Experimentální hodnocení kvality algoritmů

20. listopadu 2020, Radek Šmíd

https://github.com/Smidra/KOP-ukol-3

1 Specifikace úlohy

Cílem bylo experimentálně zhodnotit algoritmy naprogramované v předchozích cvičení. Najít a ověřit závislosti kvality a náročnosti řešení na různých druzích náhodně generovaných instancí. Instance jsou generovány pomocí dodaného náhodného generátoru. Různým nastavením parametrů generátoru docílíme různých druhů testovaných instancí.

Dále bylo cílem zhodnotit robustnost čtyřech použitých metod, tedy

1. Hrubá síla
2. Metoda větví a hranic
3. Metoda dynamického programování
4. Redux heuristika

2 Rozbor řešení

Při řešení jsem jsem měnil vždy pouze jeden parametr generátoru. Složitost jsem měřil v počtu navštívených instatncí. Pro hrubou sílu a metodu větví a hranic (B&B) je triviální, co to znamená. Pro metodu dynamické programování s dekompozicí podle ceny počítám jeden navštívený stav jako čtení/zápis zahrnující buňkou která není nekonečno. U heuristiky redux je jeden navštívený stav porovnání zda se zadaná věc ještě vejde do batohu.

Počet instancí a jejich velikost v generátoru byly zvoleny co nejtěžší a největší tak, aby jeden běh netrval na mém počítači více jak 5 minut. Změny parametrů jsem vždy krokoval nejméně čtyřikrát abych dodržel zadání[1].

Generování permutací jsem prováděl automatizovaně pomocí bash skriptu volající program kg_perm se vstupem předem vygenerovaným z generátoru kg2. Seed generátoru jsem volil defaultní statický. Abych mohl porovnávat kvalitu řešení bylo třeba vygenerovat vždy alespoň jedno správné řešení. Pokud řešení neexistuje, skript to pozná a vygeneruje ho pomocí metody dynamického programování jako referenční.

2.1 Popis metod

Různé algoritmy výpočtu jsou implementovány jako metody třídy **KnapsackInstance**, která reprezentuje jeden problém baothu. Hrubá síla je implementována pomocí rekurzivní funkce. Metoda větví a hranic je její rozšíření o ořezávání. Dynamické programování využívá dekompozici podle ceny. Heuristika redux je implementována běžným způsobem.

2.2 Pilotní experimenty

Podezření na závislosti ze zadání jsem se pokusil prověřit experimentací. Výkon metod vycházející z rozdílných vztahů prázdný/plný batoh jsem se rozhodl neověřovat, protože žádnou takovou metodu nemám z předchozích cvičení naimplementovanou.

Při experimentování s parametry generátoru a permutátoru podle zadání jsem pojal podezření na následující vztahy.

- Dynamické programování a Metoda větví a hranic nebudou robustní (3.1).
- Dynamické programování s dekompozicí podle ceny se zhoršuje s maximální cenou (3.2).
- Složitost Metody větví a hranic se zvyšuje s korelací cena–váha (3.3).
- Vyšší poměr kapacity batohu k sumární váze zlepšuje úspěšnost heuristiky redux (3.4).
- Nižší granularita zlepšuje kvalitu redux a zhoršuje složitost metody větí v hranic (3.5).

[1] https://moodle-vyuka.cvut.cz/mod/assign/view.php?id=89700

3 Naměřené hodnoty a interpretace výsledků

Tato kapitola shrne a prezentuje výsledky provedených experimentů.

3.1 Robustnost

U hrubé síly není třeba měřit robustnost, protože algoritmus jistě je robustní. Nedělá chyby a jeho složitost je neměnná 2^n. Heuristika redux je také jistě robustní, protože první krok je prvky seřadit. Průchod permutací je tedy identický. Složitost řazení do měřené složitosti nepočítáme.

Výsledky pilotního experimentu naznačují (s instancemi o deseti věcech), že zbylé dva algoritmy by mohly být na permutaci vstupu závislé. Generované instance jsou vytvářeny podle parametrů popsaných v tabulce 3.1. Všechny vygenerované hodnoty jsou přiloženy ve složce **data** a **calculated_X**. Měřit kvalitu výsledků nemá smysl, protože obě dvě zbylé metody dávají vždy správné výsledky.

Věcí	10
Instancí	100
Max. váha	1000
Max. cena	1000
Kapacita:Celková váha	0.8
Těžké věci	bal
Korelace váha/cena	uni
Granularita	1

Figure 3.1 Parametry generátoru pro testování robustnosti

Kvůli velému rozdílu hodnot jsem se rozhodl robustnost rozdělit do dvou grafů. Graf 3.2 pro metodu větví a hranic a graf 3.3 pro dynamické programování s dekompozicí podle ceny.

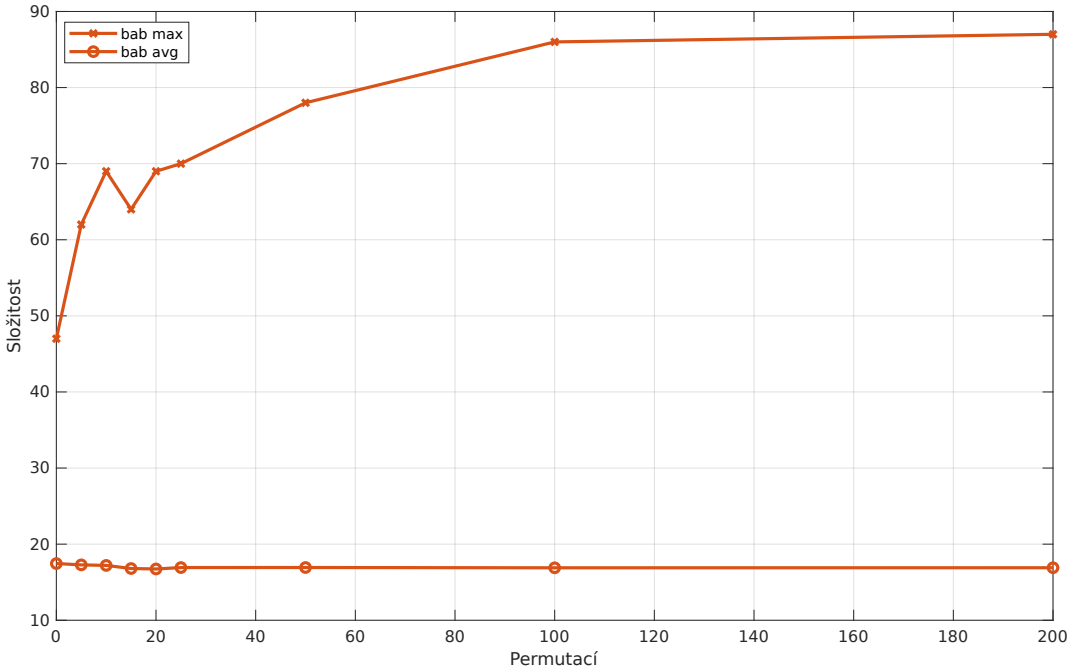

Figure 3.2 Náročnost metody větví a hranic při permutacích

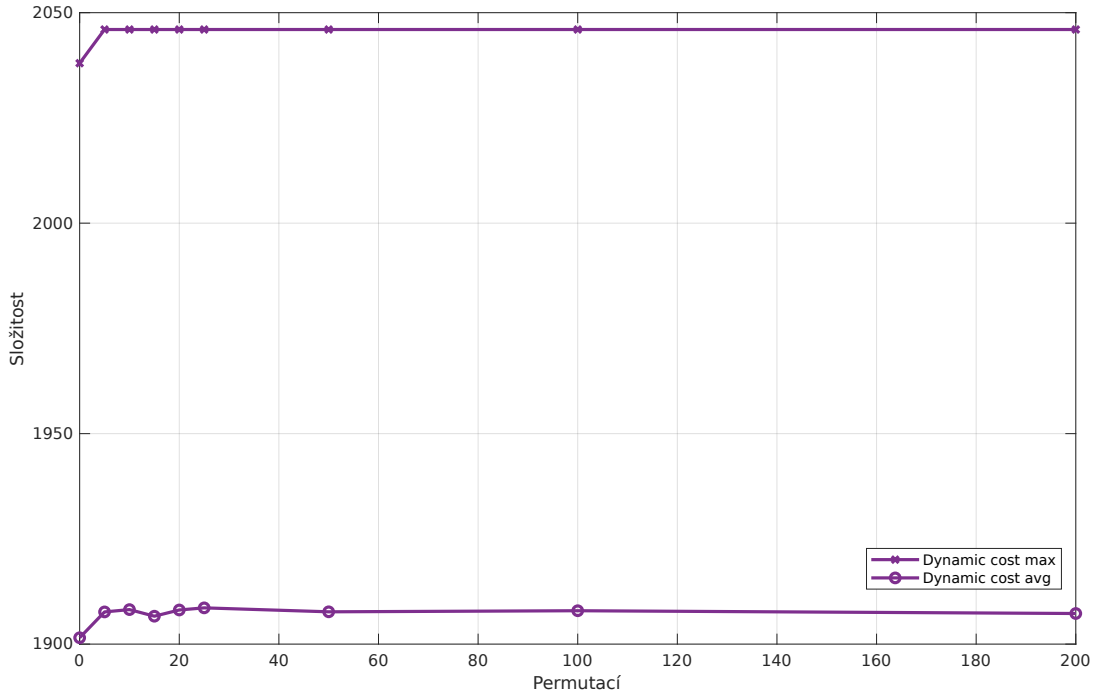

Figure 3.3 Náročnost dynamického programování při permutacích

3.2 Dynamic se zhoršuje s rostoucí max cenou

Změnu maximální ceny jsem škáloval tak, aby byly hodnoty v intervalu násobně nižší i vyšší než maximální váha. Konkrétní hodnoty generátoru jsou k dispozici v tabulce 3.2.

Kvůli velikému rozdílu hodnot jsem se znovu rozhodl rozdělit prezentaci do dvou grafů. Graf 3.5 pro metodu větví a hranic společně s redux. Graf 3.6 zachycuje dynamické programování s dekompozicí podle ceny a hrubou sílu. Kvalita výpočtu je relevantní pouze u redux a tam je konstatní na maximální hodnotě 4.9 procent s odchylkou nejvýše 0.1 a průměrné hodnotě 0.69 procent s odchylkou nejvýše 0.05.

Věcí	15
Instancí	100
Max. váha	500
Max. cena	**100/250/500/750/1000/1250/1500**
Kapacita:Celková váha	0.8
Těžké věci	bal
Korelace váha/cena	uni
Granularita	1

Figure 3.4 Parametry generátoru pro testování zvedání maximální ceny

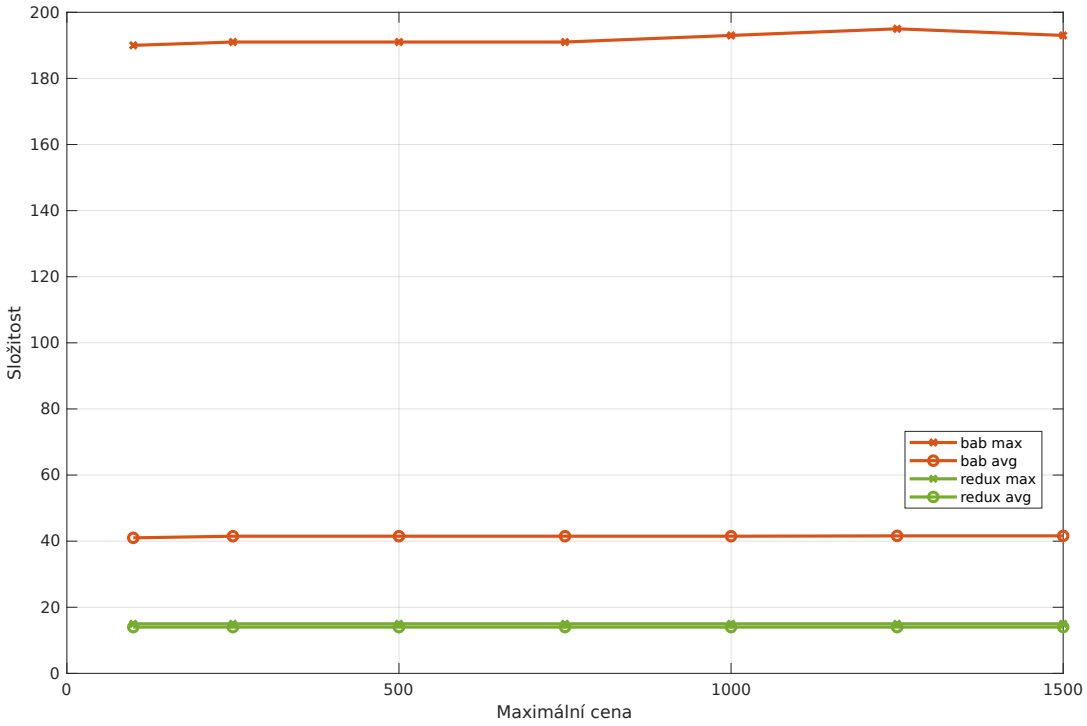

Figure 3.5 Náročnost metod redux a větví a hranic při zvedání maximální ceny

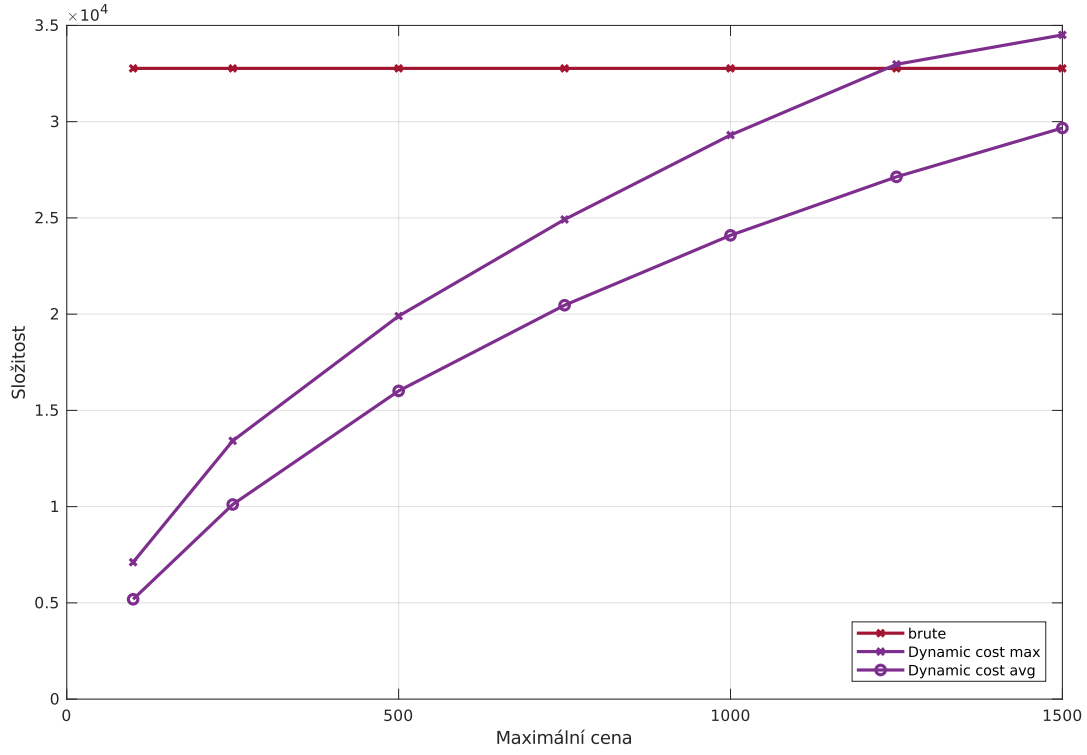

Figure 3.6 Náročnost dynamického programování a hrubé síly při zvedání maximální ceny

3.3 BaB se zhoršuje s korelací cena-váha

Generátor dovoluje nastavit korelaci pouze na tři stupně (uni, corr, strong). Abyh vyhověl požadavkům na alespoň čtyři hodnoty vykreslil jsem tyto hodnoty jako tři rozdíle křivky při rostoucím počtu věcí v batohu. Složitost hrubé síly je konstantní vůči změnám korelace. Složitost redux byla mírně nižší při hodnotách **corr**, ale v zájmu nepřehlcení čtenáře jsem z toho další graf nevykresloval.

Věcí	**10/14/16/18/20**
Instancí	200
Max. váha	1000
Max. cena	1000
Kapacita:Celková váha	0.8
Těžké věci	bal
Korelace váha/cena	**uni/corr/strong**
Granularita	1

Figure 3.7 Parametry generátoru pro testování korelace váha/cena

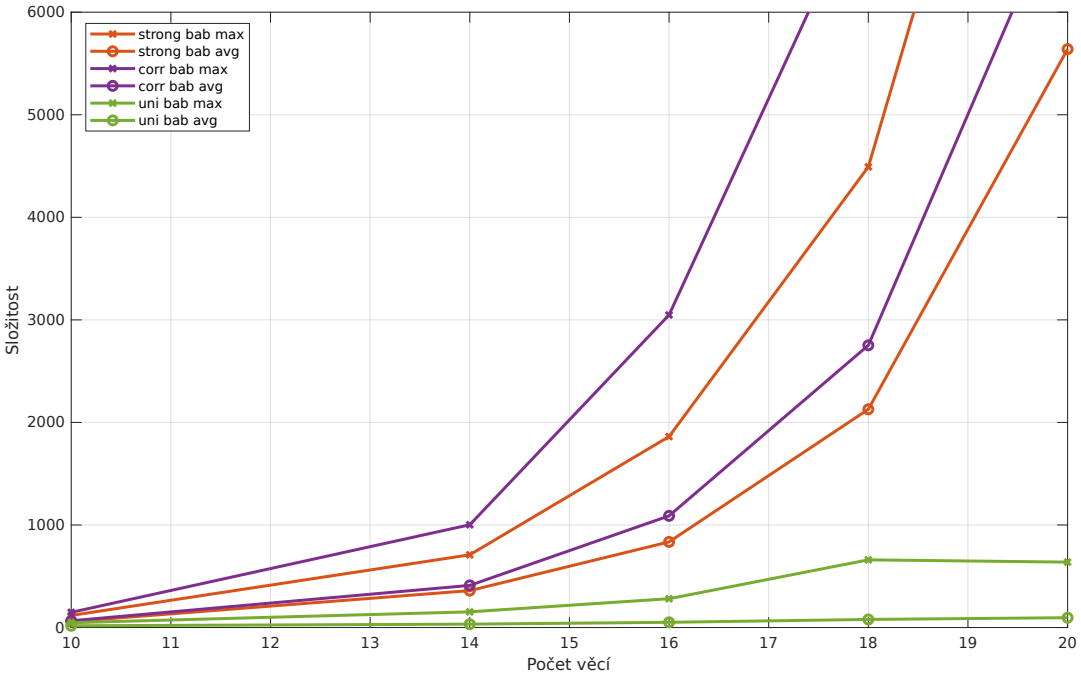

Figure 3.8 Náročnost metody větví a hranic pro druhy korelací váha/cena

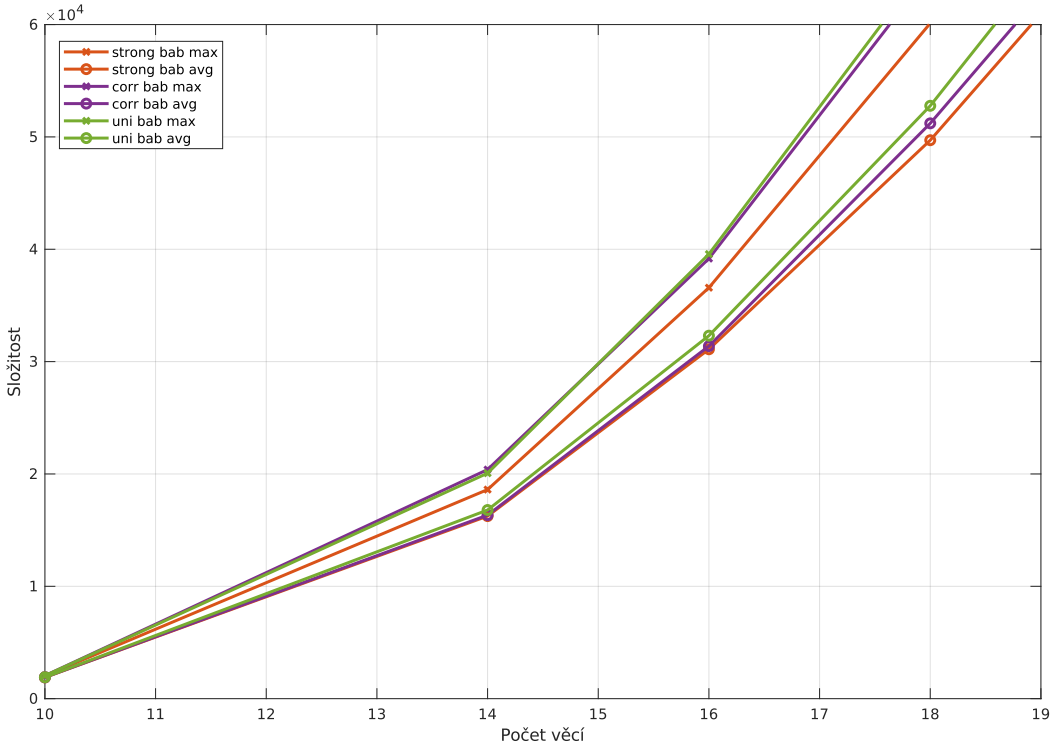

Figure 3.9 Náročnost dynamického programování pro druhy korelací váha/cena

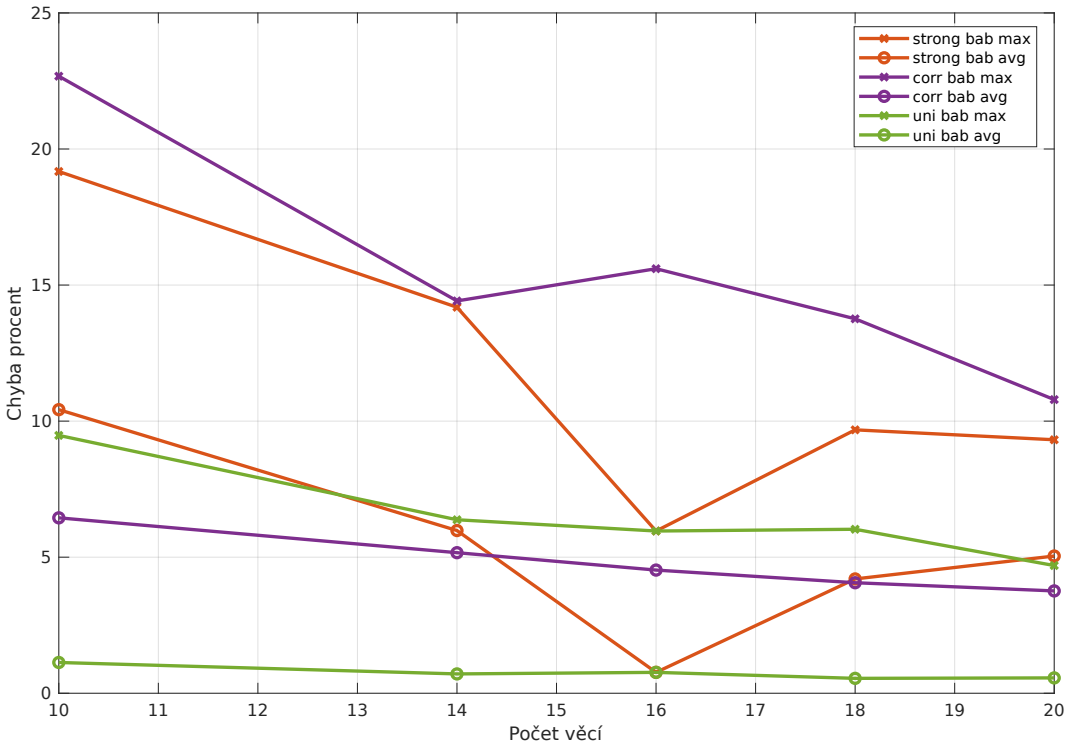

Figure 3.10 Chyba redux pro druhy korelací váha/cena

3.4 Chyba redux závisí na poměru kapacity k sumární váze

Z proběhlého výpočtu jsem vykreslil data pro redux jak ohledně chyby tak složitosti do jednoho grafu. Graf 3.12 tak má jako osu y zároveň chybovost procent, tak složitost v počtu navštívených konfigurací.

Věcí	20
Instancí	200
Max. váha	1000
Max. cena	1000
Kapacita:Celková váha	**0.1/0.3/0.5/0.7/0.8/0.9/0.99**
Těžké věci	bal
Korelace váha/cena	uni
Granularita	1

Figure 3.11 Parametry generátoru pro testování poměru kapacita – celková váha

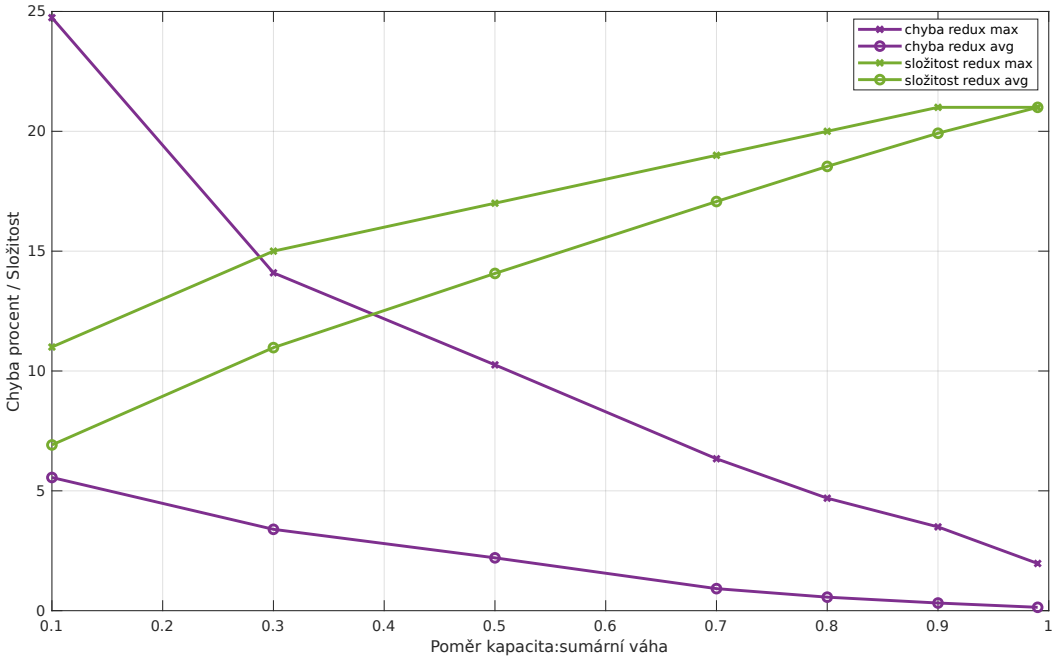

Figure 3.12 Chyba redux pro zvyšující se poměr kapacita – celková váha

3.5 Granularita ovlivňuje kvalitu redux a složitost metody větí a hranic

Protože generátor umožňuje nastavení granularity pro těžké i lehké věci, vykresil křivky v grafech pro změny granularity pro obojí. Složitost/chybovost pro hodnotu **uni** můžete nalézt v tabulce 3.14. Složitost heuristiky redux je pro všechny hodnoty konstantní s maximální odchylkou 2, a nevykazuje žádné známky změny, proto není vykreslena.

Metoda	max	avg
Hrubá síla	1048576 / 0.0	1048576 / 0.0
Metoda větví a hranic	775 / 0.0	103 / 0.0
Dynamické programování	102133 / 0.0	77908 / 0.0
Redux	20 / 4.7	18.5 / 0.6

Figure 3.13 Složitost a chybovost metod pro rozložení **uni**

Věcí	20
Instancí	300
Max. váha	1000
Max. cena	1000
Kapacita:Celková váha	0.8
Těžké věci	**bal/light/heavy**
Korelace váha/cena	uni
Granularita	**0.1/0.3/0.5/0.7/1**

Figure 3.14 Parametry generátoru pro testování poměru kapacita – celková váha

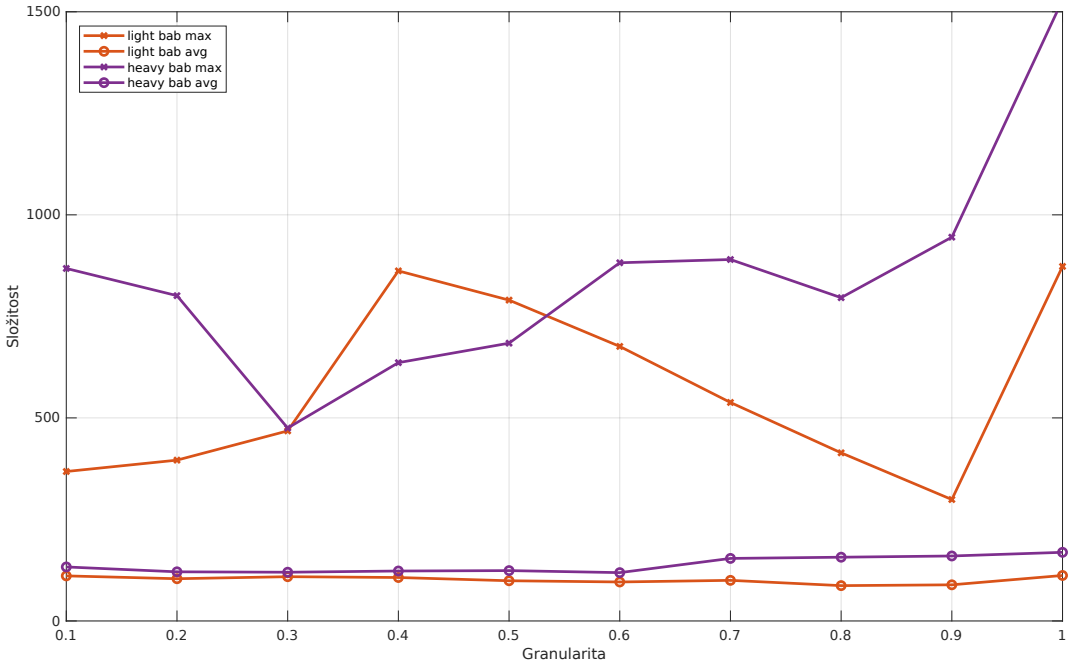

Figure 3.15 Složitost metody větví a hranic pro zvyšující se granularitu

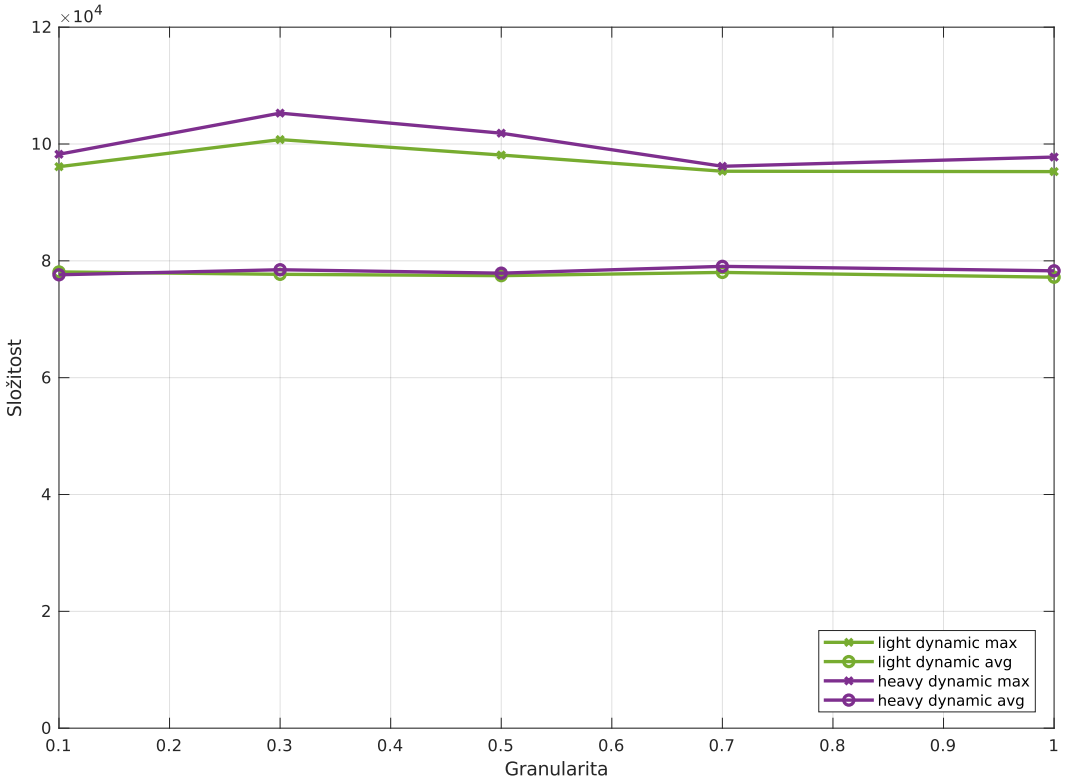

Figure 3.16 Složitost dynamického programování pro zvyšující se granularitu

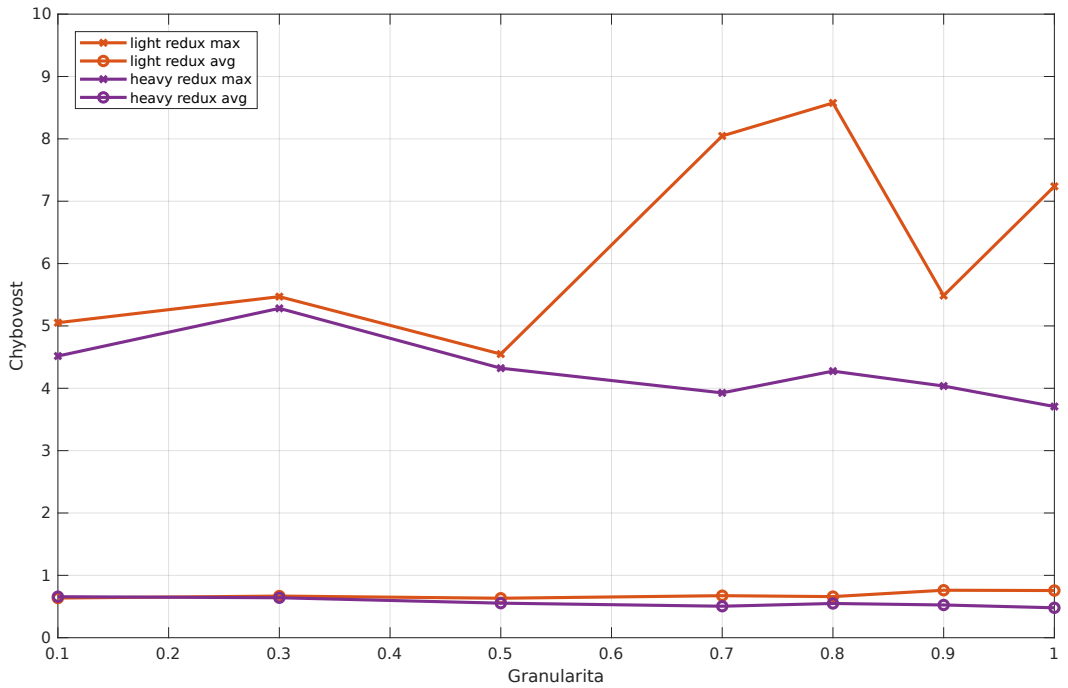

Figure 3.17 Chybovost redux pro zvyšující se granularitu

4 Závěr

4.1 Vyhodnocení robustnosti

Na grafu 3.2 je zřetelně vidět závislost počtu permutací na maximální složitosti metody větí v hranic. Průměrnou složitost to ale neovlivňuje. Po vyhodnocení grafu 3.3 s dynamickým programováním se původní předpoklad zásadní nerobustnosti neprokázal. I přesto je ale vidět, jak při nejnižší hodnotě je nižší jak maximální složitost, tak i průměrná.

4.2 Vyhodnocení maximální ceny

Z grafu 3.5 je evidentní, že zvyšování maximální ceny nijak neovlivňuje metodu větví a hranic ani metodu redux. Hrubá síla je jako vždy také neměnná. Dynamické programování s dekompozicí podle ceny ale ukazuje jasný trend. S rostoucí maximální cenou se zvyšuje jak maximální tak průměrná složitost.

4.3 Vyhodnocení korelace cena/váha

Výsledky pro metodu větví a hranic jsou přesvědčivé. Nejvyšší složitosti nabývá metoda při střední korelaci (corr). Největší rozdíl je patrný hlavně při vyšších hodntách a srovnání korelace uni a zbylých dvou, viz graf 3.8. Náročnost dynamického programování i redux není zvyšováním korelace příliš ovlivněna. Chybovost redux v grafu 3.10 ukazuje, že korelace uni dosahuje vždy nejlepších vlastností.

4.4 Vyhodnocení poměru kapacity k sumární váze

Z naměřených výsledků je zřemé, že pro rostoucí poměr kapacity batohu k sumární váze roste složitost heuristiky redux. S rostoucím poměrem ale také jasně klesá chybovost.

4.5 Vyhodnocení granularity

Zvyšující se granularita má malý vliv na průměrnou hodnotu všech měřený metod. U b&b a dynamického programování je o trochu lepší průměrná složitost u lehkých věcí než u těžkých. Naopak chybovost redux vychází ve prospěch těžších věcí. Maxilmální hodnoty složitosti se chovají u dynamického programování podobně. Maximální hodnoty složitosti pro b&b mají svá lokální minima, ale konkrétní závislost jsem nevyl schopen vysledovat. Stejně tak u maximální chybovosti pro redux.