Usefulness of lethal chromosomes in genetic algorithms solving the constrained optimization problems

Yalong Zhang[1]*, Hisakazu Ogura[2], Xuan Ma[3]

[1]*College of Electrical and Information Engineering, Quzhou University, Quzhou 324000, China*

[2]*Graduate School of Engineering, University of Fukui, Fukui 910-8507, Japan*

[3]*Faculty of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China*

Received 1 June 2014, www.cmnt.lv

Abstract

The infeasible solutions are often generated in population as evolutionary computation solving the combinatorial optimization problems. The number of infeasible solutions impacts the performance of the evolutionary computation searching the optimal solution, in the worst case the algorithm ceases to run. In genetic algorithms, encoding of infeasible solutions is referred to as lethal chromosomes. In this study, we discover a property of lethal chromosomes that: although lethal chromosomes carry out the infeasible solutions in genetic algorithms, their statistical property implies an underlying similarity with the exact solution of the optimization problems. Hereby we propose an operation using statistical property of lethal chromosomes to handle with the lethal chromosomes themselves. Simulation experiments on a large number of test cases demonstrated that it can improves obviously the performance of genetic algorithms to use the statistical property of lethal chromosomes.

Keywords: Genetic Algorithm, Infeasible Solution, Lethal Chromosome, Evolutionary Computation.

1 Introduction

With representing the solution of problem as a chromosome, genetic algorithm (GA) refers the chromosomes that violate the constraint conditions to as lethal chromosomes (LCs). In the population of GA, due to crossover and mutation operations, LCs are often generated with high rates. The greater number of LCs in the population, the worse search performance of the GA.

Early, Iima Hitoshi (1995) investigated the effects of LCs on the performance of the GA, but did not propose a method for handling these problems [1]. Mengchun Xie (1996) proposed an algorithm model to revive the LCs by random crossover and mutation operations [2]. In recent years, researches focusing on the problems associated with infeasible solutions have made some progress. Lyndon While et al. (2009-2013) have been made some achievements on problems associated with infeasible solutions in evolutionary computation [3-7].

In this paper, we propose genetic algorithm on multidimensional knapsack problem (MDKP) that handles the LCs with a reviving operation (RGA). Specifically, we (2009) previously proposed another genetic algorithm using lethal chromosomes on multi-knapsacks problems [8]. But regrettably, we mistakenly referred the multi-knapsacks problems to as multidimensional knapsack problem in [8]. This paper is really to solve the multidimensional knapsack problem, and the reviving operation is totally different.

2 A genetic algorithm using LCs on MDKP

As an NP-hard problem, MDKP is to find a subset of objects that maximizes the total profit while satisfying some resource constraints, which can be formulated as:

$$Maximize \quad \sum_{j=1}^{n} v_j x_j \,, \tag{1}$$

$$\begin{aligned} s.t. \quad & \sum_{j=1}^{n} w_{ij} x_j \le c_i \quad \forall i \in I \\ & x_j \in \{0,1\} \quad \forall j \in J, \end{aligned} \,, \tag{2}$$

where n is the number of objects, m is the number of resources, v_j is the value associated with object j, w_{ij} is the consumption of resource i for object j, c_i is the available quantity of resource i (capacity of knapsacks for the i^{th} resource), and x_j is the decision variable with object j and is set to 1 if j^{th} object is selected (and is otherwise set to 0). $I = \{1, 2, \dots, m\}$, $J = \{1, 2, \dots, n\}$.

To solve the MDKP, many researchers consider the GA or other evolutional algorithm. P.C. Chu (1998) presented a GA for MDKP with a repair operator [9]. Günther R. Raidl (1999) proposed a weight-coding in GA for MDKP [10]. Günther R. Raidl (1998) presented an improved hybrid GA for MDKP [11]. Jens Gottlieb (2000) solved the MDKP with a permutation-based GA [12].

***** ***Corresponding author*** e-mail: yalong-2008@hotmail.com

2.1 ALGORITHM STRUCTURE

Our RGA has two types of chromosome pools, namely, the *living island* and *lethal island*. The former contains chromosomes, referred to as non-lethal (feasible) chromosomes, which satisfy all constraints. The latter consists of the LCs. In the living island, chromosomes are evolved by genetic operations, and in the lethal island, LCs are revived by reviving operations. This structure is described as following way.

```
1  while initializing population do
2      Generate initial population, and then move the
       non-lethal chromosomes into the living island, the
       lethal chromosomes into the lethal island;
3  end
4  while evolving process do
5      for living island do
6          All of the chromosomes in the living island
           evolve into the next generation by genetic
           operations, and the new lethal chromosomes
           move to lethal island. In this process, the
           vaccine must be trained;
7      end
8      for lethal island do
9          Each chromosome in the lethal island is handled
           by reviving operations and is then moved to the
           living island;
10     end
11 end
12 Repeat the evolving process until meeting the
   termination conditions of the GA;
```

ALGORITHM 1 Algorithm structure of RGA

We use roulette as selection operation, two-point crossover and mutation rates of 0.05 in genetic operations. Next, we mainly focus on reviving operation, which is performed in *lethal island*.

2.2 REVIVING OPERATION

Idea of the reviving operation is introduced from immune system of biology and medicine. As FIGURE 1, the previous infection led the system to generate the immune memory; the system handles the re-infection with previous memory for avoiding from being infected again.

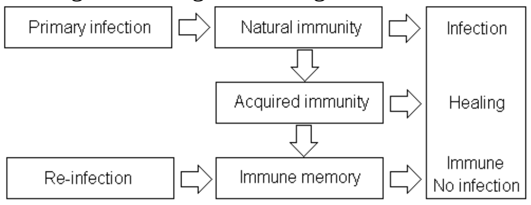


FIGURE 1 Immune system

In this context, we record the statistical information of previous LCs as immune memory, and then handle the succeeding LCs by previous memory. The operation to handle the LCs is called reviving operation in this paper.

As an immune memory, a multi-valued string schema $s_1s_2 \ldots s_n$ is constructed and has the initial value that $s_j = 0$, $\forall j \in J$. The $s_1s_2 \ldots s_n$ will be used as *immune memory* by reviving operation, so we call it *vaccine*. During the RGA evolving, while every LC referred as $x_1x_2 \ldots x_n$ generated, to perform the ALGORITHM 2 flushing the vaccine $s_1s_2 \ldots s_n$.

```
Input: a lethal chromosome $x_1 x_2 \ldots x_n$
Output: flushed vaccine $s_1 s_2 \ldots s_n$
1 for ($j = 1$ to $n$) do
2     if ($x_j = 1$) then
3         $s_j \leftarrow s_j + 1$;
4     end
5 end
```

ALGORITHM 2 Training the vaccine with lethal chromosomes

By this way the $s_1s_2 \ldots s_n$ is a changing string, which is refurbished by every lethal chromosome while evolution of population as long as LCs appearing. As LCs generated are handled by reviving operation in every generation, while the vaccine is refurbished by LCs while the vaccine is used by reviving operation in RGA.

In order to achieve an efficient implementation of the reviving operation, a pre-processing routine is applied to vaccine $s_1s_2 \ldots s_n$ that sorts and renumbers genes of LC according to the *increasing* order of s_j's. Assuming $x_1x_2 \ldots x_n$ with pre-processing routine is the LC to be revived, proposed reviving operation is described by pseudo-code as following.

```
Input: a lethal chromosome
Output: a revived chromosome
1  Let $W_i = \sum_{j=1}^{n} w_{ij} x_j, \forall i \in I$;
2  for ($j = 1$ to $n$) do
3      if ($x_j = 1$) and ($W_i > c_i, \forall i \in I$) then
4          $x_j \leftarrow 0$;
5          $W_i \leftarrow W_i - w_{ij}, \forall i \in I$;
6      end
7  end
8  for ($j = n$ to $1$) do
9      if ($x_j = 0$) and ($W_i + w_{ij} \le c_i, \forall i \in I$) then
10         $x_j \leftarrow 1$;
11         $W_i \leftarrow W_i + w_{ij}, \forall i \in I$;
12     end
13 end
```

ALGORITHM 3 Reviving operation

3 Computational experiments

A set of standard test data of the MDKP was referred and used by P.C. Chu [9], Günther R. Raidl [10, 11], and Jens Gottlieb [12]. These test data contain 10 instances for each combination of $m \in \{5, 10, 30\}$, $n \in \{100, 250, 500\}$, and $\alpha \in \{0.25, 0.50, 0.75\}$, where $\alpha = c_i / \Sigma_{j=1}^{n} w_{ij}$ being the tightness ratio of instance. Since the exact solution values for most of these problems are unknown, the quality of a solution is measured by the percentage gap of the objective value *fitns* with respecting to the optimal value of the LP-relaxed problem f^{LP}_{max}: $\%\text{-}gap = 100 \times (f^{LP}_{max} - fitns) / f^{LP}_{max}$. The proposed RGA is tested on these 270 MDKP instances and the mean results are shown in TABLE 1.

TABLE 1 Computational results of other algorithms and RGA

Problems	**Average *%-gap***					
All of 270 instances	GA with H1	GA with H2	Swap	Insert	Improved GA	RGA (proposed)
Average	0.589	0.646	0.641	0.629	1.13 - 0.53	0.5472

We also list the available results of other references in TABLE 1 to compare with. The first columns indicate test is on the 270 instances. The next columns in turn report the other results of average *%-gap*. That are *GA with H1* and *GA with H2* proposed by Günther R. Raidl [10], *Swap* and *Insert* are from Jens Gottlieb [12], and the *Improved GA* reported by also Günther R. Raidl [11] for *%-gap* obtained from initial population to $(10^6)^{th}$ generations. The last column reports the results of RGA with the computer condition that CPU is Celeron 1.0 and the algorithms were coded in Visual C++.net (2003).

To analyse the test results, above all we discuss the condition of experiment. The *GA with H1* and the *GA with H2* was tested with population size of 100 and 10^5 solutions had been evaluated. *Swap* and *Insert* had evaluated 10^6 non-duplicate solutions and then get results. *Improved GA* recorded the best solution from initial population to $(10^6)^{th}$ generations. Comparing with them, RGA obtain the results with population size of 50 and running terminates at $(10^4)^{th}$ generation.

Averagely for all instances, RGA obtained the *%-gap* as 0.5472, which is smaller than the results from the algorithm *GA with H1*, *GA with H2*, *swap* and *Insert*, but except the *Improved GA*. *Improved GA* of Günther R. Raidl [11] obtained the best average *%-gap* as 0.534 at $(10^6)^{th}$ generation, it is very great. Because such an experiments to 10^6 generations will cost too CPU seconds to finish for us, we test the RGA only for 10^4 generations and obtained the finial average *%-gap* as 0.5472 with population size 50. Relatively, the population size Günther R. Raidl adopted in *Improved GA* is 100. To compare with *Improved GA* at same generations, we calculated the average *%-gap* obtained also at $(10^4)^{th}$ generation for *Improved GA* according to data in Refs. [11], the result is 0.6211 which greater than 0.5472 despite of large population size 100.

4 Discussion on feature of LCs

To analyse the feature of LCs, we firstly give a definition of *similarity ratio* between two chromosomes. To any two chromosomes X_1 $(x_1^1 x_2^1 \ldots x_n^1)$ and X_2 $(x_1^2 x_2^2 \ldots x_n^2)$, the *similarity ratio* of them is defined by:

$$similarity\ ratio(X_1, X_2) = 100\frac{1}{n}\sum_{j=1}^{n}(x_j^1 \odot x_j^2), \qquad (3)$$

where, if $x_j^1 = x_j^2$, then $x_j^1 \odot x_j^2 = 1$; otherwise $x_j^1 \odot x_j^2 = 0$. Actually, the *similarity ratio* is used to measure how many the two chromosomes have same value genes.

In order to study feature the vaccine contained, it is hypothesis that at a certain generation $s_1 s_2 \ldots s_n$ is processed to a binary-value string $s'_1 s'_2 \ldots s'_n$ by the way that: for $\forall j \in J$, if $(s_j > t)$ then $s'_j \leftarrow 1$, otherwise $s'_j \leftarrow 0$, where t is the threshold be used to classify the s_j's into one or zero, the t is determined as an appropriate value so that it takes the max *similarity ratio* for $s'_1 s'_2 \ldots s'_n$ and *exact chromosome* (*exact solution*) of the problem. Here it is assume that the exact chromosome is known beforehand. We consider the value of *similarity ratio* of $s'_1 s'_2 \ldots s'_n$ and exact chromosome as *feature of vaccine*.

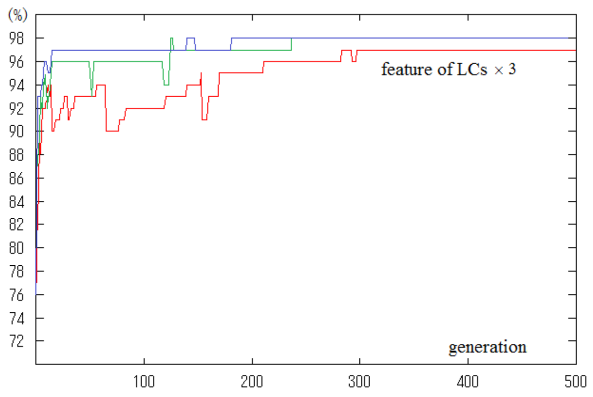


FIGURE 2 Feature of LCs [“feature of LCs” (×3).]

To three MDKP instances picked from OR-library [9] that $m = 5$, $n = 100$, we firstly solve their exact chromosomes with branch and bound method (BBM) to measure the *feature* of vaccine. Their curves of *feature of vaccine* in RGA are shown as FIGURE 2.

We could learn from FIGURE 2 that (i) the *features of vaccine* are increase in the overall; (ii) the *features of vaccine* rise to a high degree at later generation that some to 97 % and some to 98 %. That also means most of genes of exact chromosome could be indicated the value via to classifying the corresponding *bit* of vaccine with a threshold *t*.

5 Conclusions and future research

The paper discovered an important feature of lethal chromosomes, and proposed RGA to use this feature of lethal chromosomes based on reviving operation. Appling RGA to MDKP, a large number of testing results indicate that using lethal chromosomes based on reviving operation could obviously improve search performance of GA. At same time there are also some works should to be done to improve the performances of RGA deeply.

In addition to the method of reviving operations should be researched further in the future, researches should focus on not only lethal chromosomes, but also to excavate the more evolutionary resource in infeasible solution regions. On the other hand, if the proposed reviving operations are improved to decrease the time complexity and allow rapider operation, the GA will be applicable to a wider range of field.

Acknowledgments

This work is supported by the Scientific Research Foundation (Project No.Y201430654) funded by the Zhejiang Provincial Education Department (China).

References

[1] Iima Hitoshi, Sannomiya Nobuo 1995 The Influence of Lethal Gene on the Behavior of Genetic Algorithm *The society of instrument and control engineers* **31**(5) 569-76

[2] Mengchun Xie, Tetsushi Yamaguchi, Tomohiro Odaka, Hisakazu Ogura 1996 An Analysis of Evolutionary States in the GA with lethal Genes *The Information and Systems Society, Institute of Electronics, Information and Communication Engineers* **J79-D-II**(5) 870-8

[3] Lyndon While, Philip Hingston 2013 Usefulness of Infeasible Solutions in Evolutionary Search: an Empirical and Mathematical Study *Proc. of 2013 IEEE Congress on Evolutionary Computation*, Cancún, México 1363-70

[4] Deepak Sharma, Prem Soren 2013 Infeasibility Driven Approach for Bi-objective Evolutionary Optimization *Proc. of 2013 IEEE Congress on Evolutionary Computation (CEC)*, Cancun, Mexico 868-75

[5] Tapabrata Ray, Hemant Kumar Singh, Amitay Isaacs, Warren Smith 2009 Infeasibility Driven Evolutionary Algorithm for Constrained Optimization *Constraint-Handling in Evolutionary Optimization Studies in Computational Intelligence* **198** 145-65

[6] Patryk Filipiak, Krzysztof Michalak, Piotr Lipinski 2011 Infeasibility Driven Evolutionary Algorithm with ARIMA-Based Prediction Mechanism *Intelligent Data Engineering and Automated Learning - IDEAL 2011* **6936** 345-52

[7] Maristela Oliveira Santos, Sadao Massago, Bernardo Almada-Lobo 2010 Infeasibility handling in genetic algorithm using nested domains for production planning *Computers &Operations Research* **37**(2010) 1113-22

[8] Yalong Zhang, Xuan Ma, Jousuke Kuroiwa, Tomohiro Odaka, Hisakazu Ogura 2009 A Genetic Algorithm with Utilizing Lethal Chromosomes *Proc. of 2009 IEEE International Conference on Fuzzy Systems*, Jeju Island, Korea 2047-50

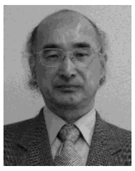
[9] Chu P C, Beasley J E 1998 A Genetic Algorithm for the Multidimensional Knapsack Problem *Journal of heuristics* **4**(1) 63-86

[10] Günther R Raidl 1999 Weight-Codings in a Genetic Algorithm for the Multiconstraint Knapsack Problem *Proceedings of the 1999 ACM symposium on applied computing*, Texas, United State 291-6

[11] Günther R Raidl 1998 An Improved Genetic Algorithm for the Multiconstrained 0–1 Knapsack Problem *Proceedings of the 5th IEEE International Conference on Evolutionary Computation*, Anchorage, AK 207-11

[12] Jens Gottlieb 2000 Permutation-Based Evolutionary Algorithms for Multidimensional Knapsack Problems *Symposium on Applied Computing Proceedings of the 2000 ACM symposium on Applied computing*, Como, Italy 408-14

Authors

	Yalong Zhang, born on April 6, 1976, China **Current position, grades:** works at College of Electrical and Information Engineering, Quzhou University, Quzhou, China; and is involved in research related to intelligent information processing. **University studies:** master's course at Xi'an University of Technology, China, in 2007, and received his D.Eng. degree from University of Fukui, Japan, in 2011
	**Hisakazu Ogura, born on October 1, 1946, Japan** **Current position, grades:** member of IECE Japan, IPS Japan, and SOFT Japan **University studies:** D.Sc. degree from Kyoto University, Japan, in 1977 **Scientific interests:** in knowledge representation and processing in the fields of artificial intelligence, medical informatics and medical image processing by applying genetic algorithms, artificial neural networks, symbol processing and/or fuzzy theory. **Experience:** was a professor of the Department of Human and Artificial Intelligent Systems at the Faculty of Engineering, University of Fukui, Japan.
	Xuan Ma, born on February 13, 1962, China **Current position, grades:** professor at the Faculty of Automation and Information Engineering, Xi'an University of Technology, China and working on evolutionary computation and intelligent information processing **University studies:** D.Eng. degree from University of Fukui, Japan, in 2002