Performance of Selected Non-deterministic Algorithms Solving the Multidimensional Knapsack Problem

Výkonnosť vybraných nedeterministických algoritmov riešiacich multidimenzionálny problém batohu

Ján Pittner

Abstract

The paper deals with the performance comparison of selected non-deterministic algorithms used for solving the multidimensional knapsack problem. After briefly explanation of these algorithms, we compare the results.

Keywords:

knapsack problem, genetic algorithm, extended local branching, ant colony optimization, comparison

Kľúčové slová:

problém batohu, genetický algoritmus, extended local branching, algoritmus mravčej kolónie, porovnanie

1. Úvod

Je zrejmé, že každá časť ľudského života je ovplyvnená rozhodnutiami. Hoci rozhodnutia týkajúce sa súkromného života môžu byť založené na osobných preferenciách alebo emóciách, proces rozhodovania v profesnom prostredí vyžaduje formalizované a nezávislo overené rozhodnutia zúčastnených. Preto je požadované kvantitatívne zloženie všetkých faktorov, ktoré ovplyvňujú rozhodnutia, ako aj výsledok rozhodovacieho procesu.

Za účelom splnenia tohto cieľa musí existovať možnosť reprezentovať efekty rozhodnutia numerickou hodnotou. V najjednoduchších prípadoch je výstup rozhodnutia možné merať jednou hodnotou predstavujúcou prínos, zisk, stratu, náklady alebo inou mernou veličinou. Porovnanie týchto hodnôt navodzuje celkové poradie na množine všetkých možností, ktoré sú prístupné po rozhodnutí. Nájdenie možnosti s najvyššou, alebo najnižšou hodnotou môže byť obtiažne, pretože množina prípustných možností môže byť extrémne veľká, alebo neznáma.

Riešenie takýchto problémov býva často obtiažne a je preto pochopiteľné, že aj niektoré algoritmy majú dlhý exekučný čas. Na najobtiažnejšie úlohy sa preto využívajú heuristické metódy, ktoré sú schopné tento čas výrazne skrátiť, avšak za cenu „len“ približného výsledku. Spomenuté metódy riešenia naznačia, že využitie metód umelej intelgencie je v súčasnosti jedna z najefektívnejších techník heuristiky.

Existuje však množstvo rozličných heuristík, ktoré možno využiť. V článku sme sa rozhodli porovnať tie, ktoré sú v súčasnosti najpoužívanejšie, s technikou, ktorá bola dlhú dobu podpriemerná avšak v poslednej dobe začína byť čoraz populárnejšia a teda genetické algoritmy.

2. Definícia problému

Problém batohu predstavuje úlohu kombinatorickej optimalizácie. Majme danú množinu vecí (tovarov), pričom každá vec má stanovenú váhu a hodnotu. Ďalej majme kontajner (batoh), ktorý má stanovenú nosnosť. Našou úlohou je potom rozhodnúť, ktoré z daných vecí vložíme do batohu a to tak, aby sme maximalizovali cenu vložených vecí a neporušili váhové obmedzenie kontajnera.

Využitie našiel tento problém napríklad pri simulovaní investičných rozhodnutí, vytváraní kryptografického systému, alebo členením a rozdelovaním surovín pre výrobný proces.

Problém batohu môžme formulovať aj ako zápis úlohy celočíselného lineárneho programovania a to nasledovne:

$$\sum_{j=1}^{n} p_j x_j$$

S obmedzením:

$$\sum_{j=1}^{n} v_j x_j \leq c$$

$$x_j \in \{0,1\}, j=1,2\ldots,n$$

Vektor optimálneho riešenia bude $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ a hodnota optimálneho riešenia z^*. Množina X^* predstavuje množinu optimálneho riešenia, t. j. množinu predmetov korešpondujúcich s vektorom optimálneho riešenia.

Multidimenzionálny problém batohu

Existuje mnoho variantov problémov batohu, odlišujúcich sa navzájom rôznymi atribútmi. Preto sa zamerajme na jeden konkrétny variant, a teda na *0/1 multidimenzionálny problém batohu*. NP-ťažký 0/1 multidimenzionálny problém batohu je zovšeobecnením jednoduchého 0/1 problému batohu. Skladá sa z vybratia podmnožiny daných objektov (teda položiek) takým spôsobom, aby sa dosiahla maximalizácia celkovej hodnoty zabalených objektov, pri dodržaní obmedzení batohu.

Formálny zápis tohto typu problému je nasledovný.

$$\sum_{j=1}^{n} c_j x_j$$

so všeobecným zápisom obmedzení

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \forall i=1,2,\ldots,m$$

$$x_j \in \{0,1\}, 1 \leq j \leq n$$

Kde n je počet objektov, m je počet obmedzujúcich dimenzií batohu, c_j predstavuje hodnotu objektu j uloženého v batohu, x_j je binárnou premennou indikujúcou stav objektu (1 – nachádza sa v batohu, 0 – nenachádza sa v batohu), b_j predstavuje hranicu i-tej dimenzie batohu

a napokon a_{ij} predstavuje vstupy z matice obmedzení. Táto matica obsahuje obmedzujúcu hodnotu pre každý objekt v jednotlivých dimenziách[1)], Matica má teda veľkosť NxM.

3. Vybrané nedeterministické riešenia problému batohu

Nedeterministickú metódu, inak nazvanú aj heuristika chápe informatika ako veda ako postup získania riešní problému, ktoré však nie je presné a nemusí byť nájdené v krátkom čase. Slúži však najčastejšie ako metóda rýchlo poskytujúca dostatočné a dosť presné riešenie, ktoré však nemožno obecne dokázať. Najčastejšie použitie heuristických metód nachádzame v prípadoch, kde je použitie deterministických algoritmov časovo neprípustné.

Medzi vybrané heuristické metódy riešiace problém multidimenzionálneho batohu sme zaradili:

- Algoritmus Ant colony optimization
- Genetické algoritmy
- Extended local branching

3.1 Algoritmus Ant colony optimization

Algoritmus mravčej kolónie bol inšpirovaný pozorovaním skutočných kolónií mravcov. Jedným z hlavných princípov je nepriama komunikácia medzi jednotlivými agentami kolónie nazvanými umelé mravce. Táto komunikácia je založená na vytváraní ciest pomocou chemickej substancie nazvanej feromón, ktorú využívajú skutočné mravce taktiež na komunikáciu. Algoritmus mravčej kolónie využíva umelé mravce, ktoré sa správajú ako kooperatívni agenti v matematickom priestore, kde im je umožnené prehľadávanie a zlepšovanie ciest – riešení za účelom nájdenia optimálneho riešenia.

Oproti skutočným mravčím cestám môžu tieto umelé obsahovať omnoho komplexnejšie informácie. Pri konštrukcii prvotného riešenia počítajú agenti (mravce) množinu prístupných krokov a zvolia podľa pravidiel pravdepodobnosti najlepší krok. Tieto pravidlá pravdepodobnosti sú založené na heuristických informáciach a úrovni feromónou na cestičke. Čím vyššia úroveň feromónu a dostupnosť heuristických informácií sa na určitej ceste nachádza, tým je viac užitočné danú cestu zvoliť a pokračovať pri prehľadávaní. Vytváranie riešenia agentami je vedené cestou feromónu a heuristickými informáciami, ktoré sú špecifické podľa riešeného problému. Algoritmus mravčej kolónie môže byť použitý na ľubovolný kombinatorický optimalizačný problém definovaním komponentov riešenia, ktoré agenti využívajú na iteratívnu konštrukciu kandidátskych riešení a na ktoré môžu uložiť svoje feromóny. Parciálne riešenia sú znázorňované ako stavy; každý agent sa presúva zo stavu *i* do stavu *j* pričom prispieva k tvorbe zlepšeného parciálneho riešenia.

V každom kroku každý agent *k* počíta množinu prípustných krokov a vykoná pohyb jedným zo smerov založený na rozdelení pravdeopodobnosti.

Pre agenta *k* pravdepodobnosť p_{ij}^{k} pohybu zo stavu *i* do stavu *j* záleží na kombinácii dvoch hodnôt: (Fidanova)

- Atraktivitu η_{ij} kroku vypočítanú heuristikou
- Úrovňou feromónu na ceste τ_{ij}

1) Podľa druhov dimenzií to môže byť napríklad váha, veľkosť, objem, cena atď.

Pravdepodobnosť $p_{ij}^{k}(t)$ výberu kroku *j* ako ďalšieho stavu je daná rovnicou:

$$p_{ij}^{k}(t)=\left\{\begin{array}{c}\dfrac{\tau_{ij}\eta_{ij}\left(S_k(t)\right)}{\sum_{q\in povolené_k(t)}\tau_{iq}\eta_{iq}\left(S_k(t)\right)}\ ak\ j\in povolené_k(t)\\ 0\ ak\ j\notin povolené_k(t)\end{array}\right\}$$

kde

τ_{ij} je úrovňou feromónu na hrane zodpovedajúcej pohybu z bodu *i* do bodu *j*

$\eta_{ij}\left(S_k(t)\right)$ je heuristikou

t je množinou všetkých stavov

$povolené_k(t)$ je množina zostávajúcich prístupných stavov

Platí teda, že čím vyššia hodnota τ_{ij} a $\eta_{ij}\left(S_k(t)\right)$, tým je rentabilnejšie zahrnúť stav *j* do parciálneho riešenia. Pri riešení multidimenzionálneho problému batohu sa využíva variant optimalizácie mravčej kolónie nazvaný systém mravčej kolónie.

Cesty feromónov sú aktualizované v dvoch krokoch. V prvom kroku každý z agentov aplikuje nasledovné aktualizačné pravidlo:

$$\tau_{i\tau(1-\rho)\tau_{ij}}+\rho\tau_0$$

kde ρ spĺňa podmienku $0<\rho\leq 1$ a predstavuje parameter úpadku feromónu, čiže predstavuje vyparovanie sa feromónu, a τ_0 predstavuje počiatočnú úroveň feromónu. Výsledkom pravidla lokálnej aktualizácie je znižovanie úrovne feromónu na hranách, ktoré boli už zvolené agentom, takže sa stávajú menej atraktívne. Toto pravidlo predstavuje druh diverzifikácie v prehľadávaní priestoru riešení.

Po ukončení cesty všetkými agentmi je na hranách tvoriacich najlepšie riešenie zmenená hodnota feromónu podľa pravidla:

$$\tau_{i\tau(1-\alpha)\tau_{ij}}+\alpha\Delta\tau$$

kde

$\Delta\tau$ predstavuje hodnotu funkcie užitočnosti doteraz najlepšieho riešenia

α, ktorá spĺňa podmienku $0<\alpha\leq 1$, je parameter úpadku feromónu

3.2 Genetické algoritmy

Genetické algoritmy sú metódy používané na riešenie mnohých, aj optimalizačných problémov. Názov a myšlienka je úzko spätá s Darwinovou teóriou evolúcie, kde sa počas mnohých generácií prírodná populácia vyvíja podľa princípov prirodzeného výberu a zákona prežitia najschopnejšieho. Týmto spôsobom je potom genetický algoritmus schopný vygenerovať riešenia pre rôzne problémy.

Genetické algoritmy uskutočnujú stochastické globálne prehľadávanie priestoru riešení. Prehľadávanie sa uskutočňuje výmenou informácií medzi chromozómami a zavádzaním nových informácií. Základné genetické operátory sú:

- rozmnožovanie
- mutácia
- prekríženie

V tomto článku využijeme na ukážku algoritmus navrhnutý Beasleym a Chunom, ktorý hoci nepatrí medzi najaktuálnejšie, zachytáva také riešenie daného problému, ktoré slúži ako podklad pre mnohé súčasné vedecké práce. Algoritmus využíva genetický operátor, ktorý funguje podobne ako transformácia v algoritme tabu-search. A to tak, že sa skladá z dvoch fáz, ADD a DROP. Úlohou DROP fázy je modifikácia neprípustného riešenia na prípustné, prostredníctvom prechádzania už uložených vecí v kontajneri a porovnávaním ich pseudo-užitočnosti. Vec s najmenšou pseudo užitočnostou je vylúčená z riešenia, čo tým pádom transformuje riešenie na prípustné. Nasleduje spustenie fázy ADD, ktorá má za úlohu zlepšiť prípustné riešenie dosiahnuté po vykonaní DROP fázy a to tým, že do kontajnera pridá vec s najvyššou pseudo užitočnosťou, ktorá zároveň neporuší žiadne z obmedzení.

Základná myšlienka tohoto genetického algoritmu má potom tvar[2)]:

1: nastav $t := 0$;

2: inicializuj $P(t) := \{S_1, \ldots, S_N\}, S_i \in \{0,1\}^n$;

3: ohodnoť $P(t) := \{f(S_1), \ldots, f(S_N)\}$;

4: nájdi $S^* \in P(t)$ platí, že $f(S^*) \geq f(S), \forall S \in P(t)$

5: **while** $t < t_{max}$ **do**

6: vyber $\{P1; P2\} := \Theta(P(t))$; // Θ predstavuje selekciu prostredníctvom binárneho turnaja

7: prekríž $C := \Omega_C(P1, P2)$; // Ω_C prestavuje uniformný operátor kríženia

8: mutuj $C \leftarrow \Omega_m(C)$; // Ω_m predstavuje mutačný operátor

9: urob C prípustným, $C \leftarrow \Omega_r(C)$; // Ω_r predstavuje repair operátor

10: **if** $\equiv$ lubovolné $S \in P(t)$ **then** // C predstavuje duplikát člena populácie

11: zahoď C a choď na 6;

12: **end if**

13: ohodnoť $f(C)$;

14: nájdi $S' \in P(t) s.t. f(S'') \leq f(S), \forall S \in P(t)$ a nahraď $S' \leftarrow C$

15: **if** $f(c) > f(S^*)$ **then**

16: $S^* \leftarrow C$

17: **end if**

18: $t \leftarrow t + 1$;;

19: **end while**

20: return $S^*, f(S^*)$

Legenda:

t – číslo aktuálnej generácie

P – rodič(parent)

S – riešenie(solution)

C – potomok(child)

f()- fitness funkcia (hodnotenie jednotlivých chromozónov

2) Chu, P., Beasley, J.: A genetic algorithm for the multidimensional knapsack problem. In: Journal of Heuristics Volume 4 Number 1, DOI: 10.1023/A:1009642405419

Tento genetický algoritmus sme modifikovali tak, že sme namiesto opravného operátora (ADD/DROP fáza) implementovali penalizačnú funkciu, ktorá zabezpečí znížený výskyt neprístupných riešení počas behu algoritmu. Naproti tomu sme však zabezpečili lokálnu optimalizáciu potomkov u ktorých je spôsobom pokus/omyl pridávaný dodatočný objekt, pričom objekty sú zoradené podľa ich pseudo užitočnosti. Tým pádom majú potomkovia v rámci možností (niektoré objekty sú fixne stanovené prekrížením) maximalizovanú účelovú funkciu.

Oproti konkurenčným algoritmom boli podstatne zmenené niektoré parametre. Algoritmus pracuje s 100 000 generáciami, pričom veľkosť populácie je 10. Šancu mutácie potom rátame ako 1/počet ukladaných objektov, šancu zamiešania ako 20/počet ukladaných objektov. Elitizmus potom zabezpečí presun najlepšieho jedinca do novo vytvorenej populácie.

3.3 Extended Local Branching

Extended Local Branching predstavuje prehľadávací algoritmus, ktorý pracuje nad algoritmom vetiev a rezov slúžiacich na riešenie zmiešaných celočíselných problémov. Algoritmus vetiev a rezov je hybridným prístupom medzi algoritmom vetiev a hraníc a algoritmom rezných rovín.

Popis princípu samotného algoritmu je výrazne nad rozsah príspevku, preto uvádzame len odkaz na odbornú literatúru.[3)]

Záver

V príspevku porovnávame dosiahnuté výsledky jednotlivých algoritmov. Algoritmy boli testované na sérii úloh z knižnice OR-Library, ktorá je voľne dostupná na internete.

Problém s názvom mknapcb9 obsahuje 29 úloh, ktoré boli dostupné pre konkurenčný algoritmus s názvom Extended Local Branching.

***Tabuľka č. 1** Výstupné hodnoty pre úlohu mknapcb9*[4)]

inštancia	ELB	g/l x200 x3000	percento hodnoty opt. riešenia	g/l x1 x500	percento hodnoty opt. riešenia	hodnota optimalneho riešenia
0	115850	115830	99,32343088	115599	99,12534996	116619,0082
1	114701	114720	99,436483	114320	99,08977281	115370,1303
2	116661	116650	99,40988841	116001	98,85 68064	117342,4514
3	115152	115152	99,3148518	114850	99,05438663	115946,4047
4	116385	116385	99,406994	116100	99,16356922	117079,2872
5	115600	115580	99,31468579	115250	99,03112595	116377,5519
6	113982	113982	99,38298609	113682	99,12141062	114689,6511
7	114190	114220	99,45333907	114150	99,39238886	114847,8282
8	115419	115425	99,58792376	115200	99,39379526	115902,6071
9	116988	116988	99,42145134	116750	99,21918867	117668,7711
10	217995	217985	99,71797031	217320	99,41376382	218601,5212

3) https://www.ads.tuwien.ac.at/publications/bib/pdf/lichtenberger_05.pdf

4) Zdroj tabuľky: Vlastná tabuľka zachytávajúca dosiahnuté výsledky

inštancia	ELB	g/l x200 x3000	percento hodnoty opt. riešenia	g/l x1 x500	percento hodnoty opt. riešenia	hodnota optimalneho riešenia
11	214626	214626	99,79136975	214300	99,63979452	215074,7109
12	215844	215844	99,74257733	215500	99,58361323	216401,0654
13	217827	217827	99,76025553	217650	99,6791932	218350,4832
14	215559	215559	99,75218718	215525	99,73645332	216094,5099
15	215722	215722	99,72016729	215643	99,68364856	216327,3547
16	215780	215790	99,72903525	214982	99,35561174	216376,3035
17	216419	216419	99,72578324	216320	99,68016408	217014,0890
18	217290	217290	99,74789577	217122	99,67077465	217839,1818
19	214633	214633	99,72796234	214320	99,58252873	215218,4753
20	301643	301643	99,86897125	301550	99,8381805	302038,7576
21	299987	299987	99,8442376	299970	99,83857952	300454,9959
22	304994	304990	99,83266583	304900	99,80320605	305501,2079
23	301955	301962	99,83660002	301857	99,80188425	302456,2134
24	304413	304413	99,83983311	304298	99,802116	304901,3510
25	296959	296900	99,82870846	296844	99,8098792	297409,4372
26	303270	303250	99,83017145	302980	99,74128721	303765,8812
27	306937	306937	99,84865121	306850	99,82034953	307402,2496
28	303111	303111	99,8369848	303025	99,80865 861	303605,9238
29	300499	300499	99,82671297	300299	99,76027234	301020,6297

ELE – Extended Local Eranching

g/l – genetický algoritmus s lokálnym vylepšovaním (x200 x3000 – 200 krát spuštěný algoritmus po 3000 generáciach)

Optimálně riešenie získané Ip relaxáciou

Tabuľka obsahuje výsledné hodnoty troch algoritmov. Ako prvý je prezentovaný algoritmus s názvom Extended Local Branching[5]. Po ňom nasleduje náš genetický algoritmus, pričom získané hodnoty boli dosiahnuté tak, že algoritmus bol pustený 200x, počet generácií bol kvôli vysokému exekučnému času obmedzený na 3000 a najlepšia dosiahnutá hodnota bola zapísaná. Veľkosť populácie bola 15. Tento algoritmus mal relatívne vysoký exekučný čas, avšak dosahoval takmer vždy nad 99,7% najlepšie nájdeného riešenia.

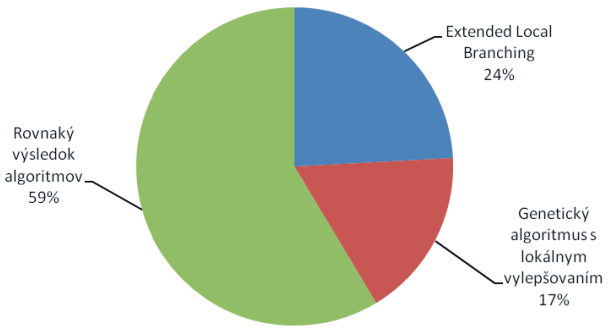
Pre praktické účely boli potom zmenené parametre nášho algoritmu tak, že bol pustený iba raz a počet generácií bol obmedzený len na 500. Po spustení sme však dosiahli pozoruhodné výsledky, všetky výsledky boli nad 99% najlepšie nájdeného riešenia, pričom exekučný čas bol neporovnateľne kratší[6] oproti pokusom s parametrami x3000 x200. V porovnaní s algoritmom

5) ELB – extended local branching

6) Exekučný čas pri pokuse s 3000 generáciami a 200 opakovaniami bol 114812,8 sekund (1913,547 minút alebo aj 31,892 hodín) pričom exekučný čas pri pokuse s 500 generáciami bol 56,828 sekúnd.

Extended Local Branching sme dosahovali porovnateľné výsledky. V 7 prípadoch mal Extended Local Branching lepšiu hodnotu účelovej funkcie, v 5 prípadoch náš genetický algoritmus a v 17 prípadoch bola hodnota účelovej funkcie zhodná. Získané výsledky prezentujeme aj vo forme grafu:

Obrázok č. 2 *Výsledok porovnávania Genetického algoritmu s lokálnym vylepšovaním a algoritmu Extended Local Branching na úlohách mknapcb9*[7)]

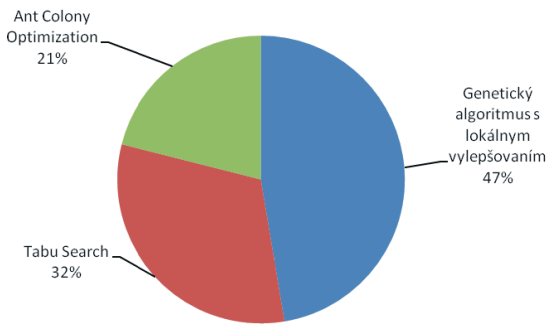


Hoci náš algoritmus nebol jasným favoritom, treba brať do úvahy aj exekučný čas. V prípade genetických algoritmov bývajú práve tieto časy ich hlavnou prednosťou. Hoci sa čas 40 minút môže zdať vysoký, musíme pripomenúť, že algoritmus generuje prípustné a relatívne kvalitné riešenia už po pár desiatkach sekúnd. V druhom prípade sme mali výsledky dosahujúce viac ako 99% najlepšie nájdeného riešenia do jednej minúty.

Následne sme porovnali náš algoritmus s algoritmami Tabu Search a Ant Colony Optimization. Kedže získané výsledky boli z roku 2008 prispôsobili sme tomu aj náš algoritmus tak, že algoritmus bol pustený 1x a maximálny počet generácií bol obmedzený na 1000. Exekučný čas bol v priemere 113,64 sekundy. K dispozícii sme mali výsledky pre jednotlivé sady úloh z knižnice ORLIB s označením mknapcb1 až mknapcb9, avšak z každej inštancie len prvých 9 príkladov. Náročnosťou sa však líšili príklady len minimálne a inštancia mknapcb9 obsahovala porovnateľných 29 príkladov líšiacich sa len hodnotou tesnosti. O dosiahnutých výsledkoch; teda hodnoty účelovej funkcie informuje tabuľka, ktorá sa nachádza v prílohe 1.

Analýza výsledkov preukázala, že v 47 % prípadov dosiahol náš genetický algoritmus s lokálnym vylepšovaním lepšiu hodnotu účelovej funkcie ako dva konkurenčné algoritmy.

Obrázok č. 3 *Výsledok porovnávania Genetického algoritmu s lokálnym vylepšovaním a algoritmov Tabu Search a Ant Colony Optimization na úlohách mknapcb1 – mknapcb9*[8)]



7) Zdroj obrázku: Vlastný obrázok analýzy výsledkov experimentov

8) Zdroj obrázku: Vlastný obrázok predstavujúci výsledky experimentov

Všetky výsledky boli získané s rovnakým nastavením parametrov genetického algoritmu, čo znamená, že pri dôkladnejšej analýze má algoritmus určite priestor na zlepšenie sa. Hľadanie ideálnych parametrov pre každý jeden príklad by však bolo časovo náročné, pretože by bolo treba uskutočniť veľký počet pokusov, aby sa potvrdila štatistická významnosť daného parametra.

Použitá literatúra

- [1] Ben-Tal, A., Ghaoui L., Nemirovskii A.: Robust Optimalization, Princeton University Press 2009, Princeton ISBN: 0691143684, 9780691143682.
- [2] Hoff, A., Lokkentangen, A., Mittet, I.: Genetic algorithms for 0/1 multidimensional knapsack problems, Molde College, Norway
- [3] Chu, P., Beasley, J.: A genetic algorithm for the multidimensional knapsack problem. In: Journal of Heuristics Volume 4 Number 1, DOI: 10.1023/A:1009642405419
- [4] Vasquez, M., Hao, J.: An hybrid approach for the 0-1 multidimensional knapsack problem. In: 17th International Joint Conference on Artificial Intelligence
- [5] Glover, F., Laguna, M.: Tabu Search, Kluwer, Norwell MA, 1997
- [6] Hanafi, S., Freville, A.: An efficient tabu search approach for the 0-1 multidimensional knapsack problem. In: CEJOR Volume 106, April 1998
- [7] Glover, F.,Laguna, M.: Tabu Search, Kluwer Academic Publishers, 1997, ISBN: 0-7923-9965-X
- [8] Thiel, J. and Voss, S. (1993) : Some experiences on solving multiconstraint zero-oneknapsack problems with genetic algorithms. In: INFOR.Vol 32, No 4, pp 226-243.
- [9] Richardson, J., Palmer, R., Liepins, G. and Hilliard, M. (1989) : Some Guidelines for Genetic Algorithms with Penalty Functions. In: Proceedings of the Third International Conference on Genetic Algorithms, pp 191-197.
- [10] Yang, X. – Teo, K. – Caccetta, L. 2001. Optimization methods and applications. Dordrecht : Springer, 2001. ISBN9780792368663.
- [11] Fidanova, S. Heuristics for multiple knapsack problem. [online]. [cit. 2011-April-27]. http://www.google.sk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CFUQFjAA&url=http%3A%2F%2Fwww.iadis.net%2Fdl%2Ffinal_uploads%2F200501C042.pdf&ei=7MXCT7rEM86M-wax96jxCQ&usg=AFQjCNHa9ztXsVjRsHEHguXLeGYDLH3O7g.
- [12] Lichtenberger, D. An Extended Local Branching Framework and its Application to the Multidimensional Knapsack Problem. [online]. [cit. 2011-Júl-4]. https://www.ads.tuwien.ac.at/publications/bib/pdf/lichtenberger_05.pdf.

Ing. Ján Pittner, PhD.
Katedra aplikovanej informatiky Fakulty hospodárskej informatiky Ekonomickej univerzity v Bratislave, Dolnozemská 1, 852 35 Bratislava 5,
E-mail: janci.pittner@gmail.com

Príloha č.1

mknapcbl

inštancia	g/l xl xl000	TS algoritmus	ACO algoritmus
1	24329	24342	24713,07143
2	24149	24343,23571	24930,21429
3	23494	23630,85714	24253,38571
4	23370	23300	23320
5	43190	43171,71429	43595,1
6	430025	42955,35714	43013,57143
7	60390	60220,23571	60394,37143
8	61199	60909,14236	61131,64236
9	61460	60749	61458,62357

mknapcb2

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	59956,22357	60306	59954,72857
2	60510,64236	60503,57143	60509,14236
3	60947,7	60361,57143	60946,2
4	109497,2357	103927,1429	109495,7357
5	109310,7714	109503	109309,2714
6	103036,3236	103773,7143	103035,3236
7	103036,3236	150050	108035,3286
8	150136,9357	149377,1429	150135,4357
9	152311,4429	150065,7143	152809,9429

mknapcb3

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	119355,0275	120302,4236	119351,6
2	120305,3561	120257,1429	120302,4286
3	123369,2275	121270,1429	123365,3
4	219901,9132	220321,4286	219898,4857
5	221250,5413	219941,4236	221247,1143
6	219263,3413	217633,5714	219265,4143
7	299264,4413	301121,4236	299261,0143
8	305076,4561	302077,1429	305 073,0286
9	302273,4347	300917,1429	302275,0571

mknapcb4

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	22657	23167,57143	23547
2	21353	22815,14286	22801,28571
3	22511	23044,42357	23286,28571
4	22614	22610	22615
5	44660	43739	44646,74236
6	63350	42392,57143	43360,37143
7	57020	59142,35714	57041,47143
8	59330	60016,57143	59348,95714
9	59640	59919	59631,3

mknapcb5

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	59651,05999	53645,35714	59646,51429
2	59200,40284	59149,71429	59195,35714
3	58777,75999	53323	58773,21429
4	110396,06	108617	110391,5143
5	109721,56	103694	109717,0143
6	108996,5336	108391,1429	108992,0429
7	151371,4023	151370	151366,8571
8	151279,7023	150907,1429	151275,1571
9	150239,3171	15 1252,3571	150235,2714

mknapcb6

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	117519,2357	117299,2357	116877,6286
2	113253,1429	113033,1429	118511,0357
3	113217,1429	117997,1429	119227,5857
4	214902,3571	214632,3571	216944,7571
5	217201,4236	216931,4236	217019,9429
6	216237,1429	216017,1429	217250,5571
7	301133,5714	300963,5714	301955,6143
8	301354,2357	301134,2357	299335,4143
9	300635,7143	300465,7143	300618,2286

mknapcb7

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	22440,21046	22290,23571	22429,68571
2	22463,33139	22333,23571	22457,35714
3	22433,43904	22115,42357	22422,91429
4	40766,66761	41295,14236	40756,14236
5	41396,23904	42075,23571	41385,71429
6	40313,55332	42093,23571	40303,02357
7	57659,33904	58663,14236	57649,31429
8	59155,01046	59634,35714	59144,43571
9	56115,31046	59380,85714	56104,78571

mknapcb8

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	57326,24141	57142,71429	57311,98571
2	57997,32713	57307,57143	57933,07143
3	57326,24141	57002,42857	57311,98571
4	107506,3414	106335,3571	107492,0357
5	106453,3123	107175,5714	106439,0571
6	107043,5271	106737,4236	107029,2714
7	147313,6557	148472,8571	147304,4
8	149434,3557	150711,4236	149420,1
9	143433,3414	150354,2857	148474,0857

mknapcb9

inštancia	g/l xl x1000	TS algoritmus	ACO algoritmus
1	115599	114250	114300
2	114320	114439	115375
3	116001	115125	115152
4	114850	115820	116250
5	116100	115500	115540
6	115250	113750	113980
7	113682	114230	114225
8	114150	115350	115250
9	116750	116720	116850