Memetic Algorithm for Solving the 0-1 Multidimensional Knapsack Problem

Conference Paper *in* Lecture Notes in Computer Science · September 2015

DOI: 10.1007/978-3-319-23485-4_31

3 authors:

Abdellah Rezoug
University of Boumerdes

Dalila Boughaci
University of Sciences and Technology Houari Boumediene

Mohamed Bader-El-Den

Memetic Algorithm for Solving the 0-1 Multidimensional Knapsack Problem.

Abdellah REZOUG[1], Dalila BOUGHACI[2], and Mohamed BADR-EL-DEN[3].

Departement of Computer Science, University Mhamed Bougarra of Boumerdes, Algeria.[1]
USTHB- FEI- Department of Computer Science Beb-Ezzouar, Algiers, Algeria.[2]
School of Computing, Faculty of Technology University of Portsmouth, United Kingdom.[3]
abdellah.rezoug@gmail.com; dboughaci@usthb.dz
dalila_info@yahoo.fr; mohamed.bader@port.ac.uk.

Abstract. In this paper, we propose a memetic algorithm for the Multidimensional Knapsack Problem (MKP). First, we propose to combine a genetic algorithm with a stochastic local search (GA-SLS), then with a simulated annealing (GA-SA). The two proposed versions of our approach (GA-SLS and GA-SA) are implemented and evaluated on benchmarks to measure their performance. The experiments show that both GA-SLS and GA-SA are able to find competitive results compared to other well-known hybrid GA based approaches.

Keywords: Multidimensional knapsack problem, stochastic local search, genetic algorithm, simulated annealing, local search, memetic algorithm.

1 Introduction

The Multidimensional Knapsack Problem (MKP) is a strong NP-hard combinatorial optimization problem [14]. The MKP has been extensively considered because of its theoretical importance and wide range of applications. Many practical engineering design problems can be formulated as MKP such as: the capital budgeting problem [17], the project selection [2] and so on.

The solutions for MKP can be classified into exact, approximate and hybrid. The exact solutions are used for problems of small size. Branch and bound, branch and cut, linear, dynamic and quadratic programming, etc. are the principal exact methods used for solving MKP [13, 21]. The approximate solutions are used when the data size is high but it is not sure to obtain the optimal results. They are mainly based on heuristics such as: simulated annealing, tabu search, genetic algorithm, ant colony particle swarm, harmony search, etc [5, 6, 20]. The hybrid solutions combine two or more exact or/and approximate solutions. These solutions are the most used in the field of optimization and especially for MKP such as [4, 8, 9, 10, 11, 12, 18] and so on.

In this paper, we propose a memetic algorithm for MKP. We developed two versions of our method for MKP. The first denoted GA-SLS is a GA combined with the stochastic local search (SLS) [3]. The second denoted GA-SA is a combination of GA with the simulated annealing (SA) [16]. The two versions are implemented and evaluated on some well-known benchmarks for MKP where the sizes of benchmarks arrange from small to large. A comparative study is done with a pure GA and some algorithms for MKP. The objective is to show the impact of the local search in the performance of the memetic approach.

The rest of the paper is organized as follows. Section 2 gives the MKP model. The proposed Memetic approaches are detailed in Section 3. Section 4 describes the experiments. Finally, Section 5 concludes the paper.

2 The multidimensional knapsack problem

The MKP is composed of N items and a knapsack with m different capacities b_i where $i \in \{1, \dots, m\}$. Each item j where $j \in \{1, \dots, n\}$ has a profit c_j and can take a_{ij} of the capacity i of the knapsack. The goal is to pack the items in the knapsack so as to maximize the profits of items without exceeding the capacities of the knapsack. The MKP is modeled as the following integer program:

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

$$\textbf{Subject to}: \sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad i \in \{1 \dots m\} \tag{2}$$

$$x_j \in \{0, 1\} \qquad j \in \{1 \dots n\} \tag{3}$$

3 The Proposed Approaches for MKP

Two versions of memetic approach have been studied. The first one is the Genetic Algorithm-Stochastic Local Search (GA-SLS) where GA is combined with SLS. The second one is the Genetic Algorithm-Simulated Annealing (GA-SA) which is GA combined with SA. The structures of both approaches are similar in the GA part. The difference is in the local search. The GA-SLS applies SLS while GA-SA applies SA. Their process consists in: Create the initial population P using the Random Key method (RK)[1] and initialize $Q = \{\}$, NI and $T = T_0$ (T for GA-SA only). Select two parents X_1 and X_2 that are the two best individuals in P and $X_1, X_2 \notin Q$. Exchange NCB items between the parents X_1 and X_2 to produce two new infeasible offspring X_1' and X_2', then if conflict exists in X_1' or X_2', repeatedly remove either the worst items or an item chosen randomly according to a probability rp. Push the two parents X_1 and X_2 in Q. Apply the local search (SLS for GA-SLS or SA for GA-SA) on offspring X_1', X_2'. Find the best individuals X_{best} in P and replace randomly a number of items in X_1' and X_2' by items in X_{best}. If the quality of X_1' and X_2' is better than the two worst individuals in P, then they replace them. If the number of iterations NI is not attend then go to *STEP 2.*. Otherwise return the best individual in P. The GA-SLS and GA-SA can be expressed by Algorithm 1.

Algorithm 1 GA-SLS Algorithm.

Require: An MKP instance, NI and $Q = \phi$.
Ensure: An best solution found X^*.

Create the initial population P by the RK method.
for ($Cpt = 1$ to NI) **do**
  Selection of the two best individuals X_1, X_2 in P and $X_1, X_2 \notin Q$.
  Crossover X_1, X_2 to produce offsprings X_1', X_2'
  Repair offsprings X_1', X_2'
  Apply the local search method on X_1', X_2'
  Mutation on X_1', X_2' with X_{best} of P
  $X_{worst} \longleftarrow$ the worst individual in P
  if $(f(X_1') > f(X_{worst}))$ **then**
    $P = P - \{X_{worst}\}$
    $P = P \cup \{X_1'\}$
  end if
  $X_{worst} \longleftarrow$ the worst individual in P
  if $(f(X_2') > f(X_{worst}))$ **then**
    $P = P - \{X_{worst}\}$
    $P = P \cup \{X_2'\}$
  end if
  $Q = Q \cup \{X_1, X_2\}$
end for
Return the best individual found.

4 The Experiments

GA, GA-SA and GA-SLS were implemented in C++ on 2 GHz Intel Core 2 Duo processor and 2 GB RAM. They were tested on the OR-Library[22] 54 benchmarks, with $m = 2$ to 30 and $n = 6$ to $n = 105$ and on the OR-Library GK [22] with $m = 15$ to $m = 50$ and $n = 100$ to $n = 1500$. In all experiments the parameters are chosen empirically such as: the number of iteration $NI = 30000$, the population size $PS = 100$, the waiting time $WT = 50$, the number of crossing bites $NCB = 1/10$. the initial temperature $T_0 = 50$, the walk probability $wp = 0.93$, the number of local iteration $N = 100$ and the number of runs is 30.

Results for the SAC-94 standard instances The average fitness (*Result*), the average gap (*GAP*), the best (*Best*) and the worst fitness (*Worst*), the number of success runs (*NSR*), the number of success instance (*NSI*) and the rate of success runs (*RSR*) have been recorded by analyzing the recorded obtained fitness. Also, the average CPU runtime (*Time*) has been calculated. All the results and statistics computed by the GA, GA-SA and GA-SLS are reported in Tables 1-2. From results, GA resolved to optimality one instance of 54 with average gap of 4,454 %, GA-SA 35 instances with a global gap of 0,093 % and GA-SLS 39 instances with a global gap of 0,0221 %. GA-SLS reached the optimum at least once in 50 instances followed by GA-SA in 49 instances then GA in 18 instances. The *RSR* show that GA-SLS totally solved instances of groups *hp, pb* and *sento* followed by GA-SA. GA-SLS obtained a total *RSR* better than GA-SA (92,83% and 81,91%, respectively). At the same time, GA-SA and GA-SLS widely surpass GA (13,65%). *RSR* shows that hybridization of GA with SA has improved the success rate of 79,18% and its hybridization with SLS of 68,49%. From Table 2, GA is the fastest with an global average CPU time of 1.818 *sec*.

Table 1. Comparison of GA, GA-SA and GA-SLS on SAC-94 datasets.

		GA		GA-SA		GA-SLS	
Dataset	Opt	Result	GAP	Result	GAP	Result	GAP
hp	3418	3381,07	1,080	3418	0	3418	0
	3186	3120,63	2,052	3186	0	3186	0
Average	3302	3250,85	1,566	**3302**	**0**	**3302**	**0**
	3090	3060,27	0,962	3090	0	3090	0
	3186	3139,13	1,471	3186	0	3186	0
pb	95168	93093,5	2,180	95168	0	95168	0
	2139	2079,93	2,762	2139	0	2139	0
	776	583,767	24,772	776	0	776	0
	1035	1018,13	1,630	1035	0	1035	0
Average	17565,666	17162,454	5,629	**17565,666**	**0**	**17565,666**	**0**
	87061	86760,1	0,346	87061	0	87061	0
	4015	4015	0	4015	0	4015	0
pet	6120	6091	0,474	6120	0	6120	0
	12400	12380,3	0,159	12400	0	12400	0
	10618	10560,9	0,538	10609,1	0,084	10608,6	0,089
	16537	16373,9	0,986	16528,1	0,054	16528,3	0,053
Average	22791,833	22696,866	0,417	**22788,866**	**0,023**	22788,816	0,024
sento	7772	7606,03	2,135	7772	0	7772	0
	8722	8569,7	1,746	8721,2	0,009	8722	0
Average	8247	8087,865	1,941	8246,6	0,005	**8247**	**0**
	141278	141263	0,011	141278	0	141278	0
	130883	130857	0,020	130883	0	130883	0
	95677	94496,2	1,234	95677	0	95677	0
weing	119337	118752	0,490	119337	0	119337	0
	98796	97525,3	1,286	98796	0	98796	0
	130623	130590	0,025	130623	0	130623	0
	1095445	1086484,2	0,818	1094579,6	0,079	1095432,7	0,0011
	624319	581683	6,829	623727	0,095	624319	0
Average	304545,375	297707,062	1,339	304362,625	0,022	**304543,22**	**0,0001**
	4554	4530,03	0,526	4554	0	4554	0
	4536	4506,77	0,644	4536	0	4536	0
	4115	4009,37	2,567	4115	0	4115	0
	4561	4131,07	9,426	4561	0	4561	0
	4514	4159,73	7,848	4514	0	4514	0
	5557	5491,73	1,175	5557	0	5557	0
	5567	5428,37	2,490	5567	0	5567	0
	5605	5509,43	1,705	5605	0	5605	0
	5246	5104,5	2,697	5246	0	5246	0
	6339	6014,23	5,123	6339	0	6339	0
	5643	5234,33	7,242	5643	0	5643	0
	6339	5916	6,673	6339	0	6339	0
	6159	5769,5	6,324	6159	0	6159	0
	6954	6495,6	6,592	6954	0	6954	0
weish	7486	6684,6	10,705	7486	0	7486	0
	7289	6878,4	5,633	7289	0	7289	0
	8633	8314,73	3,687	8629,5	0,041	8633	0
	9580	9146,5	4,525	9559,63	0,213	9568,63	0,119
	7698	7223,17	6,168	7698	0	7698	0
	9450	8632,1	8,655	9448,63	0,014	9449,37	0,007
	9074	8114,4	10,575	9073,23	0,008	9073,33	0,007
	8947	8321,17	6,995	8926,73	0,227	8938,83	0,091
	8344	7603,77	8,871	8321,97	0,264	8318,93	0,3
	10220	9685,77	5,227	10152,9	0,657	10164,2	0,546
	9939	9077,9	8,664	9900,07	0,392	9910,73	0,284
	9584	8728,87	8,922	9539,4	0,465	9560,53	0,245
	9819	8873,7	9,627	9777,9	0,419	9802,03	0,173
	9492	8653,57	8,833	9423,87	0,718	9442,17	0,525
	9410	8466,67	10,025	9359,5	0,537	9369,5	0,430
	11191	10250,1	8,408	11106,3	0,757	11128,7	0,557
Average	7394,833	6898,536	6,218	7379,386	0,157	**7386,772**	**0,109**

Table 2. Results of NSR, RSR and Time parameters obtained by GA, GA-SA and GA-SLS.

	GA			GA-SA			GA-SLS		
	NSR	RSR	Time	NSR	RSR	Time	NSR	RSR	Time
hp	1	1,67	1,798	**2**	**100**	6,077	**2**	**100**	7,101
pb	4	3,33	1,811	**6**	**100**	3,443	**6**	**100**	4,681
pet	4	32,78	1,395	**6**	**81,67**	11,296	**6**	80,55	12,179
sento	0	0,00	2,616	**2**	66,67	46,584	**2**	**100**	24,277
weing	6	39,58	1,669	**8**	76,25	10,259	**8**	**99,58**	10,586
weish	3	4,55	1,620	25	66,89	17,352	**26**	**76,88**	17,146
Average	18	13,65	1,818	49	81,91	15,835	**50**	**92,83**	**12,662**

Table 3. Results of the approaches test on the GK dataset.

Dataset		GA		GA-SA		GA-SLS	
Instance	Optimal	Result	Gap	Result	Gap	Result	Gap
1	3766	3673,5	2,456	3704,3	1,638	3704,2	1,641
2	3958	3860,7	2,458	3894,8	1,596	3897,7	1,523
3	5656	5511,5	2,554	5538,8	2,072	5535,7	2,127
4	5767	5630,6	2,365	5655,2	1,938	5655,4	1,935
5	7560	7351,3	2,76	7395,1	2,181	7391,3	2,231
6	7677	7505,7	2,231	7528,4	1,935	7528,1	1,939
7	19220	18612,1	3,162	18691	2,752	18692,4	2,745
8	18806	18330,2	2,53	18393	2,196	18392,4	2,199
9	58091	56198,5	3,257	56371,1	2,96	56381,4	2,943
10	57295	55837,9	2,543	55959,3	2,331	55961,9	2,326
Average	18779,6	18251,2	2,632	18313,1	2,484	**18314,05**	**2,479**

Results for the ten large instances From results on the GK shown in Table 3 GA-SA has the best value of *Result* and *GAP* for 1, 3, 5, 6 and 8 instances. GA-SLS has the best value of *Result* and *GAP* for instances 2, 4, 7, 9 and 10. Global, GA-SLS has the best performance for all instances with an total average *GAP* of 2.479 %. GA-SA has almost the same performance with average *GAP* of 2.484 %. Also, GA is not very far from GA-SA and GA-SLS with an total average *GAP* of 2.632 %.

Comparison with other GA approaches We compared results of the proposed GA-SA and GA-SLS to other approaches. The results of the KHBA [15], COTRO [7], TEVO [19], CHEBE [6] and HGA [10] were obtained from [10]. From Table 4 GA-SA and GA-SLS gave improved results compared to KHBA, COTRO and TEVO, for almost all instances. GA-SA and GA-SLS were able to find the optimal solutions to 6, and 3 of 7 problems respectively. Furthermore, GA-SA performs results quite similar to CHEBE and HGA.

5 Conclusion

In this paper we addressed the multidimensional knapsack problem (MKP). We proposed, compared and tested two combinations: GA-SLS and GA-SA. GA-SLS combines the genetic algorithm and the stochastic local search (SLS) while GA-SA uses the simulated annealing (SA) instead of SLS. The experiments have shown the performance of our methods for MKP. Also, the hybridization of

Table 4. Comparison of GA-SA and GA-SLS with some GA-based approaches.

		KHBA	COTRO	TEVO	CHBE	HGA	GA-SA	GA-SLS
problem	Optimum	Sol A.	Sol A.	Sol A.	Sol A.	Sol A.	Sol A.	Sol A.
sento1	7772	7626	7767,9	7754,2	**7772**	**7772**	**7772**	**7772**
sento2	8722	8685	8716,3	8719,5	**8722**	**8722**	8721,2	**8722**
weing7	1095445	1093897	1095296,1	1095398,1	**1095445**	**1095445**	1094579,6	1095432,7
weing8	624319	613383	622048,1	622021,3	**624319**	**624319**	623727	**624319**
weish23	8344	8165,1	8245,8	8286,7	**8344**	**8344**	8321,97	**8344**
hp1	3418	3385,1	3394,3	3401,6	**3418**	**3418**	**3418**	**3418**
pb2	3186	3091	3131,2	3112,5	**3186**	**3186**	**3186**	**3186**

GA with local search methods allows to greatly improving its performance. As perspectives, we plan to study the impact of local search method when used with other evolutionary approaches such as: harmony search and particle swarm.

References

[1] Bean, J.C. (1994) 'Genetics and random keys for sequencing and optimization', *in ORSA Journal of Computing*, Vol.6, No.2, pp.154-160.

[2] Beaujon, G.J. Martin, S.P. McDonald C.C., (2001) 'Balancing and optimizing a portfolio of R&D projects', *Naval Research Logistics* 48, pp.18-40.

[3] Boughaci, D. Benhamou, B. Drias, H. (2010) 'Local Search Methods for the Optimal Winner Determination Problem in Combinatorial Auctions', *Math Model Algor*, 9, No.1, pp.165-180.

[4] Chih, M. Lin, C.J., Chern, M.S. Ou, T.Y. (2014) 'Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem', *Applied Mathematical Modelling* 38, pp.1338-1350.

[5] Cho, J.H. and Kim, Y.D. (1997) 'A simulated annealing algorithm for resource-constrained project scheduling problems', *Operational Research Society*, Vol.48, pp.736-744.

[6] Chu, P. and Beasley, J. (1998) 'A Genetic Algorithm for the Multidimensional Knapsack Problem', *Heuristics*, Vol.4, pp.63-86.

[7] Cotta, C. and J. Troya (1994) 'A Hybrid Genetic Algorithm for the 0-1 Multiple Knapsack problem,' *Artificial Neural Nets and Genetic Algorithm* 3, pp.250-254.

[8] Deane, J. and Agarwal, A. (2013) 'Neural, Genetic, And Neurogenetic Approaches For Solving The 0-1 Multidimensional Knapsack Problem', *Management & Information Systems* - First Quarter 2013 Vol 17, Numb 1.

[9] Della Croce, F. Grosso, A. (2012) 'Improved core problem based heuristics for the 0-1 multi-dimensional knapsack problem', *Comp& Oper. Res* 39 pp.27-31.

[10] Djannaty, F. and Doostdar, S. (2008) 'A Hybrid Genetic Algorithm for the Multidimensional Knapsack Problem'. *Contemp. Math. Sciences*, Vol. 3, no. 9, pp.443-456

[11] Feng, L. Ke, Z. Ren, Z. Wei, X. (2010) 'An ant colony optimization approach for the multidimensional knapsack problem', *Heuristics* 16, pp.65-83.

[12] Feng, Y. Jia, K. and He, Y. (2014) 'An Improved Hybrid Encoding Cuckoo Search Algorithm for 0-1 Knapsack Problems', *Computational Intelligence and Neuroscience*, ID 970456.

[13] Fukunaga, A.S. (2011)'A branch-and-bound algorithm for hard multiple knapsack problems', *Annals of Operations Research* , Vol.184, pp.97-119.

[14] Garey, M.R. and Johnson, D.S. (1979) 'Computers and intractability: A guide to the theory of NP-completeness'. New York: W. H. Freeman & Co.

[15] Khuri, S., T. Bäck, and J. Heitkötter. (1994). 'The Zero-One Multiple Knapsack Problem and Genetic Algorithms,' Proceedings of the ACM Symposium on Applied Computing, pp.188-193.

[16] Kirkpatrick, S. Gelatt, C.D. Vecchi, P.M. (1983) 'Optimization By Simulated Annealing'. *Science* Vol.220, pp.671-680.

[17] Meier, H. Christofides, N. Salkin, G. (2001)'Capital budgeting under uncertainty-an integrated approach using contingent claims analysis and integer programming', *Operations Research* 49, pp.196-206.

[18] Tuo, S. Yong, L. and Deng F. (2014) 'A Novel Harmony Search Algorithm Based on Teaching-Learning Strategies for 0-1 Knapsack Problems', *The Scientific World Journal* Article ID 637412, 19 pages.

[19] Thiel, J. and Voss, S. (1994). 'Some Experiences on Solving Multiconstraint Zero-One Knapsack Problems with Genetic Algorithms,' *INFOR* 32, pp.226-242.

[20] Vasquez, M. Vimont, Y. (2005) 'Improved results on the 0-1 multidimensional knapsack problem', *Eur. J. Oper. Res.* 165 pp.70-81.

[21] Yoon, Y. and Kim, Y.H. (2013) 'A Memetic Lagrangian Heuristic for the 0-1 Multidimensional Knapsack Problem', *Discrete Dynamics in Nature and Society*, Article ID 474852, 10 pages.

[22] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/mknapinfo.html