2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)
Dec. 22nd, 2017
(Iran University of Science and Technology) – Tehran, Iran

A New Approach to the Restart Genetic Algorithm to Solve Zero-One Knapsack Problem

Mojtaba Montazeri
Department of Engineering
Larestan Branch, Islamic Azad University
Larestan, Iran
mojtabamontazeri2017@yahoo.com
+98-9107002915

Rasoul Kiani
Department of Engineering
Fasa Branch, Islamic Azad University
Fasa, Iran
rasoul.kiani87@yahoo.com
+98-9179325705

Seyed Saleh Rastkhadiv
Department of Engineering
Larestan Branch, Islamic Azad University
Larestan, Iran
rastkhadiv@gmail.com
+98-9171814374

Abstract—**Genetic Algorithm is one of the popular evolutionary algorithms to solve different problems specially NP problems. Therefore, there are different efforts to improve the speed and performance of genetic algorithm. The main purpose of this paper is to present a new kind of Restart Genetic Algorithm to improve the performance and reduce the run time. The proposed algorithm has been used to solve zero-one knapsack problem. The experimental results show the proposed algorithm has enhanced speed and performance in comparison with other two kinds of genetic algorithm.**

Keywords—Genetic Algorithm (GA); Knapsack Problem; Dynamic Mutation Rate; Island Genetic Algorithm; Restart Genetic Algorithm

I. Introduction

Genetic Algorithm is one of the interesting and useful technics used to solve different problems[1],[2]. Two of the main problems in Genetic Algorithm are time complexity and local optimums[1],[3]. The former shows the time required to run the genetic algorithm and get the results. The latter are some points of search space that the genetic algorithm thinks they are the best results and can not to escape and find the global one. In fact, local optimums prevent genetic algorithm to converge to the best answer.

There have been different efforts to overcome the mentioned problems. Adaptive mutation technics proposed in [4], [5] are very useful technics to get out from local optimums. In this method, when the algorithm cannot find better results, it tries to increase mutation rate to change the area of search space. Solving the local optimums problem may be helpful to reduce the time complexity, too. However there are some other efforts like Hybrid Genetic Algorithm proposed in [6], [7] to improve the speed and precision of the algorithm. This technic integrates greedy algorithm and genetic algorithm to form a hybrid genetic algorithm.

This paper proposes a new genetic algorithm to improve the speed and performance. The technic has been used to solve zero-one knapsack problem. Section II explains zero-one knapsack problem and section III presents a traditional genetic algorithm to solve it. Section IV proposes a new kind of using genetic algorithm and section V indicates experimental results of proposed algorithm and two other kind of genetic algorithm in solving zero-one knapsack problem. Finally, the conclusion is presented in section IV.

II. Zero-One Knapsack Problem

In knapsack problem, there are numbers of items and a knapsack. Each item has a weight and a profit. The knapsack has a limited capacity and endures up to a certain weight. The goal is to select some items for knapsack that can make the maximum profit where the total weight of the selected items is less than or equal to the knapsack capacity. In zero-one knapsack problem, unlike fractional knapsack, it is impossible to pick up a fraction of an item. It means, each item should be selected or not.

Let there are N items available and w_i and p_i showing the weight and profit of item i, respectively. The aim is presented in (1):

$$\text{Maximum}\ (\sum_{i=1}^{N} s_i p_i)\ ,\ \text{while}\ (\sum_{i=1}^{N} s_i\, w_i) \leq C \qquad (1)$$

Where C is the capacity of knapsack and S_i is a boolean variable which indicates that item i is selected or not. If S_i is equal to 1, so item i has been selected and vice versa. The zero-one knapsack problem is a combinatorial optimization and a NP-complete problem. So, finding the best and the exact solution for large input is practically impossible, merely, stochastic algorithm can be helpful. Section V introduces different approaches to solve zero-one knapsack and compares them.

III. Genetic Algorithm

Genetic Algorithm (GA) is a searching method based on natural genetic operations [8], [9]. A genetic algorithm begins with a set of random solutions (chromosomes in natural) called generation. After implementing genetic operations over the initial generation, a new generation will be generated. In this procedure, better solutions have more chances to contribute in genetic processes. So better solutions are expected in generated generation. This process will be repeated till satisfying some conditions or up to certain number of cycles.

As mentioned in section II, *S* is a boolean array which shows a solution for the knapsack problem. Here, *S* is a chromosome and, a set of array *S* randomly should be initialized as the first generation. Then GA starts working. A simple pseudo code for the traditional GA as formulated in below:

```
GA (M, maxCycle)
1. Generate M chromosome (s) for first population
2. Cycle = 1
3. Condition = true
4. While cycle <= maxCycle and condition = true
        a. Implement crossover operation on random
           selected chromosomes
        b. Implement mutation operation on random
           selected chromosomes
        c. Select M chromosomes from generated ones
           to make the new generation
        d. Cycle = cycle + 1
5. Return the best result of current generation
```

Where *M* is the number of chromosomes or size of generation and *maxCycle* is the most iteration of GA, both parameters are set manually [8]. With condition parameter, algorithm is terminated when iteration (*cycle*) is less than *maxCycle*. For example the condition can be satisfied when 95% of generations are same as the best solution.

IV. Proposed Algorithm

Restart-base Genetic Algorithm (RGA) and Island Genetic Algorithm (IGA) are two technics to improve GA performance [10]. RGA randomly generates a new generation and restart the algorithm when it cannot find better results. Of course, the best answer of each generation is transferred to next iteration. IGA generates *k* separate generations and perform the *k* GA in a parallel operation. However, the parallel algorithms share the best answers together.

The proposed GA (PGA) in this paper uses *M* parallel GAs (sub-GAs) to produce *M* chromosomes of a superior generation. Each parallel algorithm has a tiny size of population and will be continued until best answer cannot be improved. Because of the tiny population, the sub-GAs are stopped in local optimums quickly. However after running the *M* GAs, the best result of each one is used as a chromosome. In fact, the superior population is the collection of local optimums which prepare initial generation for the main GA (main-GA) to obtain the final results. A pseudo code of the PGA is presented in below:

```
PGA (M, subM, maxCycle, subCycle)
1. For i = 1 to M
        a. Chromosome C = GA (subM, subCycle)
        b. Add chromosome C to the superior generation
2. Implement GA iteration over superior generation
```

Where *M* and *maxCycle* indicate the generation size and, respectively, max iteration, of the main GA implemented over superior generation. Similarly, *sub-M* and *subCycle* indicate the generation size and max iteration of the sub-GA generating the superior generation. To improve the performance of PGA, dynamic mutation rate is used [5]. The technic can have efficiency in both main-GA and sub-GA in the proposed algorithm.

V. Expremental Results

The PGA was implemented over some complex zero-one knapsack problem with extensive inputs. One of the experimental results has shown in *Fig. 1*. The algorithm, as mentioned, includes two parts. In first part, some sub-GAs generate the superior generation, which shown by square shape points in *Fig. 1*. In second part, the main-GA uses superior generation to find final results. The '*' shape points in *Fig. 1*, shows the developing process of main-GA to get better results and the 'big star' shape point shows the best result.

The PGA, also, has been compared with two other kinds of GA. Table I, shows the results of the PGA in comparison with traditional GA and GA with dynamic mutation rate over four complex zero-one knapsack problems. Respectively, the number of items in *p1, p2*, *p3* and *p4* are 15, 15, 25 and 24. Each type of GAs has been performed for 20 times on each dataset and the best and the average of answers have been recorded. These features indicate the PGA has improved the performance and probability of convergence towards the best answer. All tests were done on a same PC and the real run times have been recorded that it shows the improvement of speed.

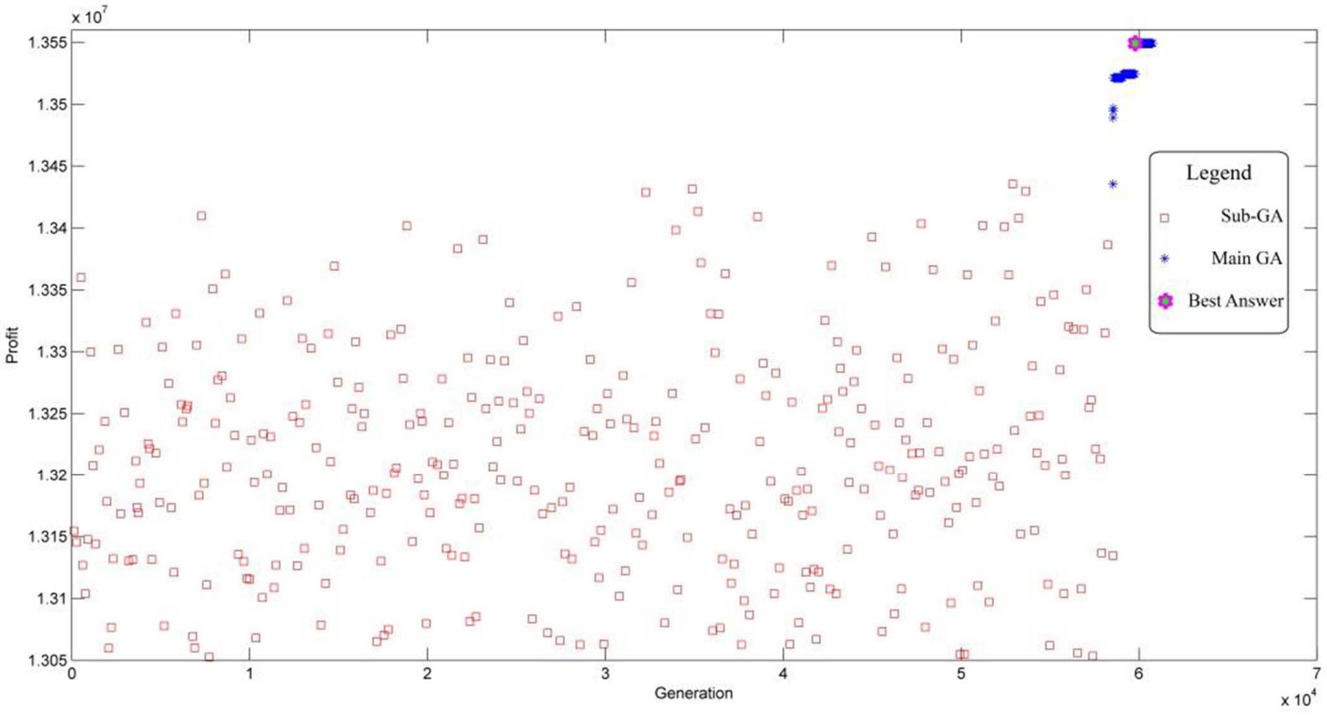

Fig. 1. Chromosomes of sub-GA and main-GA

TABLE I. THE COMPARISON BETWEEN THREE TYPES OF GAS

Criteria	Traditional GA				GA with Dynamic Mutation Rate				Proposed GA			
	P1	*P2*	*P3*	*P4*	*P1*	*P2*	*P3*	*P4*	*P1*	*P2*	*P3*	*P4*
Average Profit	987.3	1455	1416	13489249	1038	1457	1463	13532312	1039	1458	1463	13540673
Maximum Profit	1039	1458	1460	13524340	1039	1458	1463	13549094	1039	1458	1463	13549094
Real Run Time	0.61	1.89	13.46	70.32	0.85	2.61	20.63	110.51	0.79	1.80	11.8	70.32

VI. COCLUSION

The proposed algorithm tries to search all parts of search space by restarting the GA with new random population. The new technic helps the GA for the purpose of covering the whole part of the search space, because a superior generation included the best answers of different parts of search space. And the main-GA tries to enhance the superior generation.

In addition, contrary to a lot of generation produced by the proposed algorithm, the run time is reduced, because of tiny generation size of sub-GA.

REFERENCES

[1] P. G. Tharanipriya and P. Vishnuraja, "Hybrid genetic algorithm for solving knapsack problem," in *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, 2013, pp. 416–420.

[2] J. Chen and C. Zhang, "Efficient clustering method based on rough set and genetic algorithm," *Procedia Eng.*, vol. 15, pp. 1498–1503, 2011.

[3] D. Zou, L. Gao, S. Li, and J. Wu, "Solving 0–1 knapsack problem by a novel global harmony search algorithm," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1556–1564, 2011.

[4] J. Teo, "Self-adaptive mutation for enhancing evolutionary search in real-coded genetic algorithms," in *Computing & Informatics, 2006. ICOCI'06. International Conference on*, 2006, pp. 1–6.

[5] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst. Man Cybern.*, vol. 24, no. 4, pp. 656–667, 1994.

[6] Q. Chen and Y. Shao, “A hybrid genetic algorithm to solve zero-one knapsack problem,” *Appl. Inform. Commun.*, pp. 315–320, 2011.
[7] Z. Ren and Y. San, “A Hybrid Optimized Algorithm Based on Simplex Method and Genetic Algorithm,” in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, 2006, vol. 1, pp. 3547–3551.
[8] R. S. Jonathan, N. Peter, D. Ernest, R. S. Jonathan, and R. S. Jonathan, *Artificial Intelligence: A Modern Approach. Vol. 2*. Prentice hall Englewood Cliffs:, 2010.
[9] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*. Springer Science & Business Media, 2007.
[10] V. S. Gordon and D. Whitley, “Serial and parallel genetic algorithms as function optimizers,” in *ICGA*, 1993, pp. 177–183.