

MI-PAA

úkol č.3

Řešení problému batohu dynamickým programováním, metodou větví a hranic a aproximativním algoritmem

Zadání

Naprogramujte řešení problému batohu:

1. metodou větví a hranic (B&B) tak, aby omezujícím faktorem byla hodnota optimalizačního kritéria.
2. metodou dynamického programování
3. modifikujte tento program tak, aby pracoval s omezenou přesností zobrazení vah nebo cen - aproximativní algoritmus.

Řešení

Branch and Bound

K původní implementaci vytvořené k úkolu č.1 byly přidány omezující podmínky – průběžně bylo kontrolováno, zda již neexistuje lepší řešení, či zda jistě není možné lepšího řešení dosáhnout.

Dynamické programování

Zvolil jsem metodu dekompozice dle váhy, pro měření byla každá instance měřena vícekrát po sobě, jinak by nebylo možné naměřit hodnoty větší než chyba měření.

Dynamické programování s omezenou přesností

Původní řešení bylo rozšířeno o možnost omezení uvažovaných bitů – bity byly odštěhávány od nejmenšího, pro snadnou implementaci to fakticky znamenalo vydělení libovolným číslem a zaokrouhlení dolů.

Implementace

Pro implementaci byl použit jazyk C++ na platformě GNU/Linux. Údaje byly měřeny na desktopové stanici s dostatečným množstvím DDR2 1066MHz paměti, procesorem je Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz.

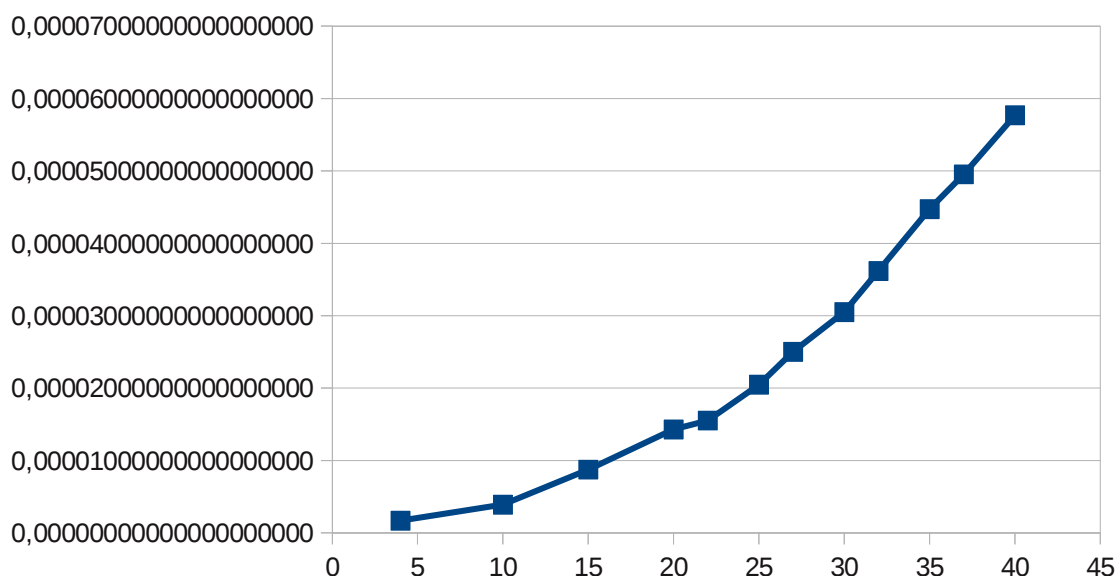
Naměřené výsledky

velikost instance	prumerny cas brute force [s]	prumerny cas dynamicky [s]	prumerny cas b&b [s]
4	0,00000030754505157474	0,00000169930076599000	0,00000033548011780000
10	0,00001055717470000000	0,00000389735155106000	0,00000733083944321000
15	0,00033995151522000000	0,00000874921636581000	0,00023622423791880000
20	0,01077801227566000000	0,00001428734745979000	0,01078719139100000000
22	0,04590893268584000000	0,00001551963763237000	0,04158598423000000000
25	0,36850724697110000000	0,00002046244339943000	0,33582431793200000000
27	1,38243263244640000000	0,00002501946840286000	1,42278566837300000000
30	11,15385844707490000000	0,00003051051964760000	11,62431035518600000000
32	44,30237143516530000000	0,00003618223886490000	47,57183345794700000000
35	359,73628925800300000000	0,00004473661155701000	
37		0,00004953073616028000	
40		0,00005767884211540000	

Zde nejsou výsledky měření s omezenou přesností, budou v samostatné části.

V rámci metody Branch and Bound navzdory očekávání nedošlo v podstatě k žádnému zrychlení, což si nedokážu přesvědčivě vysvětlit. Jedním ze způsobů, jak mohlo dojít ke zrychlení, je průběžné počítání některých hodnot, jako aktuální váha a cena batohu; což si však nemyslím, že spadá přímo do této metody, neboť to považuji za vlastnosti batohu, které není třeba pokaždé znovu přepočítávat. Díky tomuto jsem měl již původní řešení hrubou silou částečně zrychlené, a je tedy možné, že díky tomuto k žádnému velkému zrychlení na daných instancích nedošlo. Je však možné, že to bylo opravdu způsobeno těmito instancemi, a na jiných by výsledky vypadaly jinak.

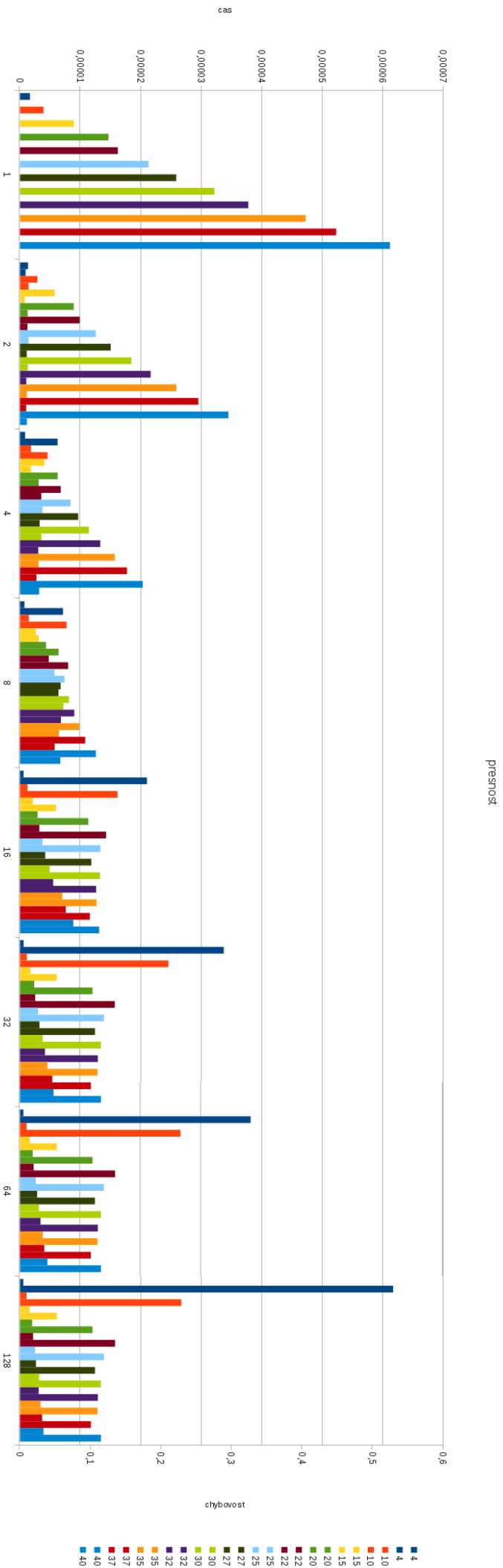
Dynamický výpočet má cca lineární tendenci, nicméně u tohoto algoritmu nezáleží pouze na počtu dostupných prvků, ale také na parametru, kterým je v mém případě maximální kapacita batohu. V grafu je vidět průměrný čas trvání výpočtu v závislosti na instanci.



V algoritmu s omezenou přesností jsem postupně umazával zanedbávané bity, a čímž se postupně i zvětšovala chyba proti původní exaktní verzi.

count	precision	time	prumerna chyba	nedelené casy
4	4	1	0,00000164278898239120	0
4	4	2	0,00000131593661308280	0,0078300395
4	4	4	0,00000084014053344760	0,0533255957
4	4	8	0,00000073178110122640	0,0609334697
4	4	16	0,00000058484339714080	0,1799714929
4	4	32	0,00000059403219223140	0,2889277001
4	4	64	0,00000056048331260640	0,3269632235
4	4	128	0,00000053517041206340	0,5291088486
10	10	1	0,00000389120244979820	0
10	10	2	0,00000286389622688320	0,012042215
10	10	4	0,00000181486158371000	0,0391212954
10	10	8	0,00000144856438636800	0,0660289088
10	10	16	0,00000123839702606140	0,1381727328
10	10	32	0,00000113928751945520	0,2103856957
10	10	64	0,00000110102624893240	0,227477249
10	10	128	0,00000107475075721800	0,2287157427
15	15	1	0,00000889426980018540	0
15	15	2	0,00000571086144447340	0,006540472
15	15	4	0,00000404693388938900	0,0154809032
15	15	8	0,00000262049541473360	0,0264599391
15	15	16	0,00000215198740959220	0,0509780924
15	15	32	0,0000017177720451400	0,0522383094
15	15	64	0,00000160238251686040	0,0522383094
15	15	128	0,00000157330603599580	0,0522383094
20	20	1	0,00001462203221321120	0
20	20	2	0,00000888813977241540	0,0108539843
20	20	4	0,00000622496476173400	0,0264074215
20	20	8	0,00000429881134033140	0,054584771
20	20	16	0,00000290376834869400	0,0968567048
20	20	32	0,00000235144944190980	0,1028218704
20	20	64	0,00000210418372154240	0,1028218704
20	20	128	0,00000200661787986760	0,1028218704
22	22	1	0,00001618223052024840	0
22	22	2	0,00000987675929069560	0,0104876727
22	22	4	0,00000673394284248400	0,0302332962
22	22	8	0,00000474742674827600	0,0682161178
22	22	16	0,00000319356060028080	0,1220092572
22	22	32	0,00000250734581947280	0,134373108
22	22	64	0,00000225290865898060	0,1348253794
22	22	128	0,00000215349159240760	0,1348253794
25	25	1	0,00002126188888549820	0
25	25	2	0,00001253543276786820	0,0121493349
25	25	4	0,00000835348801612800	0,031699174
25	25	8	0,00000568569045066840	0,0631235171
25	25	16	0,00000371185841560400	0,1139118986
25	25	32	0,00000297811322212220	0,1189622286
25	25	64	0,00000261051368713340	0,1189622286
25	25	128	0,00000247110614776640	0,1189622286

count	precision	time	prumerna chyba	nedelene casy
27	1	0,00002583705029487640	0	2,5837050295
27	2	0,00001500319657325740	0,009482923	1,5003196573
27	4	0,00000961569433212340	0,0278482187	0,9615694332
27	8	0,00000672151608467100	0,0545530298	0,6721516085
27	16	0,00000416563501358040	0,1010779823	0,4165635014
27	32	0,00000321213517189020	0,1062820999	0,3212135172
27	64	0,00000284049987792940	0,1062820999	0,2840499878
27	128	0,00000264427165985120	0,1062820999	0,264427166
30	1	0,00003215476498603840	0	3,2154764986
30	2	0,00001840767879486080	0,0109408114	1,8407678795
30	4	0,00001138560299873420	0,0303150508	1,1385602999
30	8	0,00000809022889137300	0,0614804255	0,8090228891
30	16	0,00000489026923179600	0,1132531245	0,4890269232
30	32	0,00000373382616043020	0,1146568828	0,373382616
30	64	0,00000313273329734800	0,1146568828	0,3132733297
30	128	0,00000314901094436660	0,1146568828	0,3149010944
32	1	0,00003776525402069080	0	3,7765254021
32	2	0,00002160883460044840	0,0088582227	2,16088346
32	4	0,00001327495269775420	0,025888061	1,3274952698
32	8	0,00000897296233177240	0,0579520746	0,8972962332
32	16	0,00000547991104125940	0,1078736997	0,5479911041
32	32	0,00000412321329116840	0,1103701758	0,4123213291
32	64	0,00000339742617607060	0,1103701758	0,3397426176
32	128	0,00000308252649307240	0,1103701758	0,3082526493
35	1	0,00004724961233139020	0	4,7249612331
35	2	0,00002587586355209460	0,009695244	2,5875863552
35	4	0,00001568081383705120	0,0263077421	1,5680813837
35	8	0,00000980606150627180	0,0552920493	0,9806061506
35	16	0,00000698938708305340	0,108489424	0,6989387083
35	32	0,00000454312257766740	0,1098944704	0,4543122578
35	64	0,00000376304054260260	0,1098944704	0,3763040543
35	128	0,00000338954682350140	0,1098944704	0,3389546824
37	1	0,00005230215797424380	0	5,2302157974
37	2	0,00002950778141021660	0,00872166	2,950778141
37	4	0,00001769937644004840	0,0232483483	1,769937644
37	8	0,00001076752953529340	0,0490042668	1,0767529535
37	16	0,00000757979412078840	0,0989897993	0,7579794121
37	32	0,00000532808504104560	0,1004940537	0,5328085041
37	64	0,00000401665816307040	0,1004940537	0,4016658163
37	128	0,00000366042261123640	0,1004940537	0,3660422611
40	1	0,00006120403013229300	0	6,1204030132
40	2	0,00003447230629920980	0,0097192713	3,4472306299
40	4	0,00002031943640708920	0,0272026838	2,0319436407
40	8	0,00001254349904060320	0,0570666368	1,2543499041
40	16	0,00000881899390220620	0,112136805	0,8818993902
40	32	0,00000555189194679300	0,1147759333	0,5551891947
40	64	0,00000455152325630180	0,1147759333	0,4551523256
40	128	0,00000388875446319600	0,1147759333	0,3888754463



V grafu jsou pro každý typ instance dva sloupce, ten vlevo nám zobrazuje čas, který postupně roste pro rostoucí počet prvků v instanci, leč na druhou stranu s klesající přesností podle očekávání klesá. Průměrná chybovost však samozřejmě s rostoucím počtem zanedbaných částí postupně roste.

Lepší graf se mi pro výsledky nepodařilo vymyslet, a ani jak jej vytvořit, aby byl co nejlépe čitelný.

Závěry

Pro dané instance si myslím že byla nejvhodnější metoda dynamického programování, bez jakýchkoli heuristik či zobecnění/zanedbání, nicméně však s tou podmínkou, že nebyla uvažována spotřeba paměti, které bylo dostatečné množství.

V případě omezené paměti bych nejspíše volil zanedbávání bitů, což jsem již dříve mimo tento předmět využil, v rámci počítání batohu s reálnými a nikoli celými čísly.

Branch and Bounds považuji za základní a zásadní metodu, která se tedy v mém případě nijak neprojevila, nicméně počítat cokoli hrubou silou bez alespoň minimální aplikace tohoto, považuji za skutečně zbytečný výpočet.