Problém batohu - 3. úloha

MI-PAA, zimný semester 2014

Martin Klepáč

November 16, 2014

1 Definícia problému

Problém batohu (knapsack problem) je jeden z NP-ťažkých problémov. Pokiaľ máme k dispozícii množinu predmetov, pričom každý predmet má svoju váhu, cenu, a maximálnu nosnosť batohu, potom je naším cieľom vybrať predmety do batohu tak, aby sme maximalizovali ich cenu a pritom celková hmotnosť predmetov nepresiahla maximálnu povolenú nosnosť batohu.

2 Formát vstupu

Formálne, vstup vyjadríme pomocou

- n = počet predmetov
- M = maximálna nosnosť batohu
- *vektor* w = hmotnosť jednotlivých predmetov
- *vektor* c = hodnota (cena) jednotlivých predmetov

Výstupom algoritmu je vektor x, ktorý udáva prítomnosť/neprítomnosť daného predmetu vo výbere. Ďalej požadujeme informáciu o celkovej hmotnosti predmetov v batohu a ich súhrnú hodnotu pre najlepší výber.

3 Úvod

V tejto časti sa pozrieme na experimentálne vyhodnotenie výsledkov algoritmov, ktoré sme implementovali v predchádzajúcich dvoch iteráciách. Pre potreby tejto úlohy budeme uvažovať o algoritmoch dynamického programovania, branch and bound s použitím orezávania stavového priestoru a heuristika cena-váha. K tomu budeme potrebovať generátor náhodných inštancií, ktorý budeme parametrizovať tak, že v každom momente budeme meniť hodnotu práve 1 parametru a všetky ostatné zachováme konštantné.

4 Predpoklady

Algoritmus dynamického programania z predchádzajúcej iterácie bol založený na dekompozícii podľa kapacity. Z toho vychádza, že pokiaľ budeme meniť hodnotu kapacity batohu, doba behu algoritmu by sa mala meniť lineárne - zložitosť je závislá lineárne ako na veľkosti inštancie, tak aj na nosnosti batohu - $O(nM)$

V prípade heuristiky cena-váha neočakávam, že doba behu je závislá na akomkoľvek parametri s výnimkou usporiadanosti vstupných dát, a to z toho dôvodu, že po usporiadaní algoritmus lineárne prechádza zoznamom predmetov, až kým nedôjde k naplneniu batohu. Dalo by sa namietať, že s menšou nosnosťou batohu sa doň zmestí menší počet vecí, a teda prechod poľom usporiadaných predmetov sa zrýchli. Vzhľadom na to, že zložitosť usporiadania $O(n \log n)$ je rádovo vyššia ako zložitosť prechodom poľa $O(n)$, je tento faktor zanedbateľný.

Nakoniec, v prípade algoritmu branch and bound očakávam, že sa prejaví schopnosť efektívne orezať stavový priestor. To znamená, že v prípade obmedzenej nosnosti batohu voči váham predmetov dokáže algoritmus rýchlo odstrániť neperspektívne vetvy, a tým zrýchliť trvanie behu.

V súvislosti s heuristikou cena-váha a charakterom predmetov vo výbere nie som schopný vysloviť žiadne predpoklady.

5 Výsledky

Školský generátor, pokiaľ nie je inak napísané, som spúšťal s nasledujúcimi parametrami:

- $m = 0.5$
- $d = 0$
- $k = 1$
- $w\text{-}max = 250$
- $c\text{-}max = 250$
- $n = 25$
- $N = 300$

Na nasledujúcich riadkoch popíšem závislosti, ktoré som pomocou generátora objavil. Pre hore uvedenú trojicu algoritmov budem merať, ako sa mení priemerná doba behu, kým pre heuristiku cena-váha budem ďalej sledovať zmenu relatívnej chyby s meniacim sa parametrom. Závislosť na konkrétnom parametri vždy vyjadrím formou dvojice tabuliek a grafov.

V prvom kroku sme za variabilnú zložku určili **maximálnu váhu predmetu** na vstupe. Z grafu 1 je zrejmá lineárna závislosť zložitosti dynamického programovania na maximálnej váhe predmetu. Lineárna závislosť je dôsledkom dekompozície podľa

kapacity batohu (viz kapitola Predpoklady) a ďalej tým, že nosnosť batohu je funkcia súčtu váh predmetov na vstupe. Trvanie metódy branch and bound a heuristiky vykazuje drobné fluktuácie, ale bez nejakého dlhodého trendu. Podobná úvaha platí pre relatívnu chybu heuristiky, ktorá sa pohybuje jemne nad pol percentom. Jemné zvýšenie rel. chyby nastáva pre *w-max* sa približne rovná *c-max*, t.j. vtedy keď pomer cena-váha pre jednotlivé predmety sa rovná 1. Relevantné grafy 1 a 2 a tabuľky 1 a 2.

	BB	**DP**	**CV**
w-max=40	0.002 405	0.000 102	0.000 004 020
w-max=60	0.002 726	0.000 153	0.000 004 520
w-max=80	0.002 626	0.000 201	0.000 004 600
w-max=100	0.002 750	0.000 249	0.000 004 610
w-max=125	0.002 757	0.000 308	0.000 004 620
w-max=150	0.003 343	0.000 353	0.000 004 880
w-max=175	0.002 734	0.000 437	0.000 005 090
w-max=200	0.002 957	0.000 507	0.000 005 020
w-max=250	0.002 739	0.000 636	0.000 005 330
w-max=300	0.002 606	0.000 765	0.000 005 010
w-max=350	0.002 785	0.000 889	0.000 005 310
w-max=400	0.002 653	0.001 025	0.000 005 330
w-max=450	0.002 649	0.001 133	0.000 005 150
w-max=500	0.002 698	0.001 259	0.000 005 410

Table 1: Závislosť trvania behu algoritmov BB,DP a heuristiky od maximálnej hmotnosti [s]

	CV
w-max=40	0.55638
w-max=60	0.61646
w-max=80	0.54155
w-max=100	0.50933
w-max=125	0.53428
w-max=150	0.49504
w-max=175	0.50213
w-max=200	0.54508
w-max=250	0.58260
w-max=300	0.59298
w-max=350	0.60938
w-max=400	0.54125
w-max=450	0.53188
w-max=500	0.52994

Table 2: Závislosť relatívnej chyby heuristiky cena/váha od maximálnej hmotnosti [%]

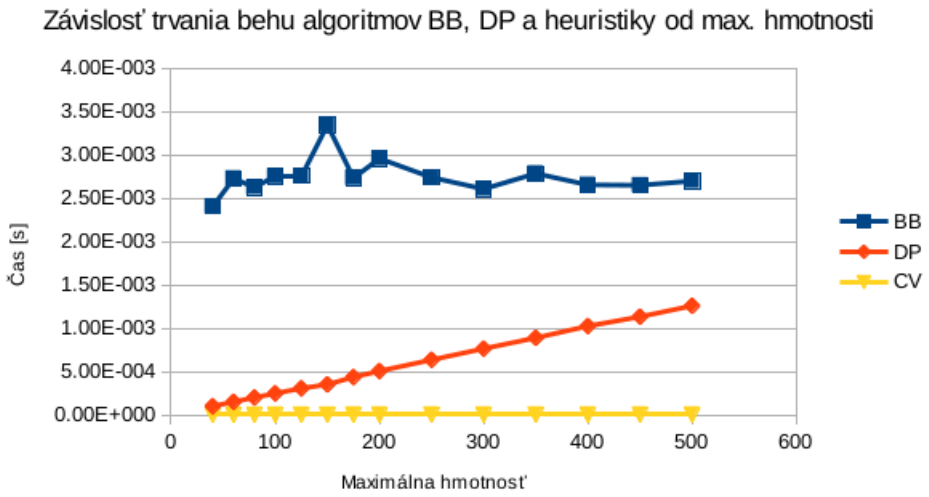

Figure 1: Závislosť trvania behu algoritmov BB, DP a heuristiky od max. hmotnosti

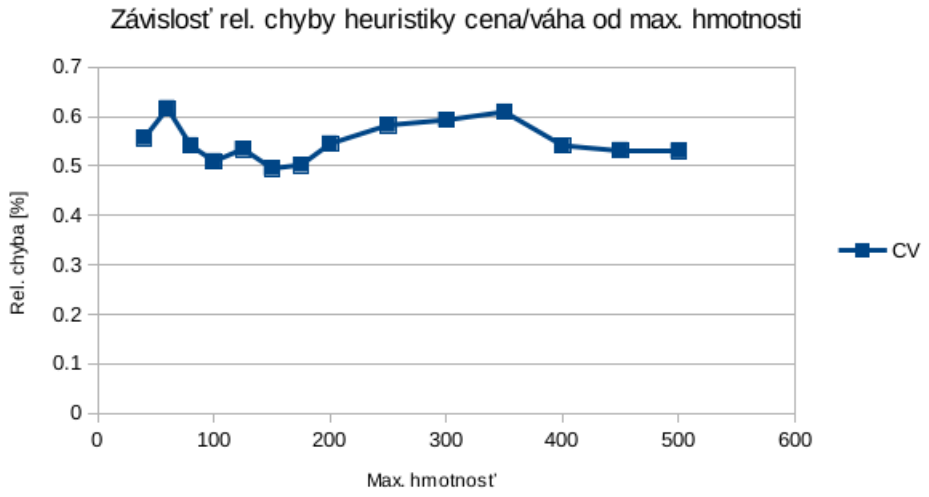

Figure 2: Závislosť rel. chyby heuristiky cena/váha od max. hmotnosti

V druhom kroku sledujeme efekt zvyšovania **maximálnej ceny predmetov**. V tomto prípade platí, že trvanie všetkých algoritmov je cenovo necitlivé. Iná situácia by nastala v prípade, ak by dynamické programovanie bolo implementované dekompozície podľa ceny. V tom prípade by sa ukázala lineárna závislosť trvania algoritmu od maximálnej ceny na vstupe, podobne ako to bolo v mojom prípade pre dekompozíciu

podľa kapacity - viz predchádzajúci bod. Relatívne chyba heuristiky opäť fluktuuje mierne nad pol percentom, ale znovu sa nedá hovoriť o nejakej očividnej závislosti. Relevantné grafy 3 a 4 a tabuľky 3 a 4.

	BB	**DP**	**CV**
c-max=40	0.002 798	0.000 659	0.000 005 380
c-max=60	0.002 654	0.000 638	0.000 005 020
c-max=80	0.002 572	0.000 637	0.000 005 040
c-max=100	0.002 770	0.000 638	0.000 005 030
c-max=125	0.002 867	0.000 640	0.000 005 060
c-max=150	0.002 651	0.000 638	0.000 004 910
c-max=175	0.002 611	0.000 634	0.000 004 940
c-max=200	0.002 695	0.000 644	0.000 005 140
c-max=250	0.002 573	0.000 641	0.000 005 080
c-max=300	0.002 686	0.000 647	0.000 005 370
c-max=350	0.002 861	0.000 637	0.000 005 080
c-max=400	0.002 736	0.000 646	0.000 005 350
c-max=450	0.002 790	0.000 655	0.000 005 500
c-max=500	0.002 686	0.000 650	0.000 005 460

Table 3: Závislosť trvania behu algoritmov BB,DP a heuristiky od maximálnej ceny [s]

	CV
c-max=40	0.56719
c-max=60	0.5142
c-max=80	0.55552
c-max=100	0.47242
c-max=125	0.63001
c-max=150	0.49272
c-max=175	0.61052
c-max=200	0.50114
c-max=250	0.55035
c-max=300	0.54376
c-max=350	0.54897
c-max=400	0.59871
c-max=450	0.55344
c-max=500	0.56539

Table 4: Závislosť relatívnej chyby heuristiky cena/váha od maximálnej ceny [%]

V treťom kroku monitorujeme závislosť trvania behu resp. relatívnej chyby algoritmov od hodnoty prepínača generátora inštancií m, ktorý vyjadruje **pomer medzi kapacitou batohu a súčtom váh** predmetov na vstupe. Graf 6 ukazuje klesajúcu chybu heuristiky v závislosti od narastajúceho pomeru. Teda, čím sa do batohu zmestí re-

latívne väčší počet predmetov, tým je chyba heuristiky nižšia (limitne sa blíži k nule, pretože každý predmet sa do batohu zlepší). Dynamické programovanie vykazuje lineárny rast - opäť je to spôsobené lineárne sa zvyšujúcou kapacitou batohu, podobne ako tomu bolo pri maximálnej váhe. Algoritmus branch and bound dokáže pri nízkom pomere kapacity batohu a sume váh predmetov efektívne orezať statový priestor, ale táto vlastnosť sa s narastajúcim pomerom zhoršuje, čoho výsledkom je exponenciálny rast doby behu pre nízke m. Relevantné grafy 5 a 6 a tabuľky 5 a 6.

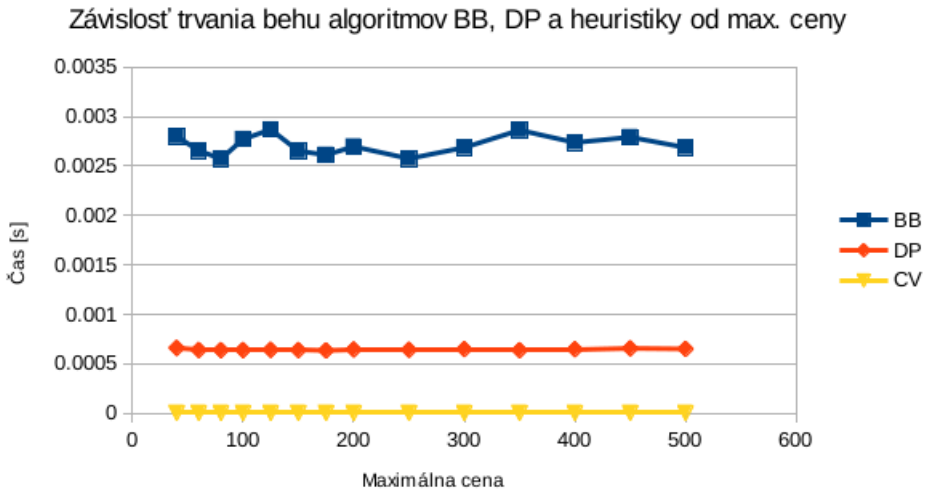

Figure 3: Závislosť trvania behu algoritmov BB. DP a heuristiky od max. ceny

	BB	**DP**	**CV**
pomer=0.1	0.000 039	0.000 1055	0.000 003 880
pomer=0.2	0.000 143	0.000 2274	0.000 003 950
pomer=0.3	0.000 447	0.000 3523	0.000 004 120
pomer=0.4	0.001 156	0.000 4783	0.000 004 170
pomer=0.5	0.002 673	0.000 6507	0.000 005 520
pomer=0.6	0.004 853	0.000 7976	0.000 006 050
pomer=0.7	0.007 638	0.000 9356	0.000 006 200
pomer=0.8	0.009 888	0.001 0469	0.000 006 060
pomer=0.9	0.010 618	0.001 1860	0.000 006 360

Table 5: Závislosť trvania behu algoritmov BB,DP a heuristiky od pomeru nosnosť/-suma váh [s]

Ako posledný faktor budeme meniť **granularitu dát** pomocou parametra k pre prevahu veľkých vecí (d=*1*). Platí, že čím väčšia hodnota tohto parametra, tým je menšia pravdepodobnosť malých dát na vstupe. Z toho vyplýva, že so zvyšujúcim sa

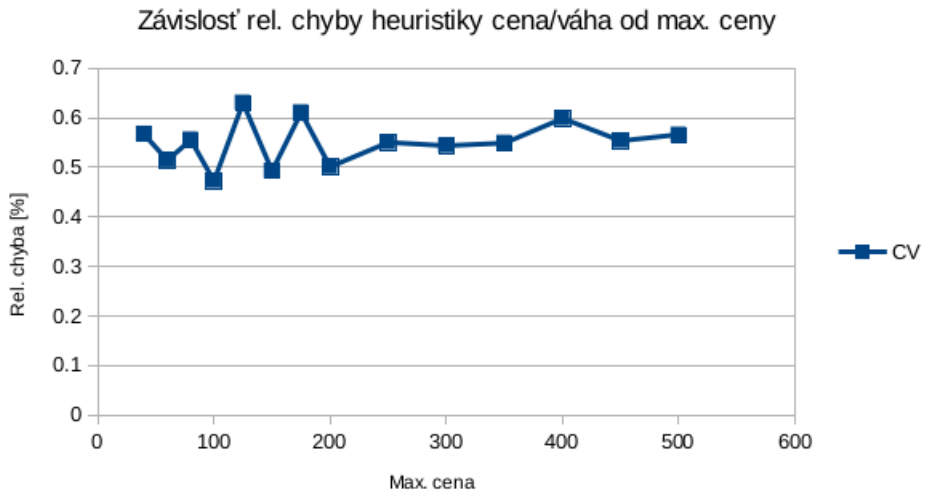

Figure 4: Závislosť rel. chyby heuristiky cena/váha od max. ceny

	CV
pomer=0.1	1.4738
pomer=0.2	1.0519
pomer=0.3	0.97004
pomer=0.4	0.78315
pomer=0.5	0.49983
pomer=0.6	0.3608
pomer=0.7	0.22867
pomer=0.8	0.17306
pomer=0.9	0.084475

Table 6: Závislosť relatívnej chyby heuristiky cena/váha od pomeru nosnosť batohu/-suma váh [%]

počtom veľkých predmetov (narastajúce k) rastie zároveň aj nosnosť batohu, a teda opäť dochádza k rastu doby behu metódy dynamického programovania. Naopak, doba behu algoritmu branch and bound klesá približne exponenciálne s narastajúcim počtom veľkých predmetov na vstupe až do fázy, v ktorej beží rýchlejšie ako dynamické programovanie. Beh heuristiky, ako tomu bolo vo všetkých predchádzajúcich prípadoch, nie je závislý na granularite vstupných dát. S narastajúcou hodnotou váh predmetov ale dochádza k zmenšovaniu relatívnych chýb až k hodnote 0.1 percenta. Relevantné grafy 7 a 8 a tabuľky 7 a 8.

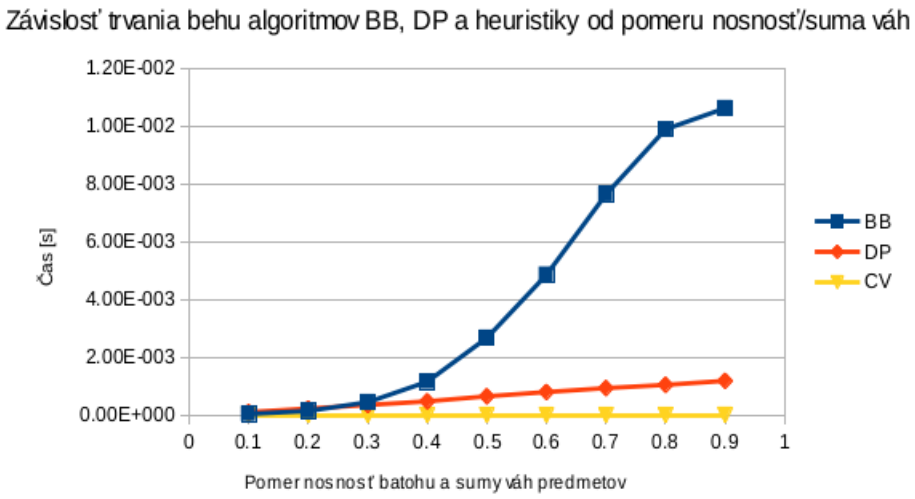

Figure 5: Závislosť trvania behu algoritmov BB, DP a heuristiky od pomeru nosnosť vs. suma váh

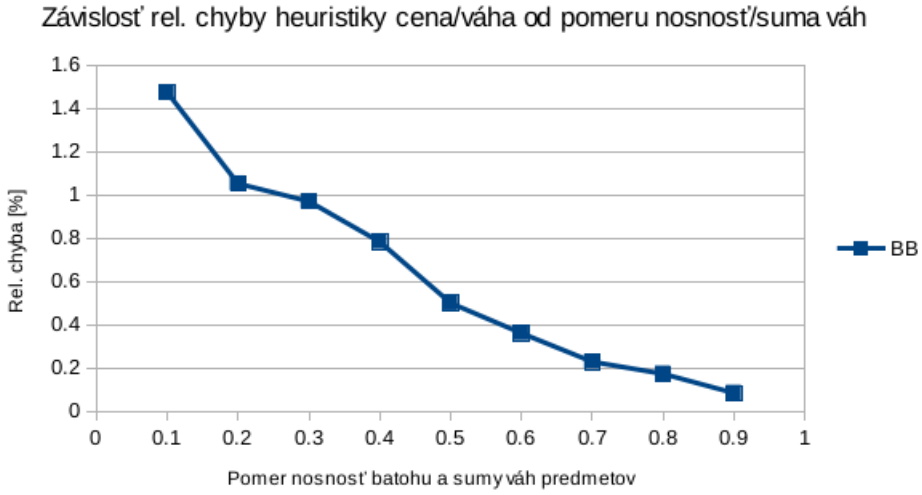

Figure 6: Závislosť rel. chyby heuristiky cena/váha od pomeru nosnosť/suma váh

6 Zhodnotenie

	BB	**DP**	**CV**
exp=0.1	0.002 830	0.000 755	0.000 005 560
exp=0.2	0.002 815	0.000 770	0.000 005 580
exp=0.4	0.002 304	0.000 828	0.000 005 350
exp=0.6	0.001 931	0.000 939	0.000 005 480
exp=0.8	0.001 636	0.001 026	0.000 005 460
exp=1	0.001 269	0.001 076	0.000 005 380
exp=1.5	0.000 972	0.001 185	0.000 005 440
exp=2	0.000 800	0.001 259	0.000 005 600
exp=2.5	0.000 785	0.001 321	0.000 005 360
exp=3	0.000 754	0.001 301	0.000 005 550
exp=4	0.000 665	0.001 197	0.000 004 760

Table 7: Závislosť trvania behu algoritmov BB,DP a heuristiky od exponentu granularity [s]

	CV
exp=0.1	0.57106
exp=0.2	0.55837
exp=0.4	0.56664
exp=0.6	0.53169
exp=0.8	0.61671
exp=1	0.53568
exp=1.5	0.45679
exp=2	0.45407
exp=2.5	0.14103
exp=3	0.29805
exp=4	0.12878

Table 8: Závislosť relatívnej chyby heuristiky cena/váha od exponentu granularity [%]

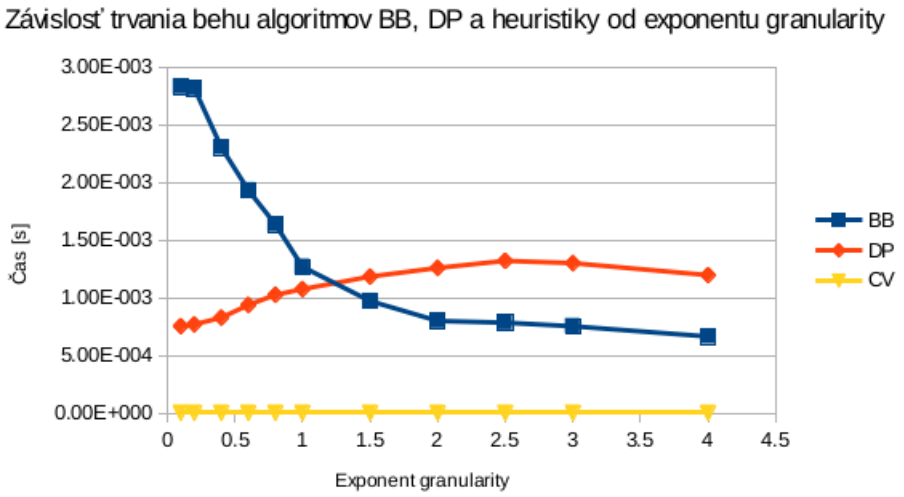

Figure 7: Závislosť trvania behu algoritmov BB, DP a heuristiky od exponentu granularity

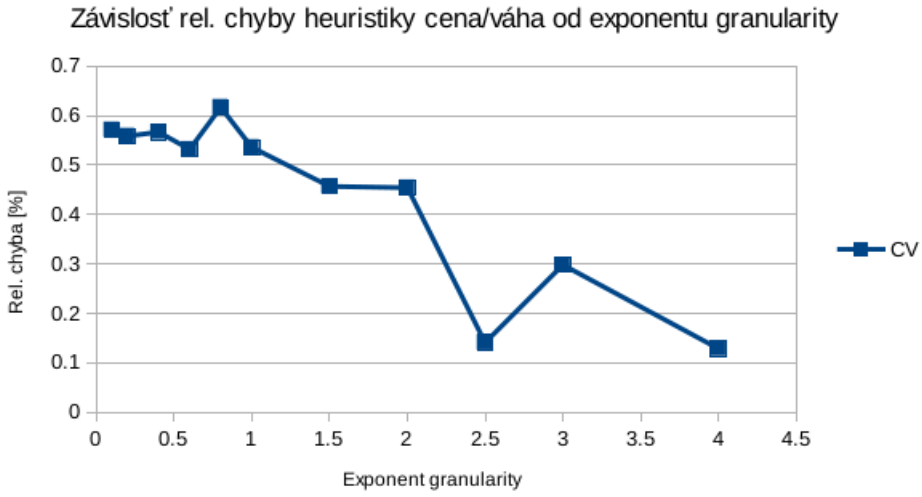

Figure 8: Závislosť rel. chyby cena/váha od exponentu granularity

Algoritmus	Presnosť	Rýchlosť	Hodnotenie
Brute force	absolútna	pomalá (exponenciálna)	poskytuje exaktné, referenčné riešenie, nepoužiteľné pre väčšie inštancie
Dynamické programovanie (dekompozícia podľa kapacity)	absolútna	rýchla (pseudopolynomálna)	pamäťovo náročná, závislá na hmotnosti predmetov
Branch and bound	absolútna	rýchla, ale závislá na charaktere vstupných dát (v najhoršom prípade rovnaká ako brute force)	efektívna pokiaľ kapacita batohu výrazne menšia ako suma váh predmetov
Heuristika cena-váha	vysoká (chyba maximálne v rámci jednotiek percent)	rýchla, nezávislá od charakteru vstupných dát	efektívna, ak sa do batohu zmestí veľké percento predmetov

Table 9: Porovnanie implementovaných algoritmov