Domácí úloha č. 4 – Knapsack problém III.

Specifikace úlohy

Cílem této domácí úlohy bylo experimentálně vyhodnotit citlivost metod řešení problému batohu na parametry instancí generovaných generátorem náhodných instancí. Na základě tohoto měření je nutno provést experimentální vyhodnocení kvality řešení a výpočetní náročnosti. Zejména závislosti výpočetního času a relativní chyby na maximální váze, maximální ceně věci, poměru kapacity batohu k sumární váze a granularitě.

Nástroje k řešení

K implementaci jsem využil programovací jazyk **Java** pod prostředím **NetBeans**. Všechny výpočty běželi na procesoru Intel Core 2 Duo 3.00 GHz a pod operačním systémem Microsoft Windows 7. Výsledky byly zpracovány tabulkovým procesorem Microsoft Excel.

Výsledný zdrojový kód je spouštěn ze souboru **Main.java**, zbytek kódu je přehledně rozdělen do tříd.

K měření času jsem použil funkci **System.currentTimeMillis().**

Generování instancí

Generování probíhalo generátorem náhodných instancí. Generátor je rozsáhle parametrizován a umožňuje nám měnit tyto parametry:

Parametr	Povinný	Typ hodnoty	Význam
`-I`	ne	celé číslo	Počáteční ID
`-n`	ano	celé číslo	počet věcí
`-N`	ano	celé číslo	počet instancí
`-m`	ano	reálné číslo	poměr kapacity batohu k sumární váze
`-W`	ano	celé číslo	max. váha věci
`-C`	ano	celé číslo	max. cena věci
`-k`	ano	reálné číslo	exponent *k*
`-d`	ano	-1, 0, 1	-1..více malých věcí, 1..více velkých věcí, 0..rovnováha

Měření instancí

Měření jednotlivých závislostí provádíme tak, že všechny parametry zafixujeme a jeden z nich měníme, proto je vhodné si najít nějaké výchozí ohodnocení parametrů, které jsem zvolil přibližně v polovině a to takto:

```
gen -n 20 -N 50 -m 0.5 -k 0 -d 0 -W 100 -C 1000
```

Při provádění jsou tedy všechny parametry přibližně uprostřed a jeden z nich měníme nejdříve na jednu stranu a poté na druhou stranu, v rámci možností.

Abychom eliminovali vliv použitého hardware, měříme počet stavů, které jsme pro danou instanci prošli.

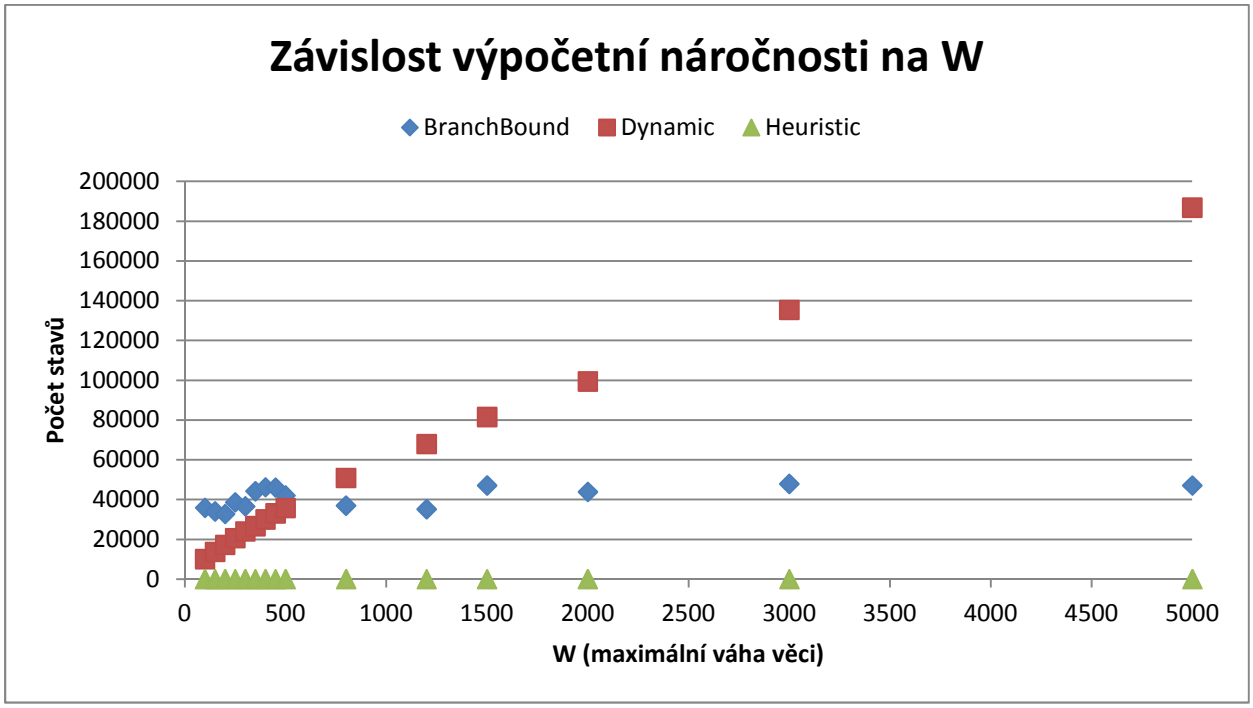
Pro měření byly použity algoritmy, které jsem programoval v minulých úlohách, akorát bylo doimplementováno počítání průměrného počtu stavů stavového prostoru.

Závislost výpočetního času a relativní chyby na maximální váze věci (W)

Pro tuto závislost jsem měnil maximální váhu věci od 100 do 5000. Původní rozsah jsem stanovil 100-1000, ale nakonec jsem provedl ještě pár měření až do 5000, pro ověření, že závislost je postupná (Dynamické postupně roste, BB a Heuristika pořád stejná). Pro tuto závislost jsem naměřil tyto hodnoty:

	BB		Dynamic		Heuristika			
W	čas	# stavů	čas	# stavů	čas	# stavů	rel.odchylka	max. odchylka
100	1878	35875	440	10051	2	19	0,00580309	0,027964785
150	1729	34060	567	13698	2	19	0,009117735	0,065474885
200	1708	32802	697	17235	2	20	0,008278062	0,038149246
250	1961	38665	821	20648	2	20	0,008428202	0,056096482
300	1861	36595	977	24027	2	20	0,005181872	0,03850239
350	2242	44246	1085	26629	3	20	0,00637347	0,038029503
400	2349	46226	1276	30048	2	20	0,005944368	0,042600313
450	2326	46048	1409	33061	2	20	0,007165684	0,051793835
500	2139	42063	1499	35771	2	20	0,005704472	0,0304805
800	1898	36967	2144	50917	2	20	0,007376677	0,053807249
1200	1788	35193	2773	67931	2	20	0,006290272	0,043466224
1500	2347	47088	3318	81519	3	20	0,008173416	0,067747539
2000	2215	43878	4047	99446	3	20	0,007235783	0,041516245
3000	2420	47899	5631	135394	3	20	0,006418295	0,03901951
5000	2413	47083	7974	186795	3	20	0,006437388	0,050721034

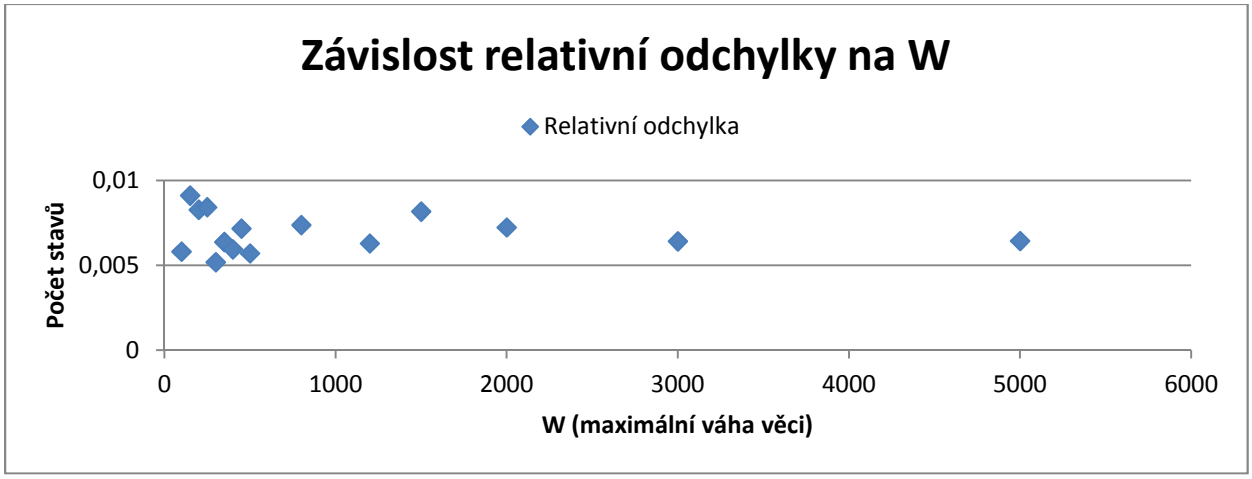
Grafické znázornění závislosti výpočetního času na W pro naměřená data:



Na grafu je vidět, že heuristika i Branch&Bounds mají pořád stejnou náročnost. Změna váhy se ale negativně projeví na Dynamickém algoritmu, protože je implementován dekompozicí podle kapacity batohu. Když zvýšíme maximální váhu věci, zvýší se i maximální kapacita batohu, takže pomyslná tabulka možných stavů se zvětší, resp. se zvětší počet řádků.

Ostatní algoritmy jsou na váze věci nezávislé, protože pro ně váha věci není důležitá, pouze věc do batohu dají a kontrolují, jestli se vejde, nebo ne. Není pro ně důležité, jestli věc váží 50 anebo 1000.

Grafické znázornění relativní odchylky na změně maximální váhy věci:



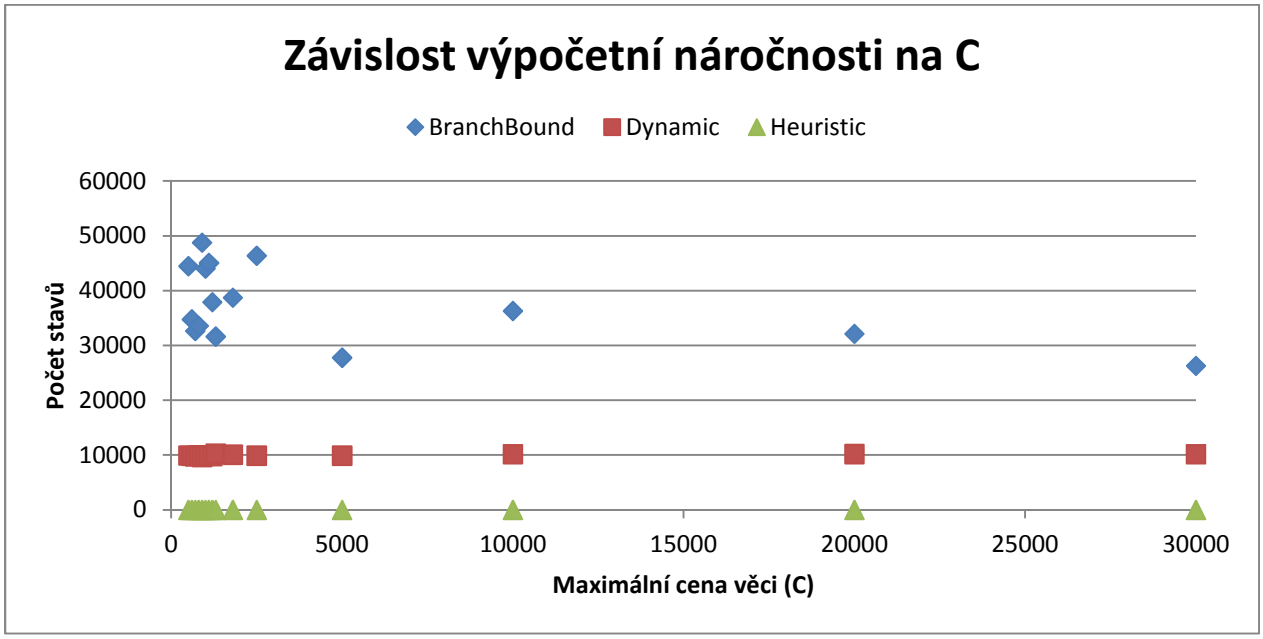
V přiloženém souboru jsem naměřil i maximální odchylku. Z grafu je vidět, že relativní odchylka heuristiky je prakticky pořád stejná. Stejně jako u měření výše je algoritmu jedno, jestli má věc váhu 20 anebo 5000, prostě jí do batohu přidá, nebo ne.

Závislost výpočetního času a relativní chyby na maximální ceně věci (C)

Pro měření této závislosti jsem měnil maximální cenu věci od 500 do 1500, nakonec jsem ale naměřil i pár měření až do 30000, abych se podíval na přibližný vývoj grafu.

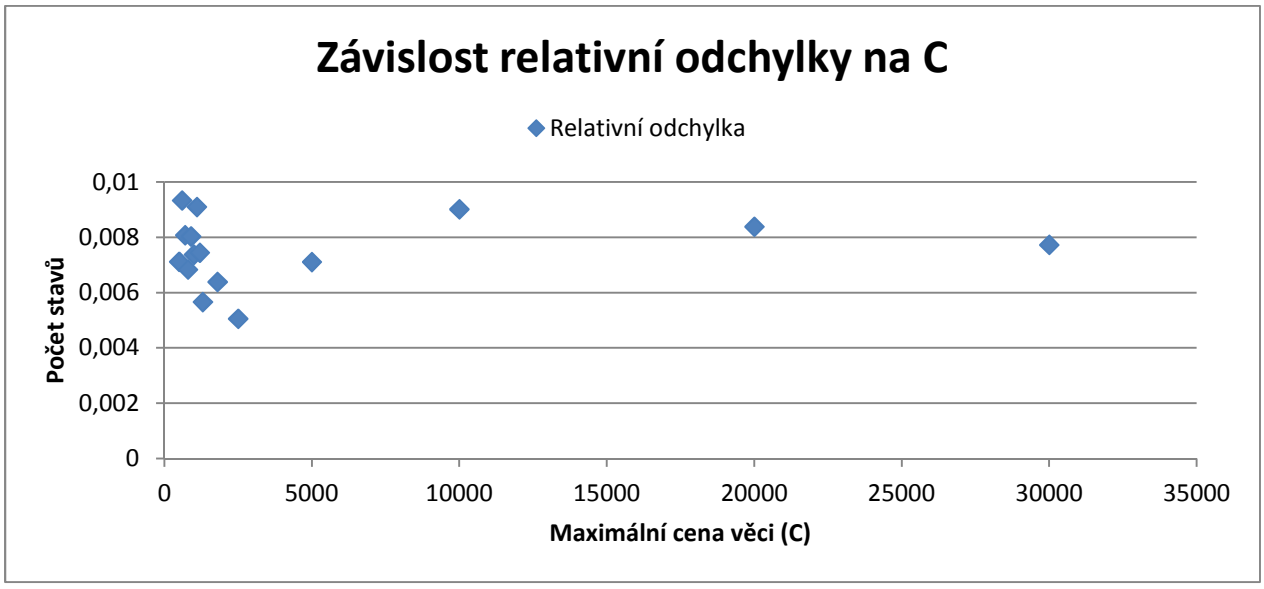
	BB		Dynamic		Heuristika			
C	čas	# stavů	čas	# stavů	čas	# stavů	rel.odchylka	max. odchylka
500	2245	44489	422	9935	3	19	0,007123963	0,037123648
600	1781	34773	418	9870	3	20	0,009339916	0,056655434
700	1675	32652	417	9778	2	19	0,008084214	0,045123468
800	1729	33599	424	9979	3	19	0,006837685	0,036195401
900	2436	48779	406	9653	3	19	0,008038724	0,032385615
1000	2228	44044	415	9875	3	19	0,007379978	0,03847491
1100	2249	45067	426	9969	2	19	0,009106391	0,047954703
1200	1899	37917	413	9829	3	19	0,007451095	0,045183191
1300	1614	31633	432	10263	3	19	0,005665787	0,061630435
1800	1954	38730	432	10092	2	19	0,006392088	0,0461549
2500	2334	46377	424	9907	2	19	0,005058998	0,040064103
5000	1426	27785	424	9911	2	19	0,007114	0,044508386
10000	1858	36304	440	10182	3	19	0,009024723	0,054201424
20000	1724	32139	449	10218	2	19	0,008390182	0,051048946
30000	1384	26293	436	10184	2	19	0,007733703	0,04935237

Grafické znázornění výpočetního času na maximální ceně věci (C):



Z grafu je vidět, že ani jeden z algoritmů není závislý na maximální ceně věci. Důvody pro Branch&Bound a Heuristiku jsem popsal již výše – algoritmu je jedno, jakou má věc cenu, prostě jí přidá, nebo ne, dle její váhy. V tomto případě je to jedno i Dynamickému programu, protože je implementován dekompozicí podle kapacity batohu. Pokud by byl implementován dekompozicí podle ceny, myslím si, že by byl na změně C závislý a průběh by byl podobný, jako jsem ho měl já u měření závislosti na W.

Grafické znázornění relativní odchylky na C:



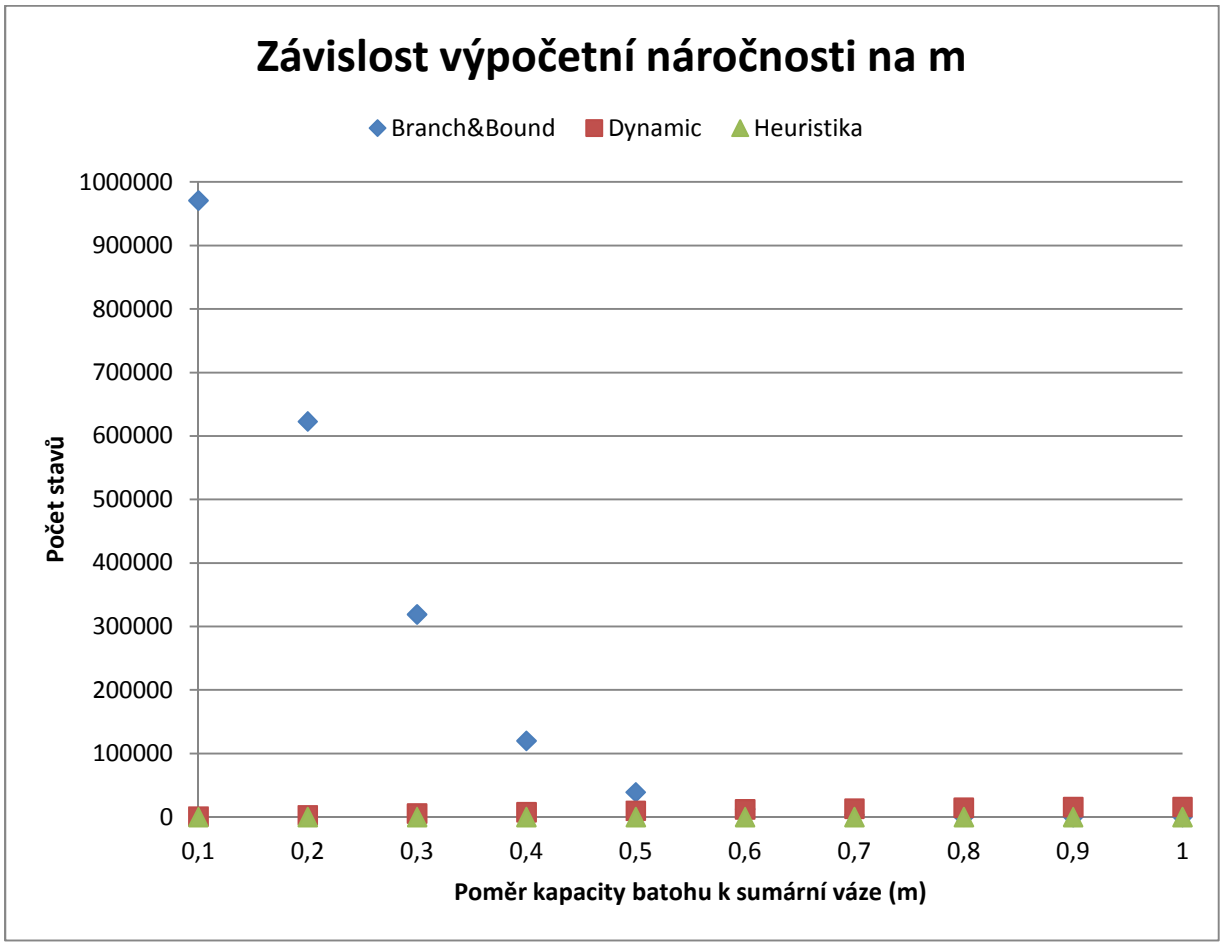
Stejně jako u závislosti na maximální váze se relativní chyba v závislosti na maximální ceně nemění. Algoritmu je jedno, jakou má věc váhu, prostě jí přidá, nebo ne.

Závislost výpočetního času a relativní chyby na poměru kapacity batohu k sumární váze (m)

U tohoto měření měním poměr kapacity batohu k sumární váze od 0,1 až do 1. Nejmenší hodnota znamená, že ve výsledném řešení nebude skoro žádná věc, číslo 1 znamená, že tam budou všechny. Toto měření má asi „nejhezčí“ výsledky, je zde patrná závislost pro všechny algoritmy i pro relativní chybu.

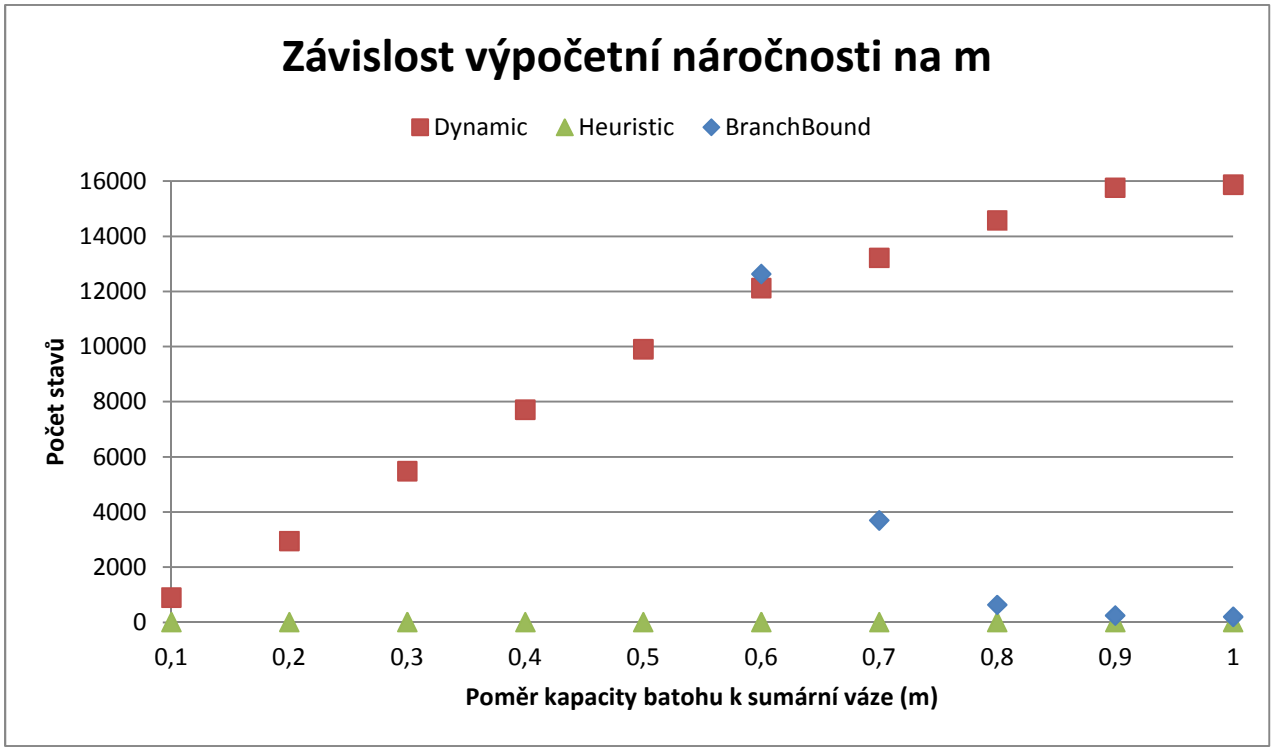
	BB		Dynamic		Heuristika			
m	čas	# stavů	čas	# stavů	čas	# stavů	rel.odchylka	max. odchylka
0,1	53393	970902	34	906	2	18	0,019318153	0,126269465
0,2	34975	622955	107	2956	2	19	0,011105516	0,073728814
0,3	17609	319062	210	5491	3	19	0,006146386	0,046864326
0,4	6481	120097	314	7718	2	19	0,007281967	0,031965443
0,5	1985	39190	422	9916	3	19	0,007442138	0,057133871
0,6	616	12641	525	12131	2	19	0,006431875	0,030576382
0,7	180	3707	545	13226	2	19	0,005767405	0,042191839
0,8	41	643	569	14584	2	19	0,001875062	0,018925952
0,9	20	255	573	15771	3	19	0,001608525	0,012456024
1	19	211	571	15880	2	20	0	0

Grafické znázornění závislosti výpočetního času na m:



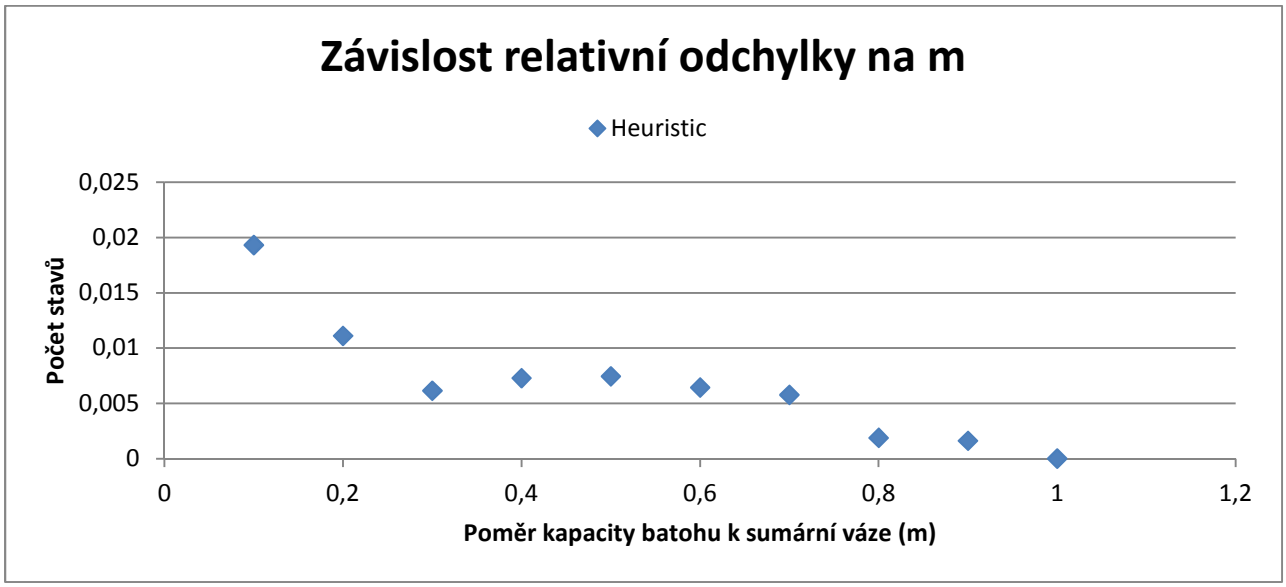
Z grafu je vidět, že nejvíce závislý je algoritmus Branch&Bound, který expanduje nejvíce stavů, pokud nastavíme, že v řešení bude jenom pár věcí. Graf nám to tak trochu kazí, takže vygenerujeme ještě jeden graf tak, aby byl vidět průběh Dynamického algoritmu a Heuristiky.

Grafické znázornění s detailem na Dynamický algoritmus:



Na grafu je vidět, že i Dynamický algoritmus je závislý na změně parametru *m*, ale velice málo, což je závislé spíše na implementaci. Heuristika je stále konstantní, ale mění se relativní odchylka viz. graf dále.

Grafické znázornění relativní odchylky na m:



Na grafu je vidět, že se relativní odchylka se zvyšujícím se parametrem m snižuje. Při m=1 je relativní odchylka nulová, protože v batohu budou vždy všechny věci. Pokud je m malé, znamená to, že počet

věcí, které v batohu jsou je malý a tak existuje mnoho kombinací a z toho plynoucí vyšší chyba. Pokud máme m=0.9 víme, že většina věcí v batohu bude, jenom hledáme těch pár věcí, které tam nebudou a možných kombinací je tak méně a menší prostor pro chybu.

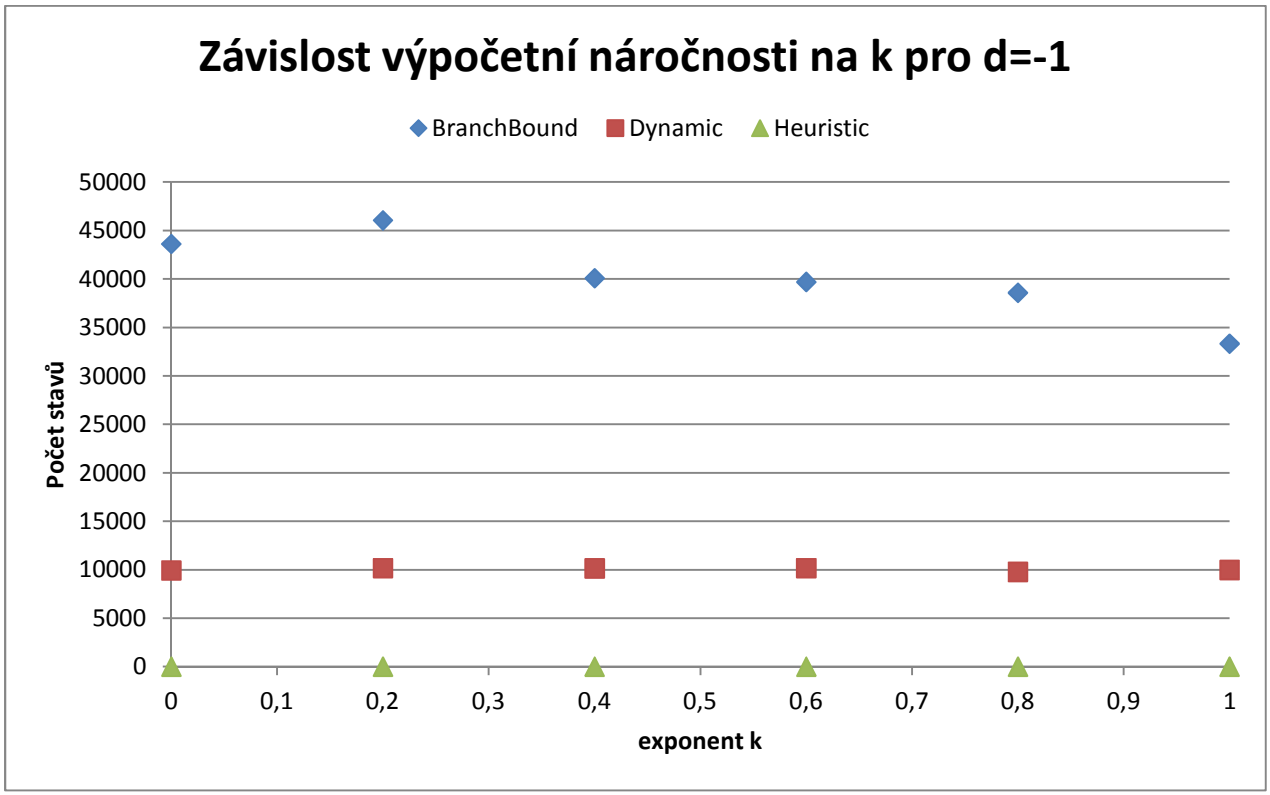
Závislost výpočetního času a relativní chyby na granularitě (d)

V posledním měření budeme měnit granularitu pomocí parametru *d*, který bude z intervalu -1 … 1 a dále pomocí koeficientu *k*, který bude dosazován do vzorečku pro výpočet pravděpodobnosti, že prvek v instanci je, nebo není. Granularita znamená, že pokud parametr *d* nastavíme na -1, budou se generovat spíše malé věci, ale bude jich více. Parametr nastavený na jedničku znamená, že se budou generovat velké věci, ale bude jich méně.

Závislost pro d=-1

	BB		Dynamic		Heuristika			
k	čas	# stavů	čas	# stavů	čas	# stavů	rel.odchylka	max. odchylka
0	1955	37863	431	9956	3	19	0,006709999	0,030319379
0,2	1327	26124	392	9215	2	19	0,004829419	0,048635537
0,4	1318	26915	357	8546	2	19	0,009980418	0,03757621
0,6	1274	26252	317	7530	3	19	0,004977944	0,043765307
0,8	1083	22910	296	7066	2	19	0,006011085	0,037959526
1	832	17861	274	6287	2	19	0,006406506	0,051111111

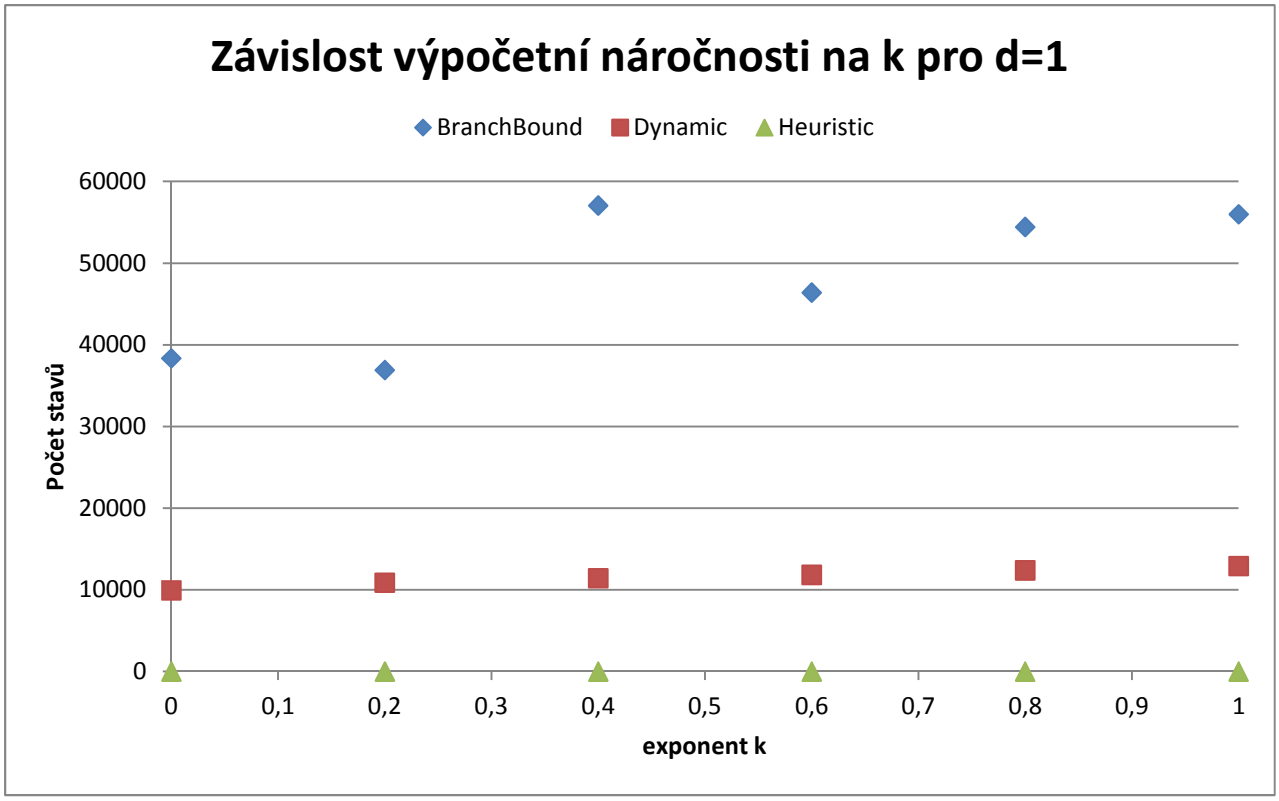
Grafické znázornění pro d=-1



Závislost pro d=1

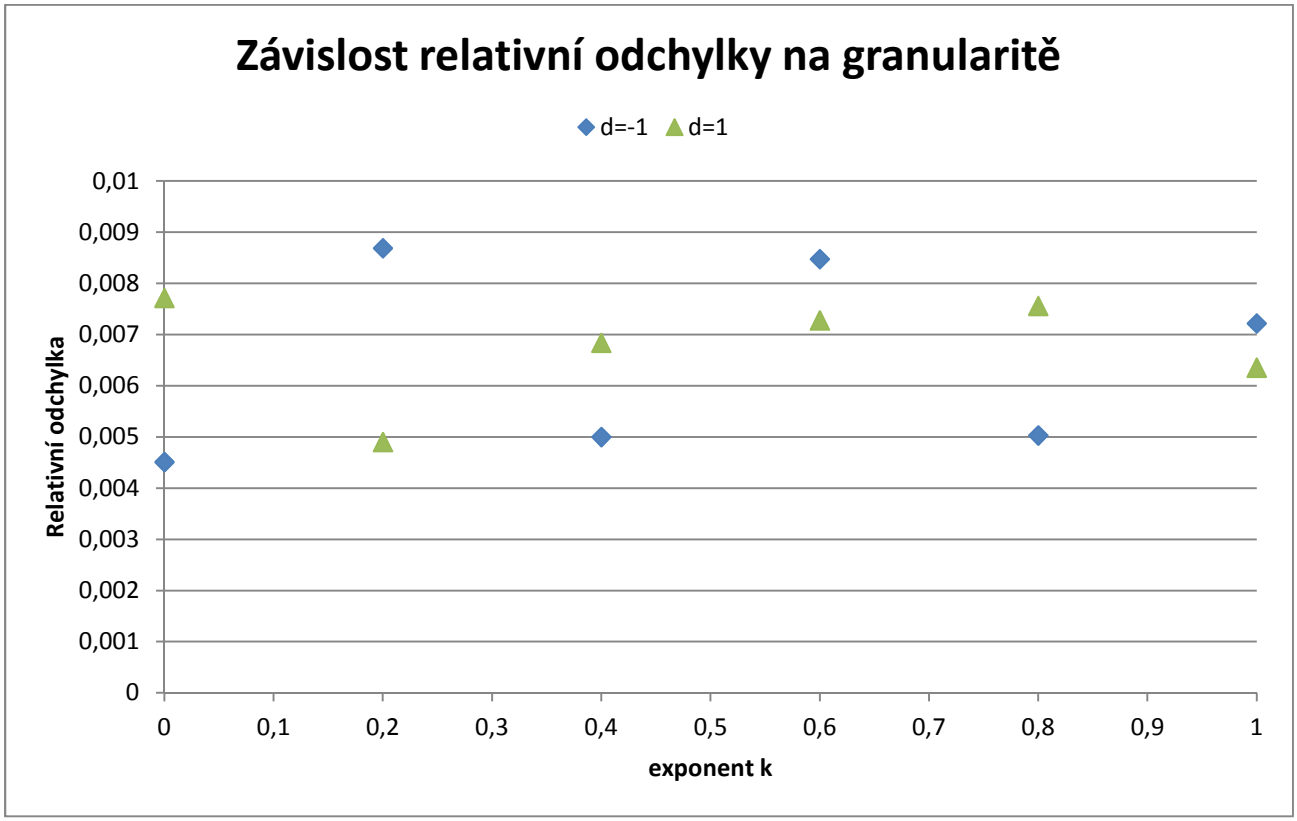
	BB		Dynamic		Heuristika			
k	čas	# stavů	čas	# stavů	čas	# stavů	rel.odchylka	max. odchylka
0	1990	38361	427	9962	3	19	0,007720129	0,047319054
0,2	1935	36925	463	10898	2	19	0,004902678	0,047455074
0,4	2952	57065	489	11443	3	19	0,006840515	0,052806546
0,6	2493	46405	505	11854	3	19	0,007278289	0,048114434
0,8	2901	54438	521	12414	3	19	0,007562278	0,038865697
1	3051	56002	553	12930	3	19	0,006356306	0,033625555

Grafické znázornění pro d=1:



U znázorněného grafu pro změnu parametru d je vidět, že pro Dynamický algoritmus a heuristiku nemá změna granularity vliv. Pro Branch&Bound algoritmus je u d=1 vidět rostoucí tendence a u d=-1 spíše klesající, z čehož by se dalo usuzovat, že algoritmu B&B svědčí spíše malé věci. U všech algoritmů je ale výhodnější větší počet malých věcí, protože je navštíveno daleko méně stavů.

Závislost relativní odchylky na granularitě



Na grafu je vidět, že granularita nemá na heuristiku prakticky žádný vliv.

Závěr

V této úloze jsme si vyzkoušeli, jaký může mít změna některých parametrů vliv na výpočetní náročnost. Znatelné je to hlavně u závislosti výpočetního času a relativní chyby na poměru kapacity batohu k sumární váze, kde je vidět, že do poměru 0.6 je výhodnější použít Dynamický algoritmus, ale dále již Branch&Bounds. Ostatní pozorování jsem okomentoval přímo u daného grafu.