Journal of Soft Computing and Artificial Intelligence

Journal homepage: https://dergipark.org.tr/en/pub/jscai

International Open Access

Volume 03
Issue 02

December, 2022

Research Article

Solving Multidimensional Knapsack Problem with Bayesian Multiploid Genetic Algorithm

Emrullah Gazioğlu[1]

[1]*Computer Engineering Department, Engineering Faculty, Sirnak University, 73000, Sirnak, Türkiye*

ARTICLE INFO

Article history:
Received
Revised
Accepted

Keywords:
Bayesian Optimization Algorithm
MKP
Genetic Algorithm
Optimization

ABSTRACT

Solving optimization problems is still a big challenge in the area of optimization algorithms. Many proposed algorithms in the literature don't consider the relations between the variables of the nature of the problem. However, a recently published algorithm, called "Bayesian Multiploid Genetic Algorithm" exploits the relations between the variables and then solves the given problem. It also uses more than one genotype unlike the simple Genetic Algorithm (GA) and it acts like an implicit memory in order to remember the old but good solutions. In this work, the well-known Multidimensional Knapsack Problem (MKP) is solved by the Bayesian Multiploid Genetic Algorithm. And the results show that exploiting relations between the variables gets a huge advantage in solving the given problem.

1. Introduction

Constraint Optimization Problems (COP) are still a big challenge in the area of computer science. One of them is the Multidimensional Knapsack Problem (MKP). The MKP is an extended version of the standard 0-1 Knapsack Problem (KP). While standard KP has only one resource, the MKP can has more than one resource. MKP is actually can be considered a real-world problem. Many real-world problems can be solved by MKP, such as cutting stock[1], loading problems [2], resource allocation for distributed computing[3], project selection[4], etc. MKP is still a current benchmark problem that continues to be solved with different approaches in recent years[5]–[8].

In the past ten years, many metaheuristic (MH) techniques are applied to solve different NP-Hard global optimization problems. Some of them are the following: Estimation of Distribution Algorithms (EDA)[9], Artificial Bee Colony (ABC)[10], Harmony Search (HS)[11], Ant Colony Optimization (ACO)[12], Whale Optimization Algorithm (WOA)[13], Bat Algorithm (BA)[14], etc. However, all of them mentioned before don't consider the interactions between the problem's variables. In an ecosystem, all the objects have connected some way and they affect each other.

For this reason, it is important to exploit interactions between the variables and then use them to solve the problem effectively.

In this work, the Bayesian Multiploid Genetic Algorithm (BMGA)[15] is used, which has both an implicit memory scheme (to remember old solutions) and a Bayesian Network (to exploit interactions between variables) in order to solve the well-known Multidimensional Knapsack Problem (MKP) considering as a real-world benchmark problem.

To evaluate the BMGA's performance, six algorithms are selected as comparative algorithms. Note that, the results for the six algorithms listed below are already taken from the [16] and compared

[1] Corresponding author
e-mail: gazioglu@sirnak.edu.tr
DOI: 10.55195/jscai.1216193

to the BMGA's results.

- Moth Search Algorithm (MS)[17, s.]: A new optimization algorithm that is inspired by the Levy flights and the phototaxis of the months. In this method, the fittest individual is considered the light source. The moths close to the fittest one fly in the form of Levy flights. On the other hand, because of the phototaxis, the moths far from the fittest one fly to the fittest one with big steps. These two different behaviors are the exploration and the exploitation of any other optimization method.
- Self-Learning MS (SLMS)[16]: In regular MS, each individual update their positions according to the fittest one. But it may cause it to fall into the local optimum. Authors, introduce a new self-learning strategy to enable individuals to update their positions not only according to the fittest one but also the closer individuals which have fitter than themselves.
- Modified multi-verse optimization algorithm (MMVO)[18]: In this algorithm, authors are inspired by the popular multi-verse theory which is based on three concepts: the wormhole, the black hole, and the white hole. In the algorithm, each universe is considered a solution candidate. If an individual gets close to the white hole, means it is getting better fitness values, and vice-versa (closing to the black hole), it is getting worse fitness values. And the wormholes are used as a diversification operator in the algorithm to maintain the diversity in the population.
- Binary Gravitational Search Algorithm (BGSA)[19]: This is the binary version of GSA which is based on Newton's laws of gravity and motion: The gravitational force affects the objects and makes them attract each other. In the algorithm, each object is considered an agent, and each one of them has its mass. And their masses define their fitness values. As time passes, objects are attracted by the fitter masses which leads them to get a better fitness value.
- Binary Hybrid Topology Particle Swarm Optimization (BHTPSO)[19]: In the simple PSO[20], the particles may fall into local optimum if their velocities are zero. Because zero velocity means the particle's fitness value is good and shouldn't be changed. To overcome this, a small value is added to their velocities for acceleration and enables them to escape from the local optimum.
- Binary Hybrid Topology Particle Swarm Optimization Quadratic Interpolation (BHTPSO-QI)[19]: This is the form of BHTPSO which is incorporated with a quadratic crossover operator.

Algorithm 1 Pseudocode of BMGA

```
$g \leftarrow number\ of\ genotypes$
$pop \leftarrow init(popSize, g)$
$randomly\ generate\ a\ probVec[\,]$
while $a\ termination\ condition\ not\ met$ do
    $genotype2phenotype(pop, probVec[\,])$
    $best \leftarrow evaluate(pop)$
    $BN \leftarrow constructBN(pop)$
    $auxPop \leftarrow sampleBN(BN)$
    $probVec[\,] \leftarrow constructProbVec(auxPop)$
    $pop \leftarrow tournamentSelection(pop)$
    $pop \leftarrow uniformCrossover(pop)$
    $pop \leftarrow bitwiseMutation(pop)$
return $best$
```

This paper continues as follows: In Section 2, the BMGA is explained. In Section 3, MKP and its datasets are explained. In Section 4, the results are shown and finally, Section 5 concludes the paper.

2. Bayesian Multiploid Genetic Algorithm

The BMGA is constructed on the simple GA. However, it differs from simple GA in many ways.

First, the individuals in the simple GA have only one chromosome to represent a candidate solution. Both genetic operators and fitness calculations are done over this chromosome for each individual. However, in BMGA, each individual has two different structures: The genotypes and a phenotype. As we know, in nature, all living things have genotypes and phenotypes. Genotypes are inherited from the parents. However, the phenotype is the one that decides a living thing will look like to what.

The number of genotypes in an individual can be more than one. This feature provides an implicit memory scheme to the algorithm. Because all the genetic operators (crossover – mutation) are executed on the genotypes and the fitness calculations are executed on the phenotype of the individual. In this way, genotypes act like a memory. On the other hand, each individual has only one phenotype and its fitness value is calculated using that phenotype. That is, no matter how many genotypes an individual has, only one phenotype determines its fitness value. The structure of an individual is illustrated in Figure 1.

Second, BMGA uses a well-known Bayesian

Optimization Algorithm (BOA)[21] in it, in order to exploit the interactions between the variables. The BOA starts with a randomly generated population and creates a Bayesian Network (BN) by using this population. Since there is no prior information, BOA uses a greedy algorithm to form different BNs and then measure their quality using a special metric. After the algorithm finds the most suitable BN, then samples new individuals using that final BN. This procedure continues until a termination condition is met.

	0	1	2	3	4	5
genotype #1	1	1	0	1	1	0
genotype #2	0	0	0	1	0	0
genotype ...	...	...	...	...	...	...
genotype #k	1	1	0	1	0	0
Prob. Vector	**0.5**	**0.3**	**0.7**	**0.1**	**0.8**	**0.9**
phenotype	?	?	0	1	?	0

Figure 1 Illustration of an individual

In BMGA, the BOA is used in order to form a probability vector to determine the phenotypes of the individuals in the population. This works like this: First, the GA part of the BMGA randomly generates a population. In this generating part, only the genotypes of the individuals are generated. For instance, if an individual has four genotypes, and the size of the population is 100, then 400 genotypes are generated. Next, a probability vector is randomly formed to determine the phenotypes of the individuals for the first iteration. The probability vector is formed via genotypes of the best k% individuals of the population by calculating the probability of being 1 for each gene using Equation 1. After the probability of being 1 is calculated the Equation 1, for each gene in the phenotype, a random real number is generated between 0 and 1. This generated value is then compared to the corresponding value in the probability vector to determine the value of the phenotype. Assuming that number of genotypes is g, this comparison is held by using Equation 2 and Equation 3.

$$ProbVector_i = \frac{number\ of\ ones\ for\ the\ i^{th}\ gene}{size\ of\ the\ given\ population} \quad (1)$$

$$val = (rand() < ProbVector_i)?\,1:0 \quad (2)$$

$$pheno_i = \begin{cases} 0, & \sum^{g} geno_i = 0 \\ 1, & \sum^{g} geno_i = g \\ val, & other \end{cases} \quad (3)$$

Once the phenotypes are determined, the fitness values of all individuals are calculated. After this first iteration, for the latter iterations, a new BN is constructed by using the same rule, and then a second population is sampled via BN. This second population is named auxiliary population, in short, “aux-pop”. The aux-pop has only one chromosome like in the simple GA. For the latter iterations, the probability vector is formed by the aux-pop, again, using Equation 1.

After explaining the key points of the BMGA, now we can explain it in general. The general pseudocode of the BMGA is given in Algorithm 1. After the BOA parts of the BMGA are executed, next, the GA part of the BMGA starts to run.

The standard tournament selection (size of n) is applied in the BMGA’s selection phase: Randomly chosen n solution candidates are compared and the fittest one is passed on to the next generation. This operation is performed until the size of the next generation is satisfied. Without recalculating its fitness value, the elite solution candidate from the last iteration is passed on to the current generation (elitism).

The uniform crossover approach is employed for crossover operation. It begins by randomly generating a mask vector in binary and then selecting two individuals, say i_1 , and i_2, at each iteration. For each variable j (genes in chromosome), if the mask vector’s corresponding value is 1, i_1’s 1st genotype’s j^{th} variable is swapped with i_2’s 2nd genotype’s j^{th} variable with a probability of p_c. This procedure is performed on each pair of genotypes in each solution candidate separately.

The simple bitwise mutation is employed as a mutation tool. The genotypes of each solution candidate are inverted with a probability of pm in the bitwise mutation process.

3. Preparing Test Environment

3.1. Multidimensional Knapsack Problem

MKP is a well-known benchmark problem for optimization algorithms. The goal of the problem finding a subset of given a number of items that obtain the optimal profit value while satisfying the given constraints. In this problem, there are n items, m resources, and $n \times m$ constraints. The formulation of the problem is given in Equation 4 and Equation 5.

$$\max \quad \sum_{j=1}^{n} p_j \times x_j \quad (4)$$

$$\text{subject to } \sum_{j=1}^{n} r_{ij} \times x_j \le c_i, \quad \forall_i \in \{1,2,\ldots,m\} \quad (5)$$

where n is the number of items, m is the number of resources (knapsacks), x_j denotes the whether the item j is collected, p_j is the profit of the item j, r_{ij} is the consumption of the j^{th} item at the i^{th} resource and finally c_i is the capacity of the i^{th} resource.

In order to test BMGA for MKPs, the first five problems are taken from the six different datasets which are provided on the well-known OR-LIB[22]. There are 30 different problems in that dataset and each has a different problem size. All problems have a tightness ratio of 0.25, however, their number of items and number of resources is changing. Also, they are encoded as "cbX-Y" in the result table, which means "Chu-Beasley, dataset X, problem Y". All the necessary information is given in Table 2.

In this work, all 30 datasets are solved by BMGA. Then, the results are compared to the results obtained from [16]

3.2. Experimental Studies

For the experimental tests, first, some parameters of BMGA are set. These settings can be seen in Table 1.

Table 1 Parameter settings

Parameter	**Value**
number of generations	1000
Size of the population	100
Probability of crossover	1.0
Probability of mutation	0.03
Tournament size	4
number of genotypes	4
Population rate to form BN	0.1

Note that, each value in Table 1 is obtained by conducting a number of preliminary sensitivity tests[15].

For each test, with the same set of seeds, 30 independent runs were performed. For each run, best-of-generation (BOG) is saved and the overall performance is calculated as shown in Equation 6.

$$\bar{F}_{BOG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BOG_{ij}}\right) \tag{6}$$

where G is the number of generations, N is the number of runs, $F_{BOG_{ij}}$ is the BOG of the j^{th} run's i^{th} generation. Last, $\bar{F}_{BOG}$ is the overall offline performance.

4. Results

After setting the parameters in Table 1, the BMGA is tested for 30 different MKP datasets. For each problem, 30 independent runs are executed. The results can be seen in Table 2. Also, the charts of the first problems of each group can be seen in Figure 2.

In the table, "Optimum" is the best-known optimum value so far provided in [22], "Best" is the best solution found during each test, and "Mean" is the average performance for the particular problem. In the "Prob. size" column, the size of the problem is given. For example, "5x100-0.25" means that the problem has 5 resources, 100 items, and a tightness ratio of 0.25.

Results show that BMGA is capable to solve MKP, is competitive, and obtains better solutions in terms of #Mean and #Best for most of the problem instances because of its Bayesian probability vector. More clearly, both in terms of #Best and #Mean, BMGA has got better results in 20 of 30 problems.

Although BMGA performs better than the other algorithms, for the test instances with 500 items, it performs slightly worse than the other test instances. That is because having more items means a bigger BN. And since BOA's BN is constructed with a 1-incoming edge rule, it is getting hard to exploit relations between the variables.

5. Conclusion

In this paper, the well-known optimization problem, the MKP, is solved by a recently proposed algorithm, the BMGA.

BMGA combines the powers of EDA and MS. By saying EDA, we mean BOA and by saying MS, we mean GA. While the MS part is responsible for the optimization process, the EDA part is responsible for exploiting relations between variables. Since it is a recently proposed algorithm, it has become mandatory to solve well-known optimization benchmark problems such as MKP.

For testing, BMGA is used to solve the most popular optimization dataset library's MKP instances. Then, the results are compared to the most recent paper. To get a fair comparison, the same number of fitness value calculations were done. The results showed that BMGA outperforms even the latest proposed algorithms.

This work tells us that exploiting the relations between the problem variables is important and useful while solving global optimization problems.

Table 2 Results for the test sets

Prob. size	Prob.	Optimum	Profit	BMGA	MS	SLMS	MMVO	BGSA	BHTPSO	BHTPSO-QI
5x100-0.25	cb1-1	24381	Best	**24311**	24253	24231	24192	24152	24169	24301
			Mean	**24072**	24004	24015	24050	23835	23822	23821
	cb1-2	24274	Best	**24274**	24258	**24274**	**24274**	23986	24109	23944
			Mean	24225	23934	24145	**24274**	23536	23657	23688
	cb1-3	23551	Best	23247	**23538**	**23538**	**23538**	23386	23435	23418
			Mean	23175	23272	23440	**23520**	23041	23072	23073
	cb1-4	23534	Best	**23330**	23256	**23330**	23288	23172	23253	23192
			Mean	**23289**	23024	23156	23120	22863	22928	22923
	cb1-5	23991	Best	**23952**	23845	23947	23947	23755	23815	23774
			Mean	**23901**	23567	23800	23900	23459	23473	23527
5x250-0.25	cb2-1	59312	Best	**59203**	58084	59107	58473	57565	57814	57800
			Mean	**58980**	57369	58736	58240	56554	56874	56685
	cb2-2	61472	Best	61227	60248	**61280**	60692	60057	59982	59767
			Mean	**61185**	59386	61041	60390	58613	58588	58680
	cb2-3	62130	Best	**61831**	61212	61787	61702	59936	60630	60524
			Mean	**61684**	59922	61476	61330	58975	59234	59186
	cb2-4	59463	Best	**59167**	58386	59101	58441	57970	57736	57884
			Mean	58777	57752	**58787**	58300	56744	56773	56584
	cb2-5	58951	Best	**58753**	57755	58485	58082	56959	57378	57550
			Mean	**58566**	56929	58097	58300	55961	56129	56361
5x500-0.25	cb3-1	120148	Best	**119992**	116296	119914	119978	111206	114493	114438
			Mean	**119921**	115444	119625	119900	108930	111017	111469
	cb3-2	117879	Best	116722	113732	**117362**	115634	108522	112821	112147
			Mean	115888	112257	**116858**	115400	106631	109276	109247
	cb3-3	121131	Best	117859	117666	**120888**	119156	111271	114774	116099
			Mean	117083	116367	**120711**	118900	109430	112035	112001
	cb3-4	120804	Best	**120501**	116454	120030	119124	111283	115828	114327
			Mean	**119662**	115396	119644	118900	109062	112200	111671
	cb3-5	122319	Best	119059	117900	**121907**	121141	112391	115889	117242
			Mean	118284	116767	**121512**	120800	110564	112253	113364
10x100-0.25	cb4-1	23064	Best	**22917**	22753	22835	22805	22836	22905	22876
			Mean	**22694**	22459	22604	22700	22334	22425	22449
	cb4-2	22801	Best	**22836**	22611	22650	22630	22441	22573	22408
			Mean	**22541**	22255	22432	22480	21991	22047	22017
	cb4-3	22131	Best	**22012**	21886	21962	22131	21849	21797	21949
			Mean	**21662**	21466	21632	21720	21313	21342	21461
	cb4-4	22772	Best	22420	22319	**22463**	22347	22325	22418	22376
			Mean	**22391**	21992	22233	22160	21961	22037	22029
	cb4-5	22751	Best	22312	22440	**22619**	22417	22168	22215	22254
			Mean	22182	22132	**22279**	22290	21840	21822	21903
10x250-0.25	cb5-1	59187	Best	**58820**	57757	58725	58476	56928	57530	57036
			Mean	**58812**	56708	58148	58310	55759	55854	55960
	cb5-2	58781	Best	**58339**	57363	58321	57937	56337	56568	56490
			Mean	**58267**	56793	58074	57790	55455	55443	55708
	cb5-3	58097	Best	**57804**	56690	57764	57062	55573	56426	55982
			Mean	**57626**	56024	57372	56960	54638	54793	54727
	cb5-4	61000	Best	**60597**	59930	**60597**	60326	58595	59030	59077
			Mean	**60460**	58934	60194	60030	57766	58057	57721
	cb5-5	58092	Best	**57567**	56863	57233	56276	56186	56217	56204
			Mean	**57559**	56066	56961	56060	54850	54941	54872
10x500-0.25	cb6-1	117821	Best	**117371**	113362	117287	–	108487	110996	111669
			Mean	**116882**	112541	116830	–	105760	107698	108367
	cb6-2	119249	Best	118730	115022	**118737**	–	109569	114262	113001
			Mean	118250	114250	**118385**	–	106775	108648	109197
	cb6-3	119215	Best	**118905**	115419	118488	–	109705	113987	112419
			Mean	**118402**	114372	118003	–	106853	108576	109004
	cb6-4	118829	Best	117354	115038	**118116**	–	108628	112476	112198
			Mean	116593	113444	**117714**	–	105679	107692	107796
	cb6-5	116530	Best	115334	112971	**116530**	–	106972	109567	109287
			Mean	114447	111707	**115301**	–	104509	106217	106212

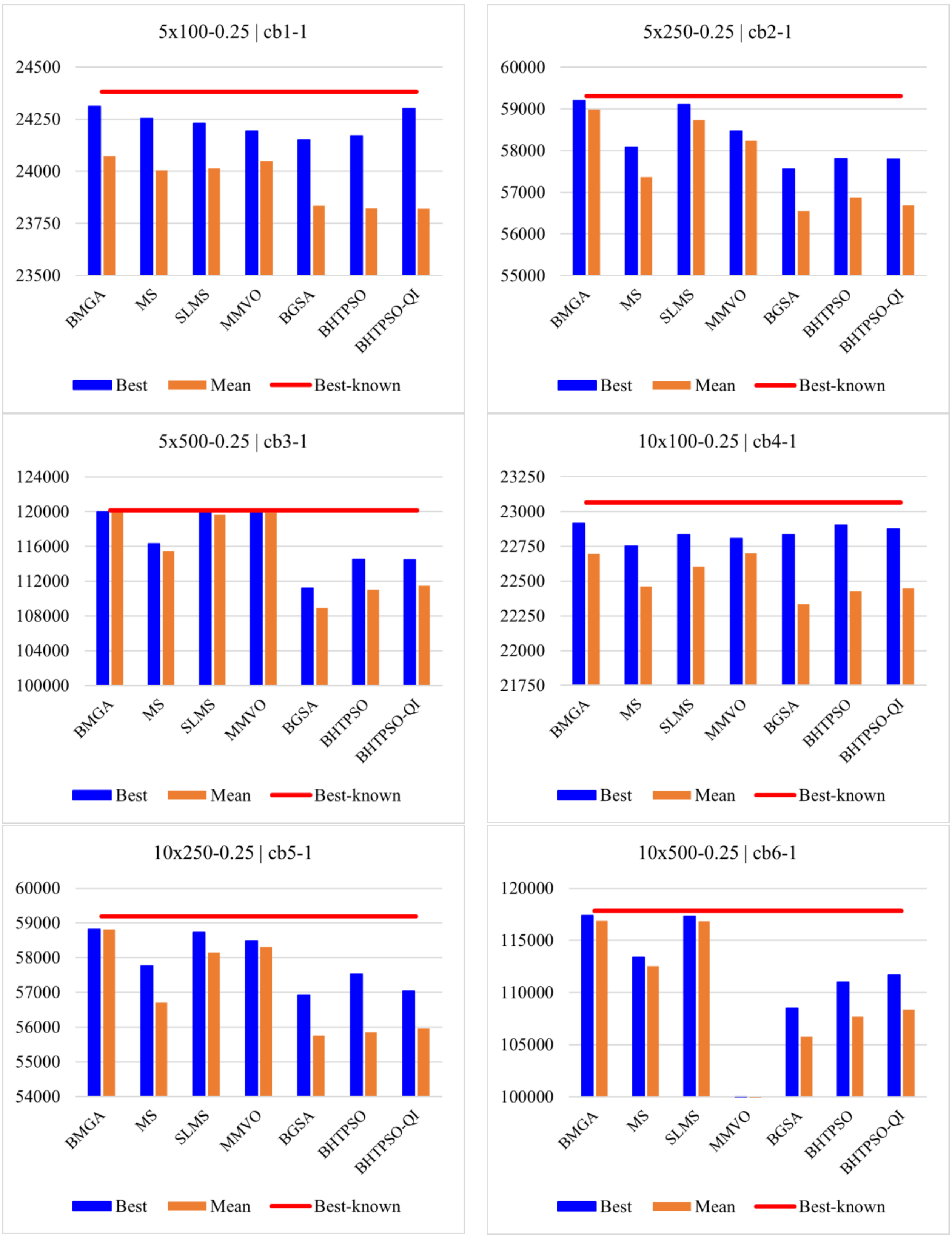

Figure 2 Results for the first dataset of each group of problem size

References

[1] P. C. Gilmore and R. E. Gomory, “The Theory and Computation of Knapsack Functions”, *Operations Research*, c. 14, sy 6, ss. 1045-1074, Ara. 1966, doi: 10.1287/opre.14.6.1045.

[2] Y. Li, Y. Tao, and F. Wang, “A compromised large-scale neighborhood search heuristic for capacitated air cargo loading planning”, *European Journal of Operational Research*, c. 199, sy 2, ss. 553-560, Ara. 2009, doi: 10.1016/j.ejor.2008.11.033.

[3] G. B, “Allocation of Databases and Processors in a Distributed Computing System”, *Management of Distributed Data Processing*, ss. 215-231, 1982.

[4] M. Engwall and A. Jerbrant, “The resource allocation syndrome: the prime challenge of multi-project management?”, *International Journal of Project Management*, c. 21, sy 6, ss. 403-409, Ağu. 2003, doi: 10.1016/S0263-7863(02)00113-8.

[5] M. Abdel-Basset, R. Mohamed, K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, “BSMA: A novel metaheuristic algorithm for multi-dimensional knapsack problems: Method and comprehensive analysis”, *Computers & Industrial Engineering*, c. 159, s. 107469, Eyl. 2021, doi: 10.1016/j.cie.2021.107469.

[6] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, “Knapsack problems — An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems”, *Computers & Operations Research*, c. 143, s. 105693, Tem. 2022, doi: 10.1016/j.cor.2021.105693.

[7] S. Gupta, R. Su, and S. Singh, “Diversified sine–cosine algorithm based on differential evolution for multidimensional knapsack problem”, *Applied Soft Computing*, c. 130, s. 109682, Kas. 2022, doi: 10.1016/j.asoc.2022.109682.

[8] A. Rezoug, M. Bader-el-den, and D. Boughaci, “Application of Supervised Machine Learning Methods on the Multidimensional Knapsack Problem”, *Neural Process Lett*, c. 54, sy 2, ss. 871-890, Nis. 2022, doi: 10.1007/s11063-021-10662-z.

[9] F. Wang, Y. Li, A. Zhou, and K. Tang, “An Estimation of Distribution Algorithm for Mixed-Variable Newsvendor Problems”, *IEEE Transactions on Evolutionary Computation*, c. 24, sy 3, ss. 479-493, Haz. 2020, doi: 10.1109/TEVC.2019.2932624.

[10] L. Ma, K. Hu, Y. Zhu, and H. Chen, “Cooperative artificial bee colony algorithm for multi-objective RFID network planning”, *Journal of Network and Computer Applications*, c. 42, ss. 143-162, Haz. 2014, doi: 10.1016/j.jnca.2014.02.012.

[11] B. Zhang, Q.-K. Pan, X.-L. Zhang, and P.-Y. Duan, “An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems”, *Applied Soft Computing*, c. 29, ss. 288-297, Nis. 2015, doi: 10.1016/j.asoc.2015.01.022.

[12] M. Dorigo and T. Stützle, “Ant Colony Optimization: Overview and Recent Advances”, içinde *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Ed. Cham: Springer International Publishing, 2019, ss. 311-351. doi: 10.1007/978-3-319-91086-4_10.

[13] S. Mirjalili and A. Lewis, “The Whale Optimization Algorithm”, *Advances in Engineering Software*, c. 95, ss. 51-67, May. 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[14] Y. Zhou, L. Li, and M. Ma, “A Complex-valued Encoding Bat Algorithm for Solving 0–1 Knapsack Problem”, *Neural Process Lett*, c. 44, sy 2, ss. 407-430, Eki. 2016, doi: 10.1007/s11063-015-9465-y.

[15] E. Gazioğlu and A. S. Etaner-Uyar, “Experimental analysis of a statistical multiploid genetic algorithm for dynamic environments”, *Engineering Science and Technology, an International Journal*, c. 35, s. 101173, Kas. 2022, doi: 10.1016/j.jestch.2022.101173.

[16] Y. Feng and G.-G. Wang, “A binary moth search algorithm based on self-learning for multidimensional knapsack problems”, *Future Generation Computer Systems*, c. 126, ss. 48-64, Oca. 2022, doi: 10.1016/j.future.2021.07.033.

[17] G.-G. Wang, “Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems”, *Memetic Comp.*, c. 10, sy 2, ss. 151-164, Haz. 2018, doi: 10.1007/s12293-016-0212-3.

[18] M. Abdel-Basset, D. El-Shahat, H. Faris, and S. Mirjalili, “A binary multi-verse optimizer for 0-1 multidimensional knapsack problems with application in interactive multimedia systems”, *Computers & Industrial Engineering*, c. 132, ss. 187-206, Haz. 2019, doi: 10.1016/j.cie.2019.04.025.

[19] Z. Beheshti, S. M. Shamsuddin, and S. Hasan, “Memetic binary particle swarm optimization for discrete optimization problems”, *Information Sciences*, c. 299, ss. 58-84, Nis. 2015, doi: 10.1016/j.ins.2014.12.016.

[20] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization”, *Swarm Intell*, c. 1, sy 1, ss. 33-57, Haz. 2007, doi: 10.1007/s11721-007-0002-0.

[21] M. Pelikan, “Bayesian Optimization Algorithm”, içinde *Hierarchical Bayesian Optimization Algorithm: Toward a new Generation of Evolutionary Algorithms*, M. Pelikan, Ed. Berlin, Heidelberg: Springer, 2005, ss. 31-48. doi: 10.1007/978-3-540-32373-0_3.

[22] J. E. Beasley, “OR-LIB http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/”, 2005.