1 MI-PAA, Úloha 3: Experimentální hodnocení kvality algoritmů

Marián Hlaváč, 30 Nov 2017 (hlavam30)
marian.hlavac@fit.cvut.cz
https://github.com/mmajko/knapsack-problem

1.1 Zadání úlohy

- Prozkoumejte citlivost metod řešení problému batohu na parametry instancí generovaných generátorem náhodných instancí. Máte-li podezření na další závislosti, modifikujte zdrojový tvar generátoru.
- Na základě zjištění navrhněte a proveďte experimentální vyhodnocení kvality řešení a výpočetní náročnosti

1.2 Postup

Aby byl celý proces jednodušší, rozhodl jsem se každý experiment udělat stejný, se stejnými výstupními daty a měnit pouze vstupní parametry takového experimentu. Jako výstup jsem zvolil dvě závislosti: Závislost časové náročnosti na zkoumaném parametru, a vyzobrazení dosažené spočtené ceny v závislosti na zkoumaném parametru. Druhý výstup je mírně zavádějící (jako uvedu níže).

Nejdříve jsem otestoval generátor, který byl poskytnut k plnění této úlohy. Muselo dojít k úpravě generátoru, jelikož jako seed pro náhodná čísla používal aktuální čas ve vteřinách, což v jedné stejné vteřině generovalo stejné výsledky.

Než jsem začal zkoumat parametry, zkoumal jsem, jaký je šum při stejných parametrech po několika opakování. Následně jsem se rozhodl postupovat zkoumáním každého parametru pro každý algoritmus a pro každý parametr uvést neobvyklé výsledky, které bylo možné pozorovat.

1.3 Samostatný notebook s daty

Zpráva je pro tento úkol mírně odlišná, než byly zprávy pro první dva úkoly. V tomto notebooku nejsou uvedeny žádné přímé výpočty ani plné formy grafů, pouze se zde nachází text zprávy s výsledky a závěry. Většina práce na tomto domácím úkolu byla odvedena v druhém notebooku, kde se nacházejí pouze "surová" data, bez závěrů a výsledků:

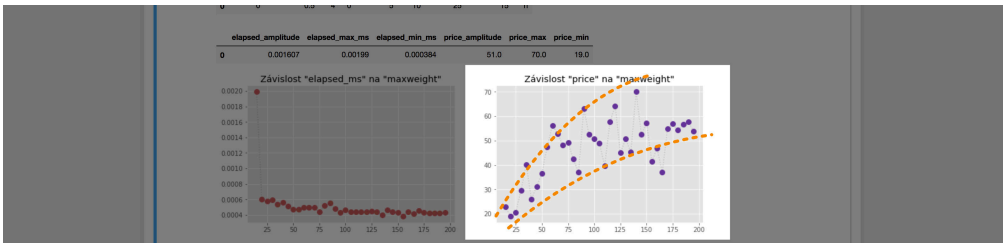
notebook na GitHubu, HTML notebook, PDF notebook

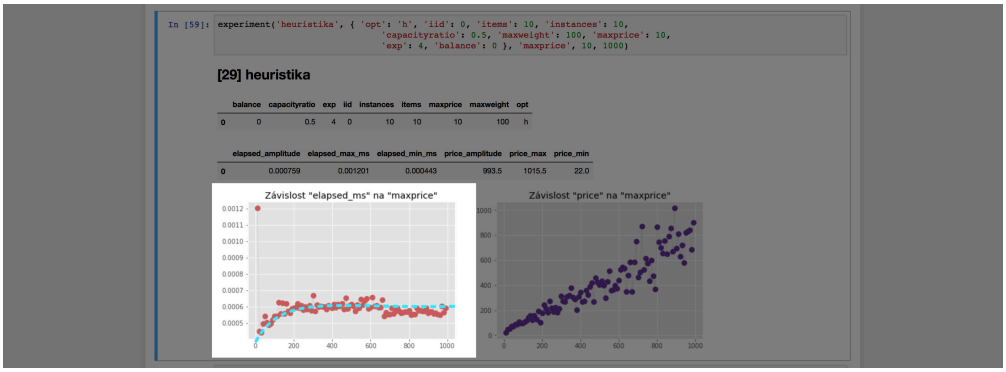
1.4 Šum v základu

Pokusil jsem se ověřit, zdali průměrem z velkého počtu instancí nedosáhneme výsledků bez výrazného šumu (který je přítomen díky náhodnosti). Experiment ukázal, že nedochází k tak velké redukci šumu, tudíž by zvýšení instancí pro průměrování v každém experimentu bylo bezpředmětné a zbytečně by to přidalo na času výpočtu.

1.5 Zkoumané parametry

Jak už jsem výše uvedl, pro každý experiment jsem měl jednotně stanoveny výstupy. Oba výstupy jsou schopné popsat kvalitu algoritmu, ovšem každý jiným způsobem. Pro zkrácení budu tyto dva grafy dále v textu označovat jako časový (průmerný čas výpočtu vs parametr) a cenový (průměrná cena batohu vs parametr).





Druhý výstup -- závislost vypočtené průměrné ceny na zvoleném parametru, je často zavádějící údaj a tak je nutné vyvozovat z něj závěry velmi opatrně. Příkladem je graf vypočtených cen v experimentu, ve kterém jsme měnili maximální cenu věci. Je zřejmé, že pokud máme nastaven experiment tak, že je dostupných pouze 10 předmětů, a jejich maximální cena je 1, tak se nemůže stát, aby celková cena batohu přesáhla číslo 10. Přesně to se projevuje na grafu.

1.5.1 Maximální váha věcí

Nejvýrazněji se projevil algoritmus s heuristikou poměru cena/váha. Na časovém grafu lze pozorovat mírný klesající trend, ale ten je pravděpodobně důsledkem režie algoritmu (práce s pamětí, loopy...), tedy není tak důležitý.

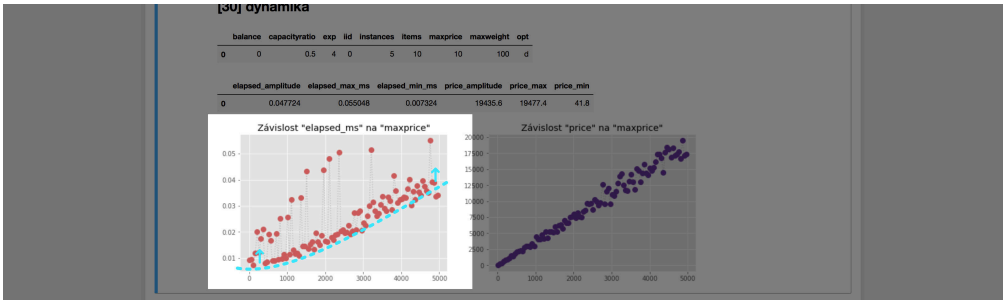
Zajímavější je stoupající trend na cenovém grafu. Ostatní grafy při zkoumání maximální váhy jsou naprosto náhodné a nevykazují žádný trend, u heuristiky je však možné pozorovat, že průměrná váha stoupá.

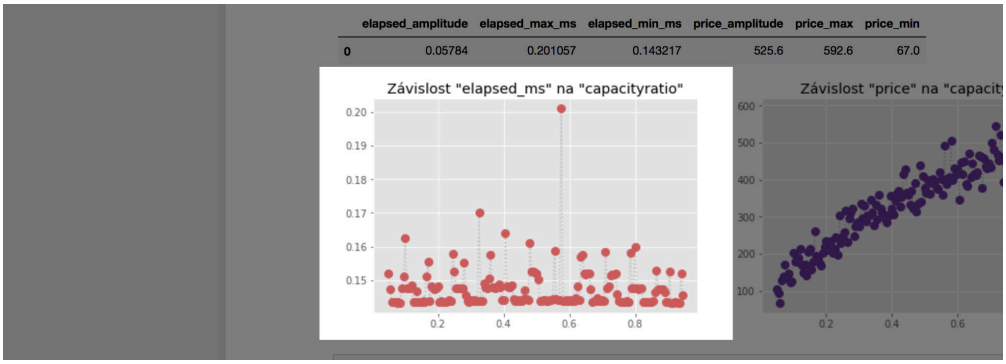
Lze tedy předpokládat, že s větší maximální váhou věcí je heuristika více přesná a podává tak kvalitnější řešení.

1.5.2 Maximální cena věcí

U maximální ceny věcí se opět projevil heuristický algoritmus (což dává smysl, když pracuje s poměrem cena/váha), tentokrát lze pozorovat zmírnění časové náročnosti při nízkých hodnotách maximální ceny věcí. Tady lze předpokládat snížení časové náročnosti díky tomu, že heuristický algoritmus byl schopen rychleji naplnit batoh, aniž by porušoval omezující podmínky (jinými slovy se mu podařilo naplnit batoh "na první dobrou").

Mnohem zajímavější reakci předvedl algoritmus založený na dynamickém programování. U něj lze pozorovat téměř lineárně stoupající minimální časovou náročnost v závislosti na maximální cenu





věcí.

Pokud by se jednalo o algoritmus s dekompozicí podle váhy, pravděpodobně bychom se dočkali zajímavého výsledku u předchozích experimentů, kde jsme měnili maximální váhu věcí.

Algoritmus se stále projevoval velmi náhodnou dobou výpočtu, ale na grafu lze jasně pozorovat vzrůstající dolní mez.

Z toho lze odvodit, že tento algoritmus je nejefektivnější pro instance, ve kterých se nacházejí věci se spíše nízkou cenou.

1.5.3 Poměr kapacity k sumární váze

Na cenových grafech všech algoritmů lze pozorovat dva druhy "chování". Křivka, kterou výsledky na grafu vytvářejí je buď konvexní nebo konkávní. Důvod tohoto zjištění se mi nepodařilo objasnit.

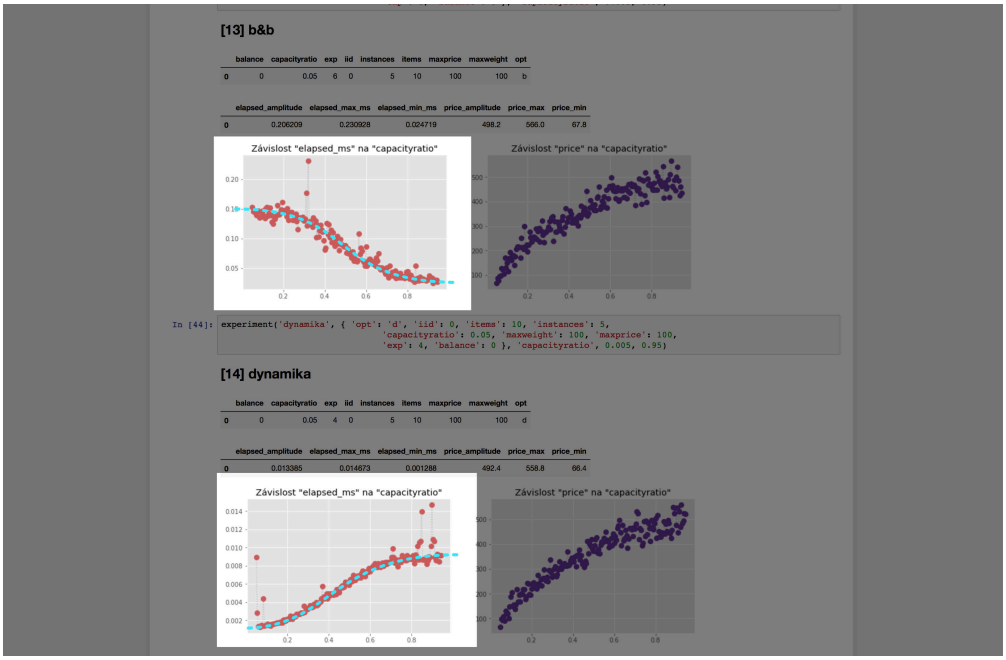
Další zajímavostí, které jsem nepřišel na přesný důvod je zajímavé chování rekurzivního algoritmu při změně poměru. Pravděpodobně všechny průběhy rekurzivního algoritmu jsou podobné, ale na tomto grafu je to nejzjevnější - doby trvání algoritmu, zdá se, přesně kopírují hloubku rekurze, do které se algoritmus dostal.

Velmi zřejmou reakci ukázaly algoritmy B&B a dynamické programování.

Dalo by se i tvrdit, že jsou v této vlastnosti tyto dva algoritmy inverzí. Zatímco B&B byl efektivní při menší sumární váze vůči kapacitě, dynamické programování bylo efektivnější při menší kapacitě vůči sumární váze.

1.5.4 Granularita instancí

U granularity se mi bohužel nepodařilo pozorovat výraznější pattern v grafech. Velmi lehce naznačovaly časové grafy algoritmu B&B nějaký vztah, ale nepovažoval bych je jako potvrzující.



1.6 Závěr

Některé z grafů poukázaly na zajímavé vztahy, jak ovlivňují vlastnosti vstupní dat výpočty problému batohu.

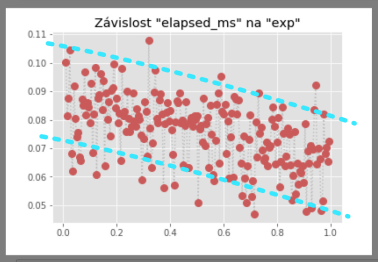
Nejvýraznější vliv jsem pozoroval u maximální ceně věcí na dynamickém programování a u poměru kapacity k sumární váze na branch & bound a dynamickém programování.

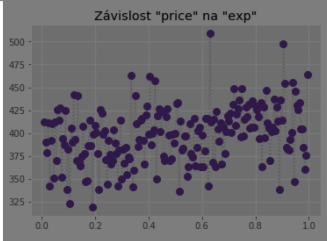
Pro jistotu opět uvedu odkaz, který je již uveden výše, na druhý dokument, zobrazující data, které jsem sesbíral a se kterými jsem pracoval: notebook na GitHubu, HTML notebook, PDF notebook

[22] balance -1, b&b

	balance	capacityratio	exp	iid	instances	items	maxprice	maxweight	opt
0	-1	0.5	0.01	0	5	10	100	100	b

	elapsed_amplitude	elapsed_max_ms	elapsed_min_ms	price_amplitude	price_max	price_min
0	0.060835	0.107778	0.046943	189.6	508.8	319.2



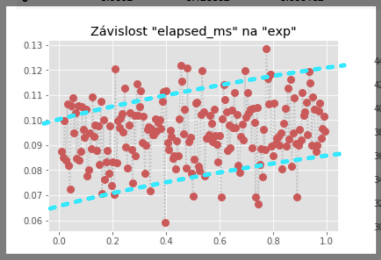



```
In [53]: experiment('balance +1, b&b', { 'opt': 'b', 'iid': 0, 'items': 10, 'instances': 5,
                                        'capacityratio': 0.5, 'maxweight': 100, 'maxprice': 100,
                                        'exp': 0.01, 'balance': 1 }, 'exp', 0.005, 1)
```

[23] balance +1, b&b

	balance	capacityratio	exp	iid	instances	items	maxprice	maxweight	opt
0	1	0.5	0.01	0	5	10	100	100	b

	elapsed_amplitude	elapsed_max_ms	elapsed_min_ms	price_amplitude	price_max	price_min
0	0.0692	0.128382	0.059182	146.8	450.2	303.4




```
In [54]: experiment('balance 0, dynamika', { 'opt': 'd', 'iid': 0, 'items': 10, 'instances': 5,
```