2012 International Conference on System Engineering and Modeling (ICSEM 2012)
IPCSIT vol. 34 (2012) © (2012) IACSIT Press, Singapore

A Binary Model on the Basis of Imperialist Competitive Algorithm in Order to Solve the Problem of Knapsack 1-0

Shirin Nozarian[*a], Hodais Soltanpoor[a], Majid Vafaei Jahan[b]

[a] Young Researchers Club, Mashhad Branch, Islamic Azad University, Mashhad, Iran
[b] Islamic Azad University,Mashhad Branch,Iran.
[*]snozarian@yahoo.com

Abstract. Though imperialist competitive algorithm is repeatedly utilized to solve different problems, its discrete binary model has been ignored. A discrete method is offered in this article according to Imperialist competitive algorithm to solve the problem of knapsack 1-0. Resulted outcomes of utilizing this method represent the ability of this algorithm in solving such problems. In order to evaluate the efficiency of it, this method is compared with some heuristic algorithms such as Genetic algorithm, Particles swarm optimization, Ant colony system, Tabu search, and simulated annealing. Discrete binary Imperialist competitive algorithm offered better suggestive responses in all of mentioned tests. Results cleared that this algorithm performs around 30% better than other compared methods.

Keywords: Discrete Problems, Knapsack Problem, Imperialist Competitive Algorithm, Genetic Algorithm, Mimetic Algorithm, Particles Swarm Optimization, Ant Colony System, Tabu Search, Simulated Annealing

1. Introduction

Imperialist competitive algorithm (short of “ICA”) was introduced by Atashpaz et al. in [1, 2] in 2007. Then it was utilized for optimum design of skeletal structures by Kaveh et al. in [3]. ICA is an initial collection of possible responses, exactly the same as other algorithms in this group, these initial responses are known as "chromosome" in the genetic algorithm (short of “GA”), as "particle" in Particles swarm optimization (short of “PSO”), and as "country" in ICA one. ICA gradually improves these initial responses (countries) –in a special process that is explained in [1]- and finally responds to the problem of optimization (favorite country) properly [1]. A discrete binary method is offered in this article for mentioned algorithm which will be explained in following parts.

On the other hand, GA was introduced by Holland in early 1980s[4]. This method solved most of optimization problems, especially binary ones, successfully. Though GA performed well in solving optimization problems, its efficiency is less than local searches as it is explained in [5]. PSO performed well similarly in solving optimization problems. Generally movement of particle swarms is an efficient technique to solve the optimization problems. It works according to the probability rules and population. The binary sample of this method was introduced by Kennedy and Eberhart in [6]. Kennedy and Spears engaged in comparing GA and PSO in [7]. Results of this research showed that PSO performs more flexible than GA and is faster than it in convergence to the response. Bak and Snappen offered a model in 1993 which was the basis of a new method, called Extermal optimization (short of “EO”). This model is according to Bak-Sneppen model [8]. On the contrary to GA that solves the problems by applying a population of possible responses, EO looks for the response only by mutating of a solution [9]. The initial model of this method is offered in [10]. The mutation operator in this model was performed on the worst parts only. This method was able to produce just approximate optimization responses, and finding the main response was hardly probable in it. Hence in later studies, Boettcher and Percus defined the probability of mutation for each component. This increased the efficiency of the model greatly [11].

Moser and Hendtlass compared EO and Ant colony optimization (short of "ACO") in [9]. Their evaluations showed that EO performs faster and more flexible than ACO in convergence to the response, but it was fluctuating in response. It is hardly probable to achieve a global optimum by this pattern.

Tabu search (short of "TS") is another method which is compared in this article. This method was introduced by Glover and Laguna in 1997. They proved its efficiency in [12] for solving different problems. But due to its need to a big memory, methods of optimizing the memory had to be added to it. It is explained in [13] that if TS executes properly, it will be able to find a global optimum. But finding no optimal response is probable in this method, so an exact time of complication cannot be defined for it.

Simulated annealing (short of "SA") is another algorithm which is compared in this research. A new viewpoint was introduced by Metropolis et al. in 1953 to solve the problem of compound optimization, this method was called SA. Fundamentally, their algorithm is a kind of simulation in a system whose temperature is constantly decreasing until it gets to a stable (freezing) condition [14]. Kirkpatrick et al. and Cerny used this idea in [15, 16] to solve the compound problems like "Traveling salesman". Thereafter SA is used to solve different problems, including knapsack 0-1. Incapability in escaping from local optimum is the most important short come in this method.

An extensive definition of the knapsack problem is presented in the second part of the article. Explaining the ICA is offered in third, suggested discrete binary model is offered in fourth, tests and comparing the algorithms to solve the mentioned problem is offered in fifth part of this article. Finally, some results and suggestions for future works are added to section six.

2. An Explanatory Definition Of The Problem

Problem of Knapsack 1-0:

A knapsack and some packs are supposed to be in this problem. Every pack possesses a specific weight and value. On the other hand the knapsack has a specific volume, too. It means that definite number of packs can be put in it. We want to put the packs into the knapsack in a special way to have the maximum value of them [17].

Mathematical Formulation of the Problem:

Maximize:

$$\sum_{i=1}^{n} v_i x_i \quad , \quad x_i \in \{0,1\} \tag{1}$$

In order to:

$$\sum_{i=1}^{n} w_i x_i \leq W \, , \quad x_i \in \{0,1\} \tag{2}$$

It should be mentioned that the problem of knapsack 1-0 is an NP-Complete Problem [18]. According to the algorithm which is used to find an exact solution for this problem, its time consequence is expressed as follow:

$$T(n) = \theta(2^n) \tag{3}$$

Solving this problem would be hard(and even impossible) at the polynomial time, when the number of (n)s is getting big.

3. Imperialist Competitive Algorithm

Like other evolutionary optimization methods, this method begins with some initial population. Every element of the population is called a *country* in this algorithm. Countries are classified in two different groups: *Colony* and *Imperialist*. Every colonialist brings some colonies under its dominion and controls them according to its power [1]. Assimilation policy and imperialist competition is the core of this algorithm. This policy is accomplished by moving the colonies towards empires on the basis of a determined connection which is defined completely in [17].

Generally, imperialist competitive algorithm is unlimitedly used to call every continues optimization problem. Therefore this algorithm is used easily in different fields, such as electrical engineering, mechanics, industries, management, civic developments, artificial intelligence, etc.

4. Discrete Binary Imperialist Competitive Algorithm (Db-Ica)

However imperialist competitive algorithm was introduced to solve the problem of optimization, its binary model is ignored for solving the discrete problems. Two different groups of parameters can be classified in imperialist competitive algorithm:

Independent Constructional operators:

These operators do not depend on the kind of coding in the problem. They usually have stable and fixed construction. For example, all decisions are made about the costs of countries and empires in imperialist competition part and kind of their coding or kind of performing the countries are not interfered in the process.

Dependent operators:

Such operators should be designed dependent on the problem. For example, the operator of assimilation in a connected problem is as follow: the colony moves toward the imperialist in a specific trend and specific vector. The colony approaches to the imperialist by this movement in a Euclidean way. But the Euclidean approach in a discrete problem does not mean approach necessarily. Euclidean approach is absolutely meaningless in some samples. Only operators and related parts to the problem should change in a imperialist competitive algorithm to solve a discrete problem. These related parts are as follow: 1-creating the initial country, 2-the assimilation operator, and 3-the revolution operator. Created changes in creating the initial countries depend on the kind of discrete problem. Discrete problems can be classified in three different groups: 1-discrete problems with continues nature, 2-permutation discrete problems and 3-binary discrete problems. There is no need to big changes in the algorithm in discrete problems with connected nature. But several functions should change in two other groups. The binary knapsack problem classifies in third group.
In this condition, the function which creates initial countries should be defined in a special way, in that created initial countries are in conformity with the favorite coding in the discrete problem. These codes are arranged by 0s and 1s in the problem of knapsack 1-0 according to the number of objects just like a Chromosome in Genetic Algorithm.

In this article, the crossover is used as an assimilation operator, exactly the same as genetic algorithm to define a new model for a discrete imperialist competitive algorithm. The best way to choose a revolution operator is applying the creating function of initial countries. If this operator get defined properly, it will be a proper substitute for the revolution operator for changing the continues condition to the discrete condition. Generally, new random changes can be defined and different revolution operators can be created in the countries according to the defined problems. The most important point is that due to such a revolution, the new country which is created by this operator should differ from the initial country extremely, randomly and unexpectedly (but not necessarily unexpectedly). The mutation operator is used in this article, the same operator which is used in genetic algorithm.

Following algorithm is used to execute the assimilation operator:

1- Two genes are chosen randomly from the empire genes.
2- The existing amounts between these genes of the empire are replaced by the genes of colonies at the same place.
3- First and second levels repeat for all colonies of all empires.

Tab.1 A sample on assimilation in DB-ICA

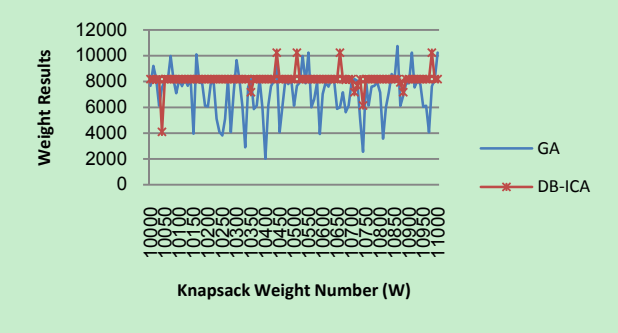
Imperialist	1	*0*	1	*1*	1	1
Colony 1	0	1	0	1	0	0
Changed Colony 1	0	*0*	*1*	*1*	0	0

5. Simulation Results

Results of the tests are offered in this part. Mentioned algorithms are executed in a system with 1.67 GHz processor and 1G of RAM memory, in MATLAB programming software. Utilized data in the problem are in concordance with the same data in [20]. The number of objects equals 16 in these data (n = 16), weight of each object(i = 0, 1, …, 15): as $w_i = 2^i$. The value of all objects equals 1 ($v_i = 1$). The optimum number (W) in this problem was arranged between 10000 and 11000. The optimal response in this problem is: W = 8191. It should be mentioned that changing the weight of the knapsack in the determined period of time does not change the optimum response. It only makes the condition more complicated for its finding. All algorithms are executed at the same condition in 50 generations, and the best created amounts by them are

compared with different optimization algorithms in discrete binary imperialist competitive algorithm, the results are depicted in figures 1 to 5.

Superiority of DB-ICA and its stability comparing with other optimization algorithms can be seen in all figures.



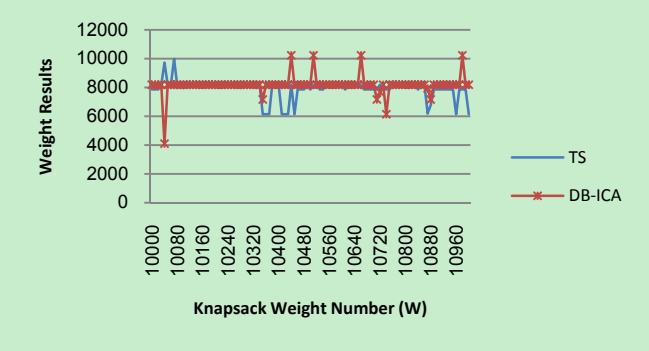
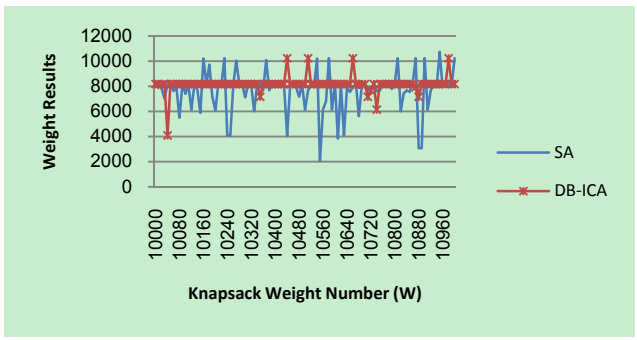


Fig. 1: Results of solving the knapsack 1-0 problem with GA and DB-ICA after 50 generations. GA is fluctuating; however DB-ICA is stable relatively.

Fig. 2: Results of solving the knapsack 1-0 problem with TS algorithm and DB ICA after 50 generations. Though TS is relatively stable, like DB-ICA, this stability is on a specific response near the global optimum, But DB-ICA is stable on the main response.



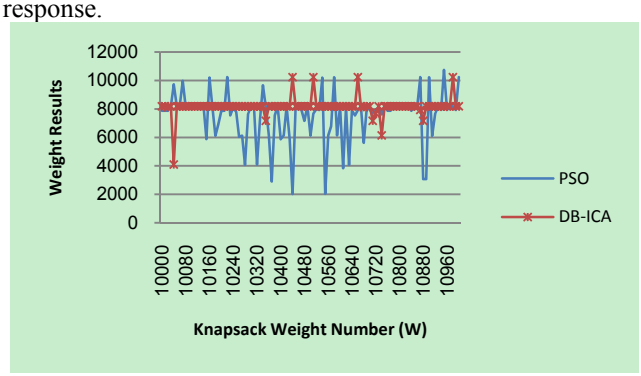
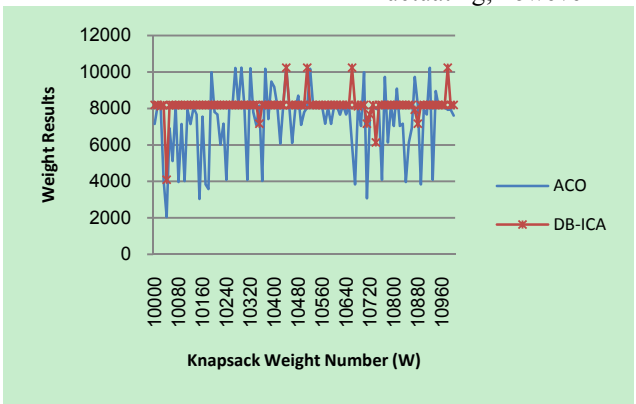


Fig. 3: Results of solving knapsack 1-0 problem with SA and DB-ICA after 50 generations. SA is extremely fluctuating; however DB-ICA is stable relatively.

Fig. 4: Results of solving the knapsack 1-0 problem with PSO and DB-ICA after 50 generations. PSO is extremely fluctuating; however DB-ICA is stable relatively.



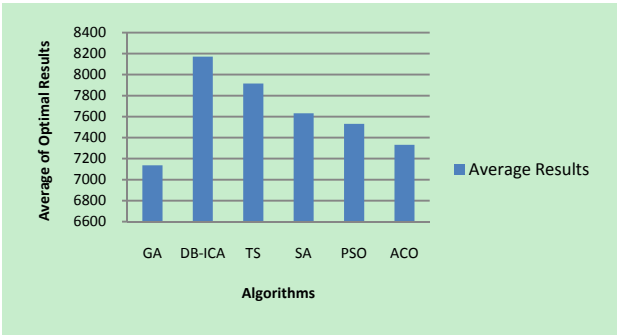


Fig. 5: Results of solving the knapsack 1-0 problem with ACO and DB-ICA after 50 generations. ACO is extremely fluctuating; however DB-ICA is stable relatively.

Fig. 6: Average of the best optimal results after 50 repetition cycles in all algorithms. DB-ICA got the highest average.

The average of optimals for each algorithm in 50 repetition cycles is depicted in figure 6. As you can see, DB-ICA possesses has the nearest average in approaching to optimal response.

A scale is introduced in [18] to evaluate the level of efficiency in optimization algorithms in order to solve the problem of knapsack 1-0, called *Hit Rate*. This scale calculates the level of ability in approaching the pattern to the optimal response, regarding the domain at the size of a *Hit Scale*. For example, if global optimization equals 100, and *Hit Scale* equals 0.3, then if the algorithm response equals a number between 70 to 130, it will be claimed that the response is hit, it means the *Hit Rate* equals 1. Consider the *Hit Scale*

equals a number between 0.002 and 0.03, so the level of algorithms ability in approaching to global optimum is being compared. DB-ICA possesses the nearest level to 1, as it can be seen in figure 7. Therefore this algorithm is certainly able to approach the optimal response.

6. Results And Future Works

Firstly, a review of heuristic methods was accomplished in this article. Then a new method of imperialist competitive algorithm was offered to solve the binary discrete problems. Regarding the accomplished executions, it can be deduced that the offered binary model in algorithm ICA performs better than some other well-known heuristic algorithms. According to the Hit-Rate, DB-ICA performed 30% better than the best pattern in evaluating the level of efficiency. One advantage of suggested algorithm is the novelty of its idea as one of the first discrete algorithms on the basis of a social-political process, and also its ability in reaching the global optimum, comparing with different optimization algorithms faced the problem of knapsack 1-0.

Suggested pattern in this article has only focused on a sample of binary problem in evaluated discrete problems. It suggests that mentioned method can be utilized for other non-binary discrete problems in future articles. No comparison is done in the level of processor usage in this article, regarding the weakness of imperialist competitive algorithm in this field, optimizing the usage of the processor can be searched in discrete models in future researches as well.

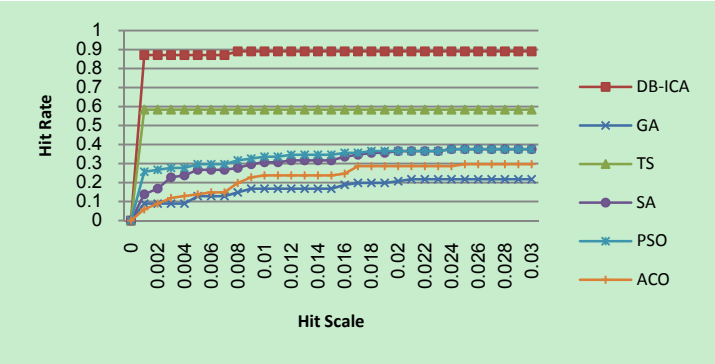


Fig. 7: The level of ***Hit Rate*** in all tested algorithms. DB-ICA got the highest Hit Rate.

7. References

[1] Atashpaz-Gargari, E. and C. Lucas. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. 2007: IEEE.

[2] Atashpaz-Gargari, E. and C. Lucas. Designing an optimal PID controller using Colonial Competitive Algorithm. 2007.

[3] Kaveh, A. and S. Talatahari, Optimum design of skeletal structures using imperialist competitive algorithm. Computers & Structures. 88(21-22): p. 1220-1229.

[4] Holland, J.H., Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1992: The MIT press.

[5] Ishibuchi, H., T. Murata, and S. Tomioka. Effectiveness of genetic local search algorithms. 1997.

[6] Kennedy, J. and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. 1997: IEEE.

[7] Kennedy, J. and W.M. Spears. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. 1998: IEEE.

[8] Bak, P. and K. Sneppen, Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters, 1993. 71(24): p. 4083-4086.

[9] Moser, I. and T. Hendtlass. Solving problems with hidden dynamics-comparison of extremal optimization and ant colony system. 2006.

[10] Boettcher, S. and A.G. Percus, Optimization with extremal dynamics. Physical Review Letters, 2001. 86(23): p. 5211-5214.

[11] Boettcher, S. and A. Percus, Nature's way of optimizing. Artificial Intelligence, 2000. 119(1-2): p. 275-286.

[12] Glover, F. and M. Laguna, Tabu search. Vol. 1. 1998: Kluwer Academic Pub.

[13] Andersson, P., Using Tabu search to solve the 0-1 Knapsack. 2004.

[14] Metropolis, N., et al., Equation of state calculations by fast computing machines. The journal of chemical physics, 1953. 21(6): p. 1087.

[15] Cerny, V., Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications, 1985. 45(1): p. 41-51.

[16] Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, Optimization by simulated annealing. science, 1983. 220(4598): p. 671.

[17] Atashpaz-Gargari, E., Expanding the Social Optimization Algorithm and Evaluating its Efficiency. Computer and Electerical Engineeing, 2008. Masters.

[18] Wang, G., C. Chen, and K. Szeto, Accelerated genetic algorithms with Markov Chains. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010): p. 245-254.