IEICE Proceeding Series

Recycling Lethal Chromosomes Based on Immune Operation in Genetic Algorithm for Multi-Knapsack Problem

Jing Guo, Jousuke Kuroiwa, Hisakazu Ogura, Izumi Suwa, Haruhiko Shirai, Tomohiro Odaka

Vol. 1 pp. 219-222
Publication Date: 2014/03/17
Online ISSN: 2188-5079

2012 International Symposium on Nonlinear Theory and its Applications
NOLTA2012, Palma, Majorca, Spain, October 22-26, 2012

Recycling Lethal Chromosomes Based on Immune Operation in Genetic Algorithm for Multi-Knapsack Problem

Jing Guo†, Jousuke Kuroiwa†, Hisakazu Ogura†, Izumi Suwa†, Haruhiko Shirai‡ and Tomohiro Odaka†

†Graduate School of Engineering, University of Fukui
‡Faculty of Engineering, University of Fukui
3-9-1 Bunkyo, Fukui, 910-8507 Japan
Email: {jguo, jou, ogura, suwa, shirai, odaka}@ci-lab.jp

Abstract—In order to enhance GA's performance in solving constrained optimization problems, we proposed a recycling method of lethal chromosomes (LCs) in GA with a double island algorithm model based on an immune operation. The method revives LCs by applying vaccines abstracted from LCs. We apply our method into a multi-knapsack problem (MKP), which corresponds to a classic combinatorial problem. The exhaustive simulation results indicate that our method acts a superior performance of finding optimal solutions.

1. Introduction

Genetic Algorithm (GA), pioneered by Professor Holland in the late 1960s, has been widely used in combinatorial optimization for the global search and robustness. It reflects the natural selection and survival of the fittest in evolutionary process. In general, GAs usually contain three basic steps, which are respectively crossover operator, mutation operator, and selection operator. According to these operators, it can improve the adaptability of each individual by learning the law of biological evolution. The framework of GAs as mentioned above is referred as a simple GA (SGA) in this paper. GAs have global searching capability, greater robustness and parallelism. Since GAs have these advantages, they always outperform classical optimization methods so much that they have been used to solve various complex problems [1].

In spite of these achievements, GAs are not flawless. Almost optimization problems have some constraints. In the course of evolution, unfortunately, the chromosomes which do not satisfy the constraints are generated. A large number of these chromosomes are generated, affecting the search performance of GA greatly. In practical, these chromosomes are eliminated in SGA, and mentioned as LCs hereafter. However, we consider that the LCs would have some excellent features [2]. For instance, some LCs take quite higher fitness value even though they do no satisfy the constraints.

In this paper, based on a double islands algorithm model, we propose a recycling method of LCs in GA with a double island algorithm model based on an immune operation. In our method, LCs are recycled by applying vaccines extracted from all the LCs. Lastly, we apply our method to two kinds of MKP, small size and large size, in order to investigate performance by comparing with SGA.

2. Recycling Method of LCs in GA

2.1. MKP

Let us explain MKP briefly. MKP is one of classic constrained combinatorial optimization problems, which belongs in NP-hard problem[3]. MKP can be applied in a wide variety of fields, such as project selection, processor allocation in distributed system, cutting stuck or capital budgeting. The goal of MKP is to find a subset of objects that maximizes the total profit while satisfying the constraint conditions.

In this paper, a state of jth object is presented by $x_j^{(k)}$, where the variable $x_j^{(k)} \in \{0,\ 1\}$, "0" means that the jth object is not put into kth knapsack and "1" means that the jth object is put into kth knapsack. Therefore, MKP is described as follows:

$$\text{Maximize} \sum_{k=1}^{M} \sum_{j=1}^{N} v_j x_j^{(k)}, \tag{1}$$

where v_j represents the value of the jth object, N the number of the objects, and M the number of knapsacks. The constraint conditions are as follows:

$$\sum_{j=1}^{N} w_j x_j^{(k)} \leq W^{(k)} \quad \text{for all } k, \tag{2}$$

$$\sum_{k=1}^{M} x_j^{(k)} \leq 1 \quad \text{for all } j, \tag{3}$$

where w_j represents the weight of the jth object and the $W^{(k)}$ the maximum weight of the kth knapsack. The first constraint condition, Eq.(2), means that for all the knapsacks, the total weight of the knapsack must be smaller than its maximum weight. The second constraint condition, Eq.(3), means that for all the objects, one object must be put into a certain knapsack once.

2.2. GA for MKP

At first, let us explain our coding method of GA for MKP. In our GA, each chromosome is composed of N genes, $\mathbf{g} = (g_1, g_2, \cdots, g_N)$, and each $g_j \in \{0, 1, \cdots, M\}$. In our coding, $g_j = m\ (\neq 0)$ means that the jth object is put into the mth knapsack. On the other hand, $g_j = 0$ means that the jth object isn't put into any knapsacks. Thus, the jth gene g_j corresponds to the jth object variable $x_j^{(k)}$ as follows:

$$g_j = k \implies x_j^{(k)} = k. \tag{4}$$

where k takes $\{0, 1, \cdots, M\}$. According to Eqs.(1) and (4), in this paper, the fitness is evaluated as follows:

$$\text{Fitness} = \sum_{k=1}^{M} \sum_{j=1}^{N} v_j x_j^{(k)}. \tag{5}$$

At last, we explain our genetic operation. In a crossover operation, we employ uniform one, where a binary random string with the length of N is applied. The element of the string takes either "0" or "1", where "0" means that at the corresponding point, the pair of genes do not cross. On the other hand, "1" means that at the corresponding point, the value of genes are changed each other. The procedure is represented in Fig.1. The candidates of the pair of chromosome are selected according to the ranking selection. In our crossover operation, indeed, whether the uniform crossover operation is applied or not depends on the probability of P_{C}. In the latter simulation, we employ $2N$ pairs.

In mutation operation, the candidates among the $2N$ children chromosomes, where the crossover operation has completed, are selected with the probability of P_{m}. Applying point and replacing value of the gene are randomly determined.

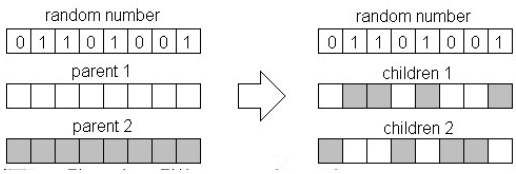

Figure 1: Uniform crossover operation.

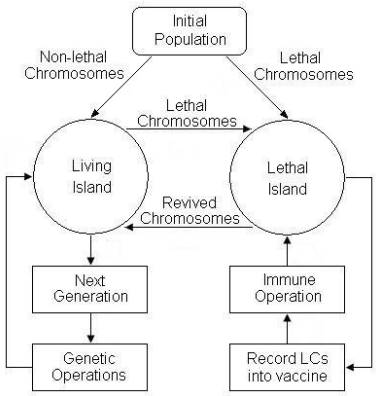

Figure 2: Double island model of GA.

Before the selection operation, we discriminate whether the chromosome is LC or not, where LC is a chromosome which does not satisfy the constraint conditions. LCs are revived by our recycling method with the immune operation. The details are presented later. In our GA, thus, the chromosomes, where the crossover operation and the mutation operation have completed, are categorized into two kinds of population pools, (i) a living island which contains living chromosomes that satisfy all the constraint conditions and (ii) a lethal island which contains LCs, as shown in Fig.2. This type of GA is referred as double islands model of GA proposed by [4]. It should be noted that the revived chromosomes of LCs are moved into the living island.

According to either a ranking selection or a roulette wheel one, in the selection operation, we select the next generation of chromosomes with the population of N among the N parent chromosomes and $2N$ children chromosomes adding the revived chromosomes of LCs. The solving ability extremely depends on the selection methods. The details are presented in Sec. 3.

2.3. Recycling Method of LCs

The immune operation will perform after the crossover operation and the mutation operation have completed. Let us assume that genetic operations have completed at tth generation in GA. In this paper, the immune operation consists of two procedures, (i) vaccine production procedure and (ii) recycling procedure. First is to generate the vaccine. The other is to revive LCs into the living island by making use of the vaccines.

2.3.1. Vaccine Production Procedure

In the vaccine production procedure, we make vaccines from all the LCs including parent ones. The procedure are as follows:

1. After genetic operations have completed at tth generation in GA, choose all the LCs until $T(t)$ generations ago. The ath LC denotes as $\mathbf{g}^{(a)}(t)$ $(a = 1, \cdots, L(t))$, where $L(t)$ is the total number of LCs until $T(t)$ generations ago. In addition, $l(t)$ is the number of LCs at the tth generation. In other word, $l(t)$ LCs have to be recycled and revived into the living island.

2. Generate LCs' pool from all the LCs as follows:

$$G(t) = \begin{pmatrix} \mathbf{g}^{(1)}(t) \\ \vdots \\ \mathbf{g}^{(L(t))}(t) \end{pmatrix}$$

$$= \begin{pmatrix} g_1^{(1)}(t) & \cdots & g_N^{(1)}(t) \\ \vdots & \ddots & \vdots \\ g_1^{(L(t))}(t) & \cdots & g_N^{(L(t))}(t) \end{pmatrix} \quad (6)$$

3. According to the LCs pool, the vaccine is produced as follows:

$$\upsilon_j^b(t) = \sum_{a=1}^{L(t)} f(g_j^{(a)}(t); b), \quad (7)$$

where $j = 1, \ldots, N$, $b = 1, \ldots, M$ and

$$f(g_j^{(a)}(t); b) = \begin{cases} 1 & (g_j^{(a)}(t) = b), \\ 0 & (g_j^{(a)}(t) \neq b). \end{cases} \quad (8)$$

In the procedure of the producing vaccine, the vaccine $\upsilon_j^b(t)$ represents how many times the jth object is put into the bth knapsack among the LCs' pool of G(t).

2.3.2. Recycling Procedure based on Vaccines

In the recycling procedure, we recycle and revive $l(t)$ LCs at the tth generation. Let us assume that the ath LC, $\mathbf{g}^{(a)}(t)$, is tried to recycle and revived into the living island. Based in the vaccines, find the index i^* from overweight kth knapsack, which satisfies the following condition,

$$\sum_{j=1}^{N} w_j x_j^{(k)} > W^{(k)}. \quad (9)$$

The index i^* is given by,

$$i^* = \text{Arg} \min_{j \epsilon 1,\ldots,N} \upsilon_j^k(t). \quad (10)$$

Finally, let the value of the i^*th gene in ath LC of $\mathbf{g}^{(a)}(t)$ to be,

$$g_{i^*}^{(a)}(t) = 0. \quad (11)$$

The finding procedure of the i^* index means that we search an object which was put in the kth knapsack with the smallest opportunity among LCs' pool of $G(t)$. If such the i^* object is removed from the kth knapsack, the effect on the fitness given by Eq.(5) would be small. Then, we set $g_{i^*}^{(a)}(t) = 0$.

The above procedure is continued while the ath LC, $\mathbf{g}^{(a)}(t)$, satisfy the constraint conditions of Eqs.(2) and (3). In addition. the above procedure is continued while $l(t)$ LCs are moved into the living island from the LCs' pool.

3. Computer Experiments

3.1. Purposes and Methods

Hereafter, our GA with the recycling method based on the immune operation is referred as IGA. In computer experiments, we investigate the practical ability of IGA for MKP. We perform two scales of computer experiments as follows:

- small size of MKP
- large size of MKP

In the small size of MKP, the number of objects is 15, and the number of knapsacks is 3. In the large size of MKP, the number of objects is 1000, and the number of knapsacks is 50. In the small size problem, it is possible to solve the exact solution. In the large size problem, on the other hand, it is difficult to solve the exact solution.

In our GA, for the small size problem, the population size of chromosomes is 50, and for the large size problem, the population size of chromosomes is 50. The other parameters of GA are as follows: $P_c = 0.8$ and $P_m = 0.05$. In our recycling method, we set the duration $T(t)$ of generations to be t for simplicity. Thus, at the tth generation, we calculate LCs' pool from the 1st generation to the tth generation.

At first, we investigate dependence on the selection methods. Thus, we study which one is suitable for IGA in the large size of MKP, the ranking selection or the roulette wheel selection. The procedure of GA is stopped at 500th generations. We evaluate the fitness with respect to the CPU time.

At last, in order to show the practical ability, we compare the fitness of IGA with SGA. SGA means that GA without our recycling method. In this experiment, we employ the ranking one in the selection operation.

In the above two experiments, we perform GA 10 times for different initial chromosomes, and evaluate the average of the fitness for the 10 trial. It should be noted that we performe all the computer experiments under the same situation with PC of Core*i*5 2410M 2.3 GHz.

3.2. Results

3.2.1. Dependence on the selection methods

The result of the best fitness is shown in Fig.3, where the red line represents the result of IGA with the ranking selection and the blue line IGA with the roulette wheel selection. From result, that IGA with the ranking selection achieved a very fast and accurate convergence. So it is a good suitable selection operator for IGA in solving MKP.

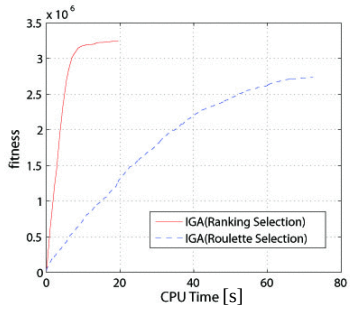

Figure 3: Dependence on the selection methods.

3.2.2. Small Size of MKP

The result is shown in Fig.4, where the red line represents the result of IGA and the blue line SGA. From this figure, the converged optimal solution found by IGA is better than that of SGA. In addition, IGA has succeeded in converging to exact solution, whose fitness takes $47,996$. Thus, it means that IGA works well in solving MKP.

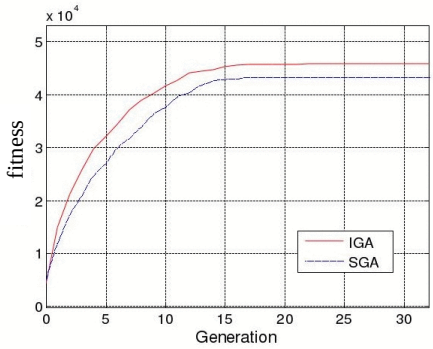

Figure 4: Fitness for the small size problem.

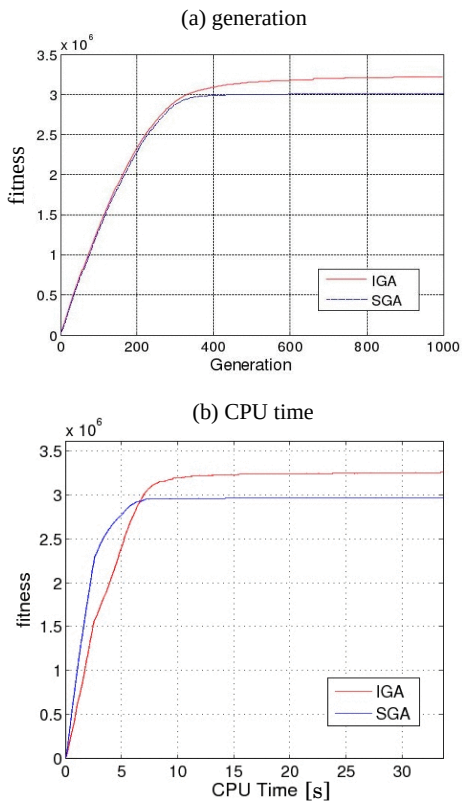

Figure 5: Fitness for the large size problem W.R.T (a) generation and (b) CPU time.

3.2.3. Large Size of MKP

The result is shown in Fig.5(a), where the red line represents the result of IGA and the blue line SGA. At the 1,000th generation, the fitness of IGA takes 31788281 and that of SGA takes 30119416, indicating the imroving ability with 5.5%. In addition, the dependence of the fitness on CPU time is given in Fig.5(b), indicating that IGA gives better solutions after 6.6[s] comparing with CPU time. Obviously, our proposed method is definitely superior to the SGA, on the aspect of the optimization mechanism. Thus, IGA is practical in solving MKP.

4. Discussions

From our computer experiments, the ranking selection is much better than the roulette wheel selection for the selection operation. inferior is implemented via adopting the ranking selection operator. In our method, we utilize LCs by reviving them with the immune operation. Thus, the revived LCs would takes higher fitness, introducing the variety of chromosomes at the next generation. Therefore, the ranking selection could succeed to choose better chromosomes with keeping the variety. It is future problem to investigate whether the revived LCs actually takes higher fitness or not.

Introducing the immune operation, we have succeeded to improve the performance of GA for MKP. However, the improving rate is not so large of 22% at 10,000th generation, evaluating as follows,

$$\text{improving rate} = \frac{\text{improving rate of fitness}}{\text{increasing rate of cpu time}}, \qquad (12)$$

where IGA take 19.5[s] and SGA 15.6[s] at the 1,000th generation. It suggest us that there is a possibility of improving the performance by changing the immune operation, for instance, the duration time of $T(t)$ and the candidate of revived LCs.

References

[1] K. S. Tang, K. F. Man, S. Kwong, Q. He, "Genetic Algorithm and their Applications," *IEEE Signal Processing Magazine*, vol.13, pp.22–37, 1996.

[2] I. Hitoshi, S. Nobuo, "The Influence of Lethal Gene on the Behavior of Genetic Algorithm," *Society of Instrument and Control Engineers*, vol.31, pp.569–576, 1995.

[3] D. Pisinger, "An Exact Algorithm for Large Multiple Knapsack Problem," *European Journal of Operational Research*, vol.114, pp.528–541, 1999.

[4] M. C. Xie, T. Yamaguchi, T. Odaka, H. Ogura, "An Analysis of Evolutionary States in the GA with Lethal Genes," *IEICE Trans. DII*, vol.J79-D-II, pp.870–878, 1996.