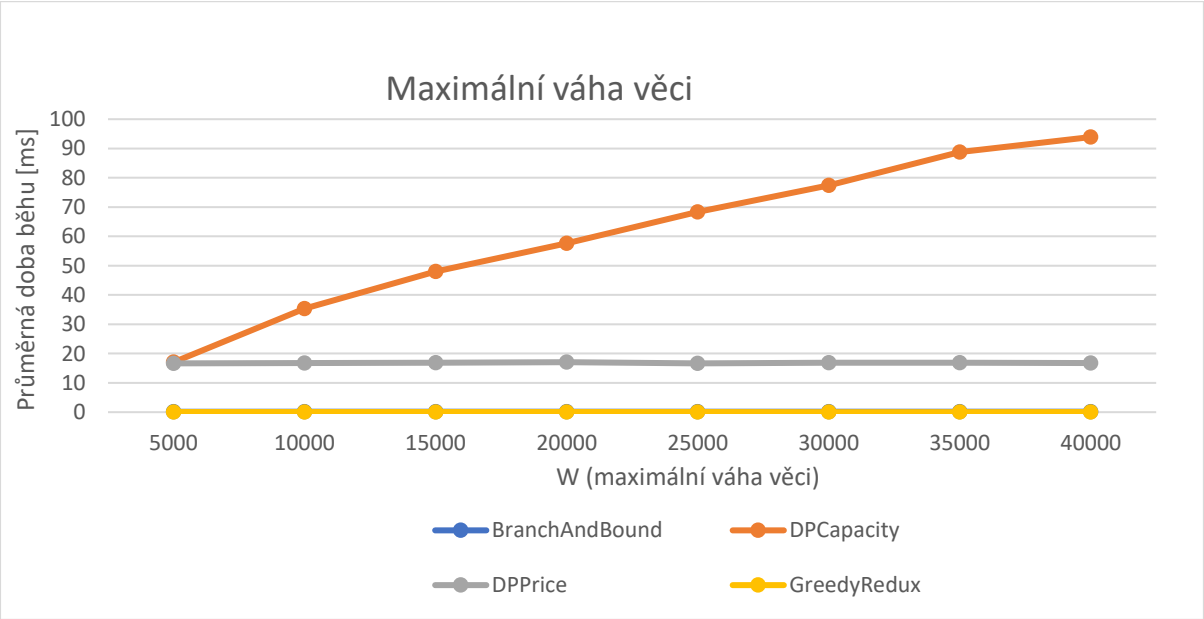
Pro tento parametr lze s jistotou říct, že ovlivní dobu běhu dekompozice dle ceny – memoizační tabulka se bude zvětšovat s rostoucí maximální cenou předmětu (za předpokladu, že hodnoty ceny předmětů budou mít rovnoměrné rozložení).



U ostatních algoritmů nebyla závislost vypozorována. Relativní chybovost greedy heuristiky také nebyla nijak ovlivněna, graf proto nebudu přikládat (epsilon byl roven cca 0.2).

Maximální váha věci

Podobně jako u předchozího parametru lze s jistotou určit závislost, tentokrát pro dekompozici dle váhy.



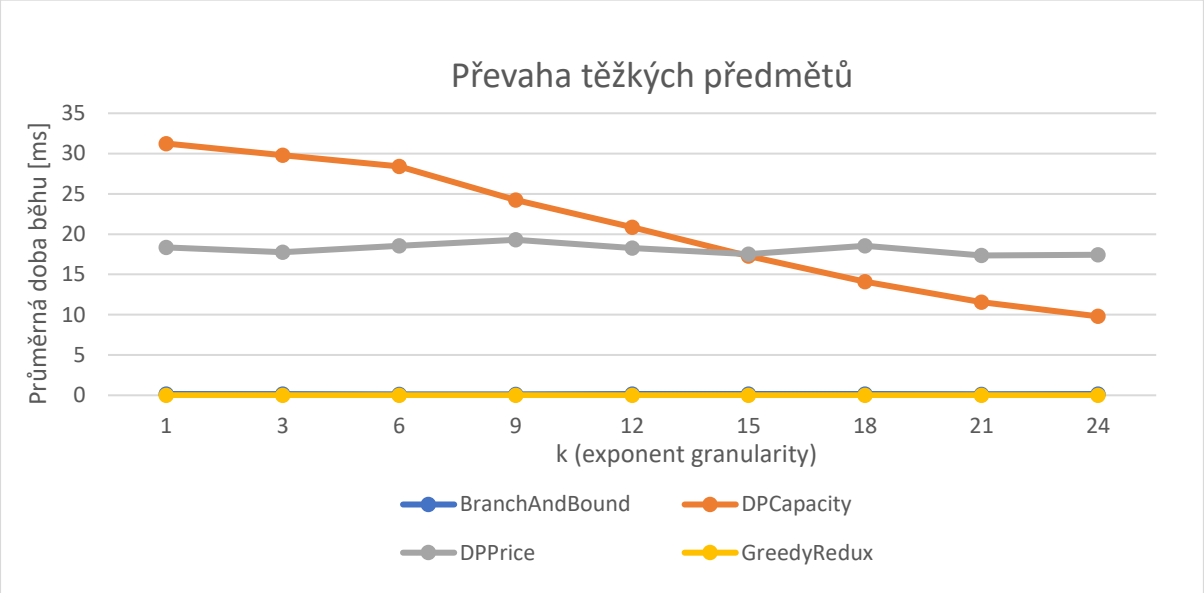
U ostatních algoritmů opět nebyla vypozorována žádná závislost. Relativní chybovost greedy heuristiky opět nebyla ovlivněna.

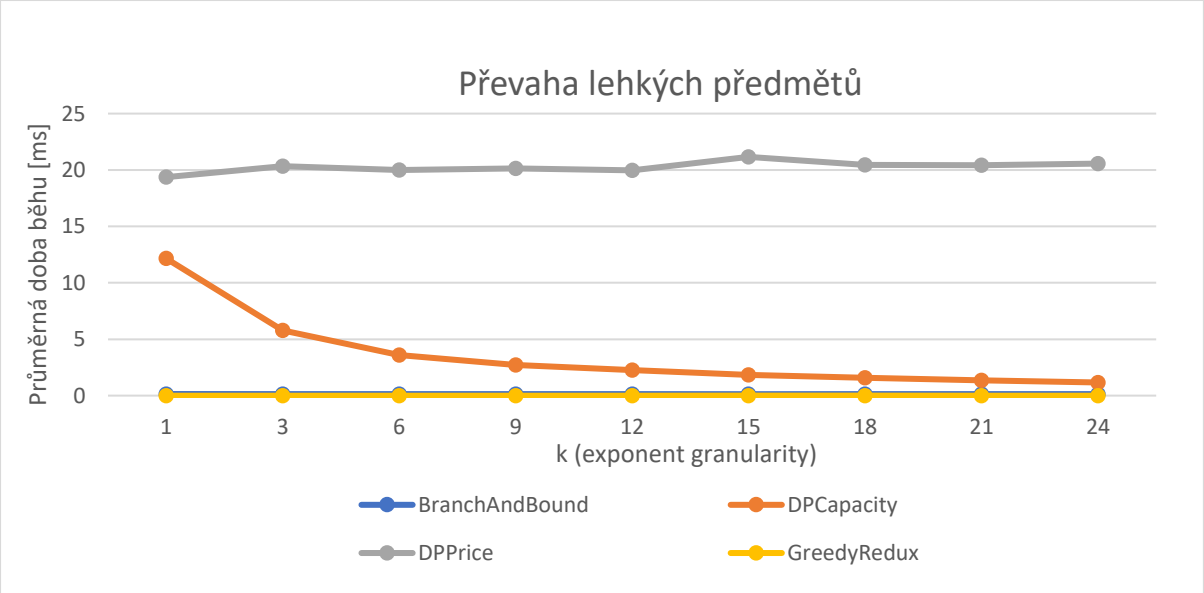
Převaha lehkých/těžkých věcí a exponent granularity

Před samotným měřením byla předpokládána závislost výpočetní doby dekompozice dle váhy na tomto parametru.

Pokud budou v batohu převahovat pouze těžší předměty, pak se sice nezmění velikost memoizační tabulky, ale bude v ní obsazeno méně políček (budou obsazována převážně pouze políčka v horní části tabulky). Tím pádem dojde k více kolizím, které „spojí" více výpočetních větví do jedné.

Pokud budou v batohu převahovat pouze lehčí předměty, tak bude ovlivněna i velikost tabulky, protože bude klesat kapacita batohu. Kapacita batohu bude klesat z toho důvodu, že je přímo závislá na celkové váze všech předmětů (parametr m, poměr kapacita/sumární váha).





Tento předpoklad se ukázal být na základě naměřených hodnot pravdivým. Zároveň se ukázalo, že algoritmus je mnohem rychlejší při převaze lehkých předmětů oproti těžkým předmětům – u lehkých

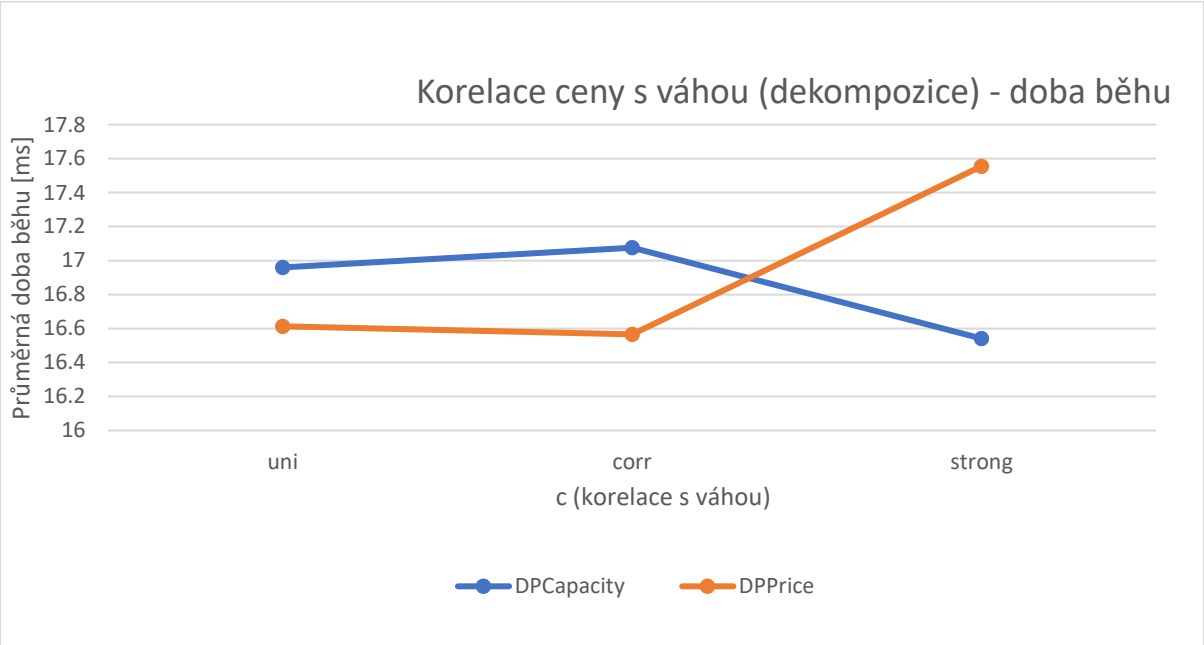
předmětů je tabulka menší, čímž opadne část režie potřebná na inicializaci pole. U dekompozice dle ceny a metody větví a hranic se žádná závislost neprojevila.

Byla také pozorována hodnota relativní chyby, závislost však také nebyla objevena, graf proto přikládat nebudu.

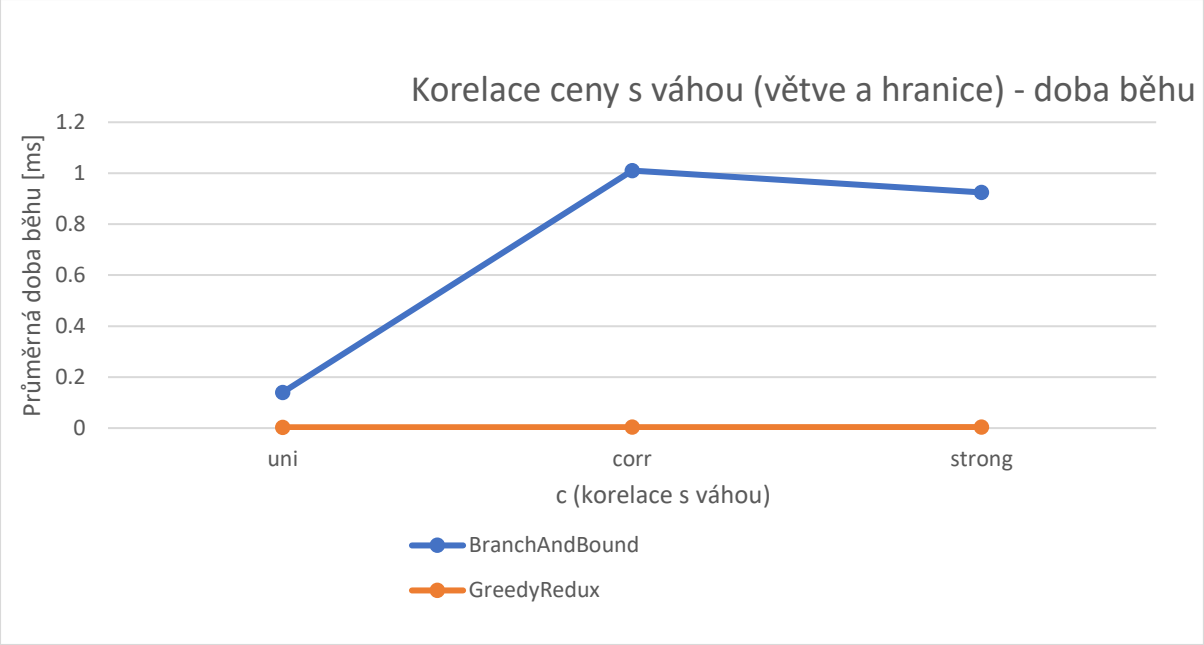
Korelace ceny s váhou

Při korelaci ceny s váhou bez změny ostatních parametrů jsem před začátkem měření neměl podezření na žádnou závislost.

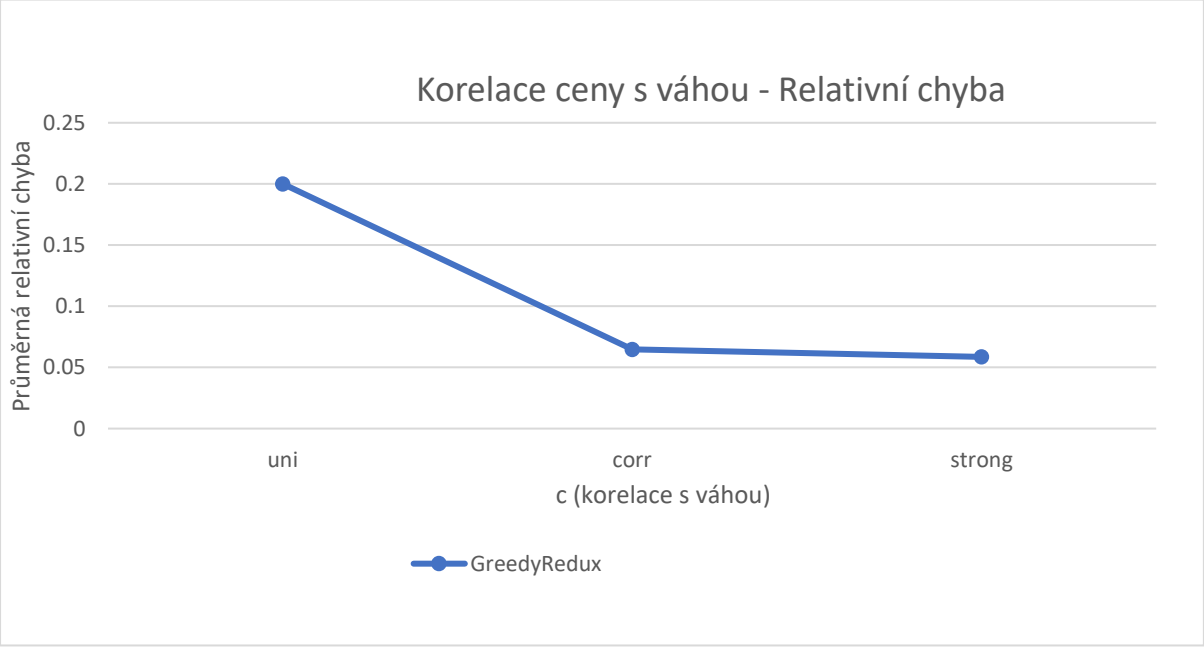
Nejprve ukážu měření při změně pouze tohoto parametru, poté také měření se změnou tohoto parametru a parametru určujícího převahu lehkých/těžkých předmětů.



U dekompozičních algoritmů je doba běhu pro hodnoty *uni* a *corr* téměř totožná. Určitá změna je pozorovatelná až pro hodnotu *strong*, tedy když cena koreluje s váhou silně. Vzrůst časové složitosti u dekompozice dle ceny bych tuto skutečnost odůvodnil tím, že rozložení váh předmětů se mírně přiklánělo k těžším předmětům, větší korelace ceny tedy způsobí i vzrůst ceny. Mírný pokles složitosti u dekompozice dle váhy však bohužel příliš dobře odůvodnit nedovedu, parametr by měl měnit pouze cenu předmětů. Není vyloučené, že vznikla určitá nepřesnost při měření, protože pokles není poměrově drastický (pouze průměrně o 0.5 ms).

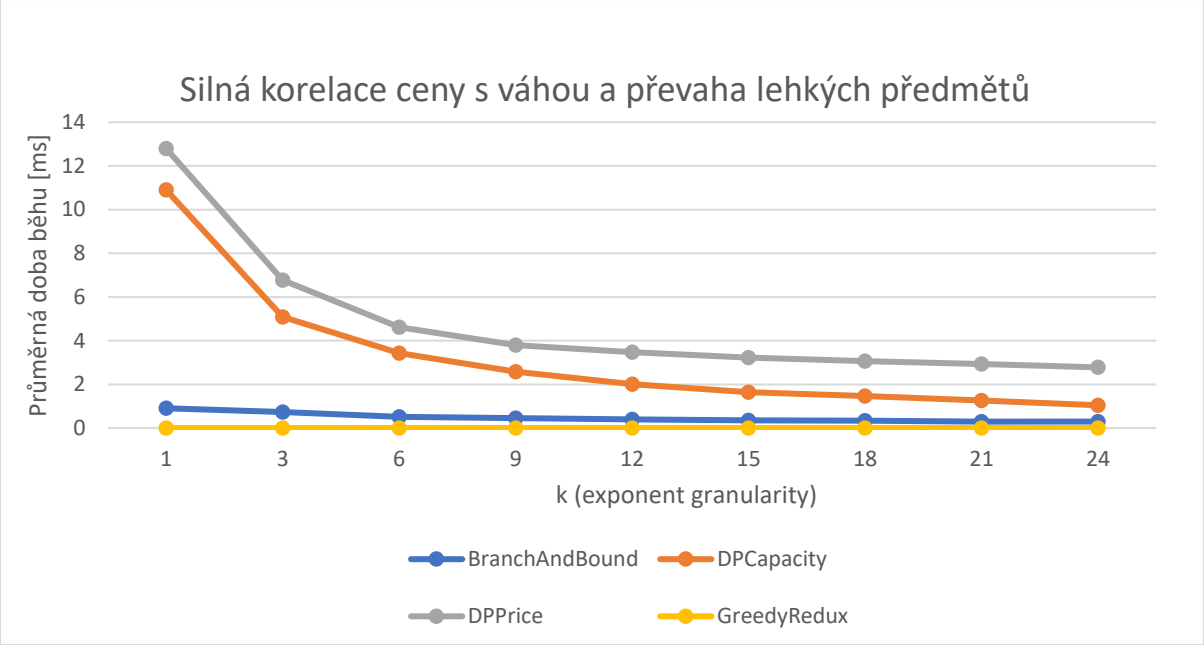


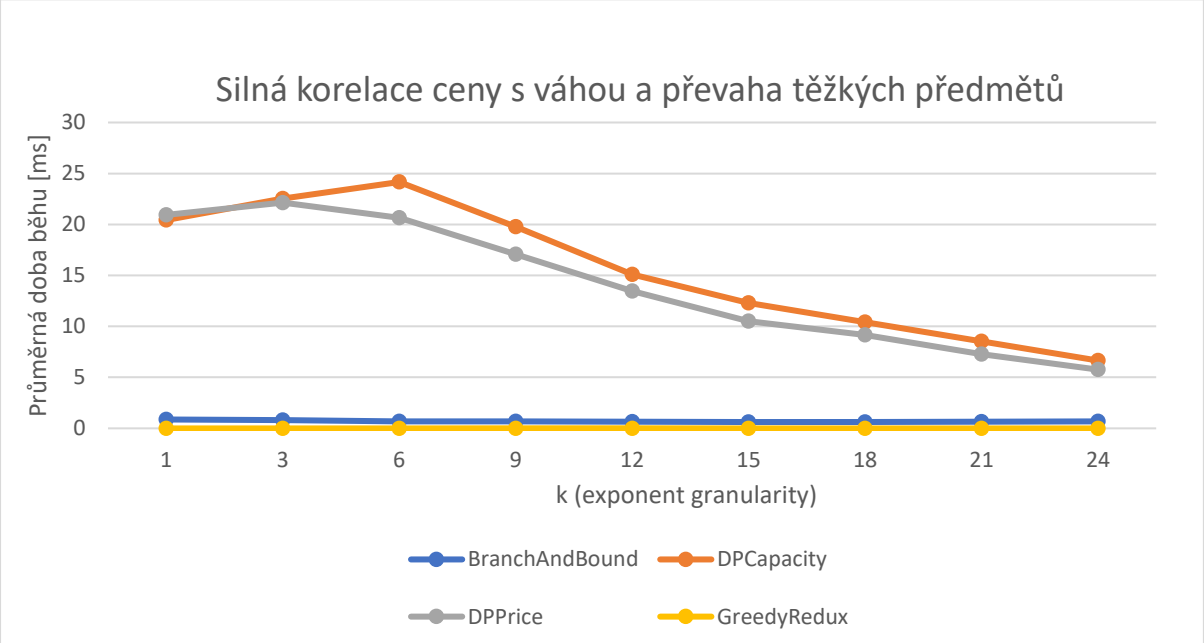
U metody větví a hranic stoupla průměrná výpočetní složitost pro hodnoty *corr* a *strong* až pětinásobně. Přesné odůvodnění také nedokážu poskytnout, nicméně podezřívám, že větší korelace ceny s váhou komplikuje ořezávání větví podle ceny.



Zajímavý výsledek poskytla hodnota průměrné chyby pro greedy heuristiku. Ukázalo se, že pro hodnoty *corr* a *strong* klesne průměrná chybovost zhruba na jednu třetinu. Toto je pravděpodobně způsobeno tím, že se při korelaci váhy a ceny výpočet poměru váha/cena stane spolehlivějším ukazatelem na vhodné kandidáty pro přidání do batohu.

Korelaci ceny s váhou lze dále kombinovat s ostatními parametry pro odhalení dalších závislostí. Silnou korelaci (hodnota *strong*) jsem zkombinoval s hodnotami exponentu granularity (k) pro převahu lehkých, resp. těžkých předmětů. Zde je podezření na závislost nejen dekompozice dle váhy, ale i dekompozice dle ceny, protože spolu s váhou předmětů bude nyní klesat, resp. stoupat i cena předmětů.





Pro metodu větví a hranic opět nebyla pozorována žádná závislost.

Robustnost algoritmů

Měření stability algoritmů je zkomplikované tím, že měření výpočetního času není naprosto přesné, v některých případech mohou při měření identických instancích značné rozdíly mezi daty (i přes průměrování 5ti běhů). Po vyzkoušení několika různých variant měření a porovnávání (minima a maxima, variance) jsem se rozhodl v zájmu zachování svého duševního zdraví tuto kategorii měřit jako počet navštívených konfigurací, kde problém s nepřesností opadne. Počet navštívení u metody větví a hranic jsem měřil jako prvním úkolu, tedy počtem rekurzivních volání výpočetní funkce. U dekompozice jsem počet navštívení měřil přičtením o jedna při navštívení nové buňky (kolize a překročení kapacity se tedy nezapočítává).

Hrubá síla byla z měření opět vynechána, protože nezáleží na pořadí předmětů ve vstupní instanci, vždy se projdou všechny konfigurace. Vynechána byla také greedy redux heuristika, u které lze (ne)robustnost snadno dokázat. U té se robustnost odvíjí od zvoleného algoritmu pro řazení. V tomto případě byl použit quicksort, který má průměrnou časovou složitost $n\,log(n)$, v nejhorším případě může mít však složitost n^2. Zvolený řadící algoritmus tedy není robustní, což znamená, že ani tato konkrétní implementace greedy redux algoritmu není robustní, co se doby běhu týče. Pořadí předmětů v instanci ale nijak neovlivní hodnotu relativní chyby, protože si algoritmus předměty sám seřadí.

Generátor byl spuštěn s parametry n = 18; N = 500; W = 500; C = 500 (ostatní parametry byly ponechány na default hodnotě). Ke každé takto vytvořené instanci bylo vytvořeno 100 permutací. Pro každou instanci s indexem i (rozmezí 1-500) byla vypočtena hodnota D_i:

$$D_i = \frac{MAX(ns_i)}{MIN(ns_i)}$$

kde ns_i označuje vektor obsahující počty kroků potřebných pro výpočet vygenerovaných permutací instance i. Hodnota tedy vyjadřuje poměr mezi maximálním a minimálním počtem kroků potřebných k výpočtu permutované instance i. Pro představu je poskytnuta tabulka, která naměřené hodnoty sumarizuje:

	Branch and Bound	DP Capacity	DP Price
$\mathrm{AVG}(D_1, D_2, \dots, D_N)$	3.412669	1.22603	1.323221
$MIN(D_1, D_2, \dots, D_N)$	2.423517	1.144192	1.210765
$MAX(D_1, D_2, \dots, D_N)$	5.771139	1.379033	1.519926

Z naměřených dat tedy vyplývá, že žádný z pozorovaných algoritmů není robustní, jelikož průměrný rozdíl mezi minimálním a maximálním počtem kroků permutovaných instancí je pro všechny pozorované algoritmy větší, než 1.

Z tabulky navíc vyplývá, že nejvíce poměrově ovlivněna pořadím zápisu předmětů v instanci je metoda větví a hranic, která může být pro některé instance průměrně až více než třikrát pomalejší než permutovaná instance obsahující stejné předměty. Oproti tomu doba běhu dekompozičních algoritmů je pořadím předmětů ovlivněna znatelně méně.

U metody větví a hranic je důvod ten, že je optimalizace prováděna na základě momentálního vybíraného předmětu a předchozích vložených předmětů. Pro dobu běhu je výhodnější, když se tyto optimalizace provedou co nejdříve, protože se tím vyloučí větší podstrom konfigurací, než kdyby se tyto optimalizace provedly déle.

U dekompozic je důvod podobný. Vznik kolizí a ukončení výpočtů z důvodu přesažení kapacity batohu je závislý na pořadí předmětů. Při kolizi dochází k sloučení dvou a více výpočetních větví do jedné. Dobu běhu tedy ovlivní moment, ve který dochází ke kolizím.

4. Závěr

Měření potvrdilo některé očekávané závislosti, ale zároveň pomohlo odhalit závislosti, které nebyly na první pohled zřejmé.

Hrubá síla

Tento algoritmus z důvodu jeho implementace nebyl uváděn v měření, protože je jeho doba běhu závislá pouze na velikosti instance.

Metoda větví a hranic

U tohoto algoritmu byla odhalena závislost doby běhu na parametrech:

- m (poměr kapacity a sumární váhy)
- c (korelace ceny s váhou).

Zároveň bylo také ukázáno, že tento algoritmus není robustní.

Dekompozice podle váhy

U této dekompozice byla odhalena závislost doby běhu na parametrech:

- m (poměr kapacity a sumární váhy)
- W (maximální váha předmětu)
- w (převaha lehkých/předmětů) spolu s k (exponent granularity)
- c (korelace ceny s váhou)

Zároveň bylo také ukázáno, že tento algoritmus není robustní, pořadí předmětů však výpočet ovlivňuje méně, než u metody větví a hranic.

Dekompozice podle ceny

U této dekompozice byla odhalena závislost doby běhu na parametrech:

- m (poměr kapacity a sumární váhy)
- C (maximální cena předmětu)
- w (převaha lehkých/předmětů) spolu s k (exponent granularity), pokud je zároveň korelace ceny s váhou nastavena na *strong*
- c (korelace ceny s váhou)

Zároveň bylo také ukázáno, že tento algoritmus není robustní, pořadí předmětů však výpočet ovlivňuje méně, než u metody větví a hranic.

Greedy redux

U této heuristiky byl sice v grafech uváděn čas výpočtu, relevantní byla spíše hlavně relativní chybovost heuristiky. Čas výpočtu se odvíjí především od zvoleného řadícího algoritmu. Byla odhalena závislost relativní chybovosti na parametrech:

- m (poměr kapacity a sumární váhy)
- c (korelace ceny s váhou)

Zároveň bylo uvedeno, že tato konkrétní implementace není robustní, protože používá quicksort. Pořadí předmětů v instanci sice tedy nepatrně ovlivní dobu běhu, ale relativní chybovost neovlivní.