An Immune Operation for Lethal Chromosomes of Genetic Algorithm

Yalong Zhang† Hisakazu Ogura† Xuan Ma† Jousuke Kuroiwa† Tomohiro Odaka†
University of Fukui† Xian University of Technology†

Abstract: This paper proposes an immune operation to revive the lethal chromosome (LC) for genetic algorithm (GA) so that LCs can be utilized instead of being abandoned. In general GA, LCs correspond to the infeasible solution and are eliminated from the population. However, the LCs with evolutions contain some excellent building block, we can reuse them instead of wasting resources. By this way the performance of GA is improved available.

Keywords: genetic algorithm, lethal chromosome, artificial immune, MKP

1 Introduction

In a population pool of GA, due to crossover and mutation operations, sometimes, LCs are produced at high rate, especially to the combinatorial optimization problems having rigorous constraint. The more number of LCs increase in the population, the worse performance of GA decreases. At the worst case, algorithm would stop. Researches focusing on problem of LCs are still rare. Iima Hitoshi[1] (1995) had investigated the effects of LC to performance of GA, but hadn’t supplied a method to deal with. The general method to LCs is eliminating them from its population. In this paper we propose a method to revive and utilize the LCs for GA based on artificial immune theory, which combining the evolutionary information of the chromosome with the characteristic information of the problem. We apply the proposed algorithm to one of the typical constrained optimization problem, Multidimensional Knapsack Problems (MKP).

2 Multidimensional Knapsack Problems

Let us explain the Multidimensional knapsack problem (MKP) briefly. MKP is a well-known NP-hard problem which is formulated as:

Maximize \( \sum_{j=1}^{n} v_{j} x_{j} \) \hspace{1cm} (1)

s.t. \( \sum_{j=1}^{n} w_{j} x_{j} \leq c_{i} \) \hspace{1cm} (i = 1 \cdots m) \hspace{1cm} (2)

\( x_{j} \in \{0, 1\} \) \hspace{1cm} (j = 1 \cdots n),

where \( n \) is the number of objects and \( m \) is the number of resources, \( v_{j} \) is the value associated with object \( j \), \( w_{j} \) is the consumption of resource \( i \) for object \( j \), \( c_{i} \) is the available quantity of resource \( i \) (capacity of knapsacks for \( i^{th} \) resource), and \( x_{j} \) is the decision variable with object \( j \) and is set to 1 if \( j \) is selected (and set to 0 otherwise). Constraints \( c_{i} \) \hspace{1cm} (i = 1 \cdots m) \hspace{1cm} described in Eq.(2) are called knapsack constraints, so the MKP is also called the \( m \)-dimensional knapsack problem.

3 An Immune Operation for Lethal Chromosomes of Genetic Algorithm

In order to utilize the LCs instead of abandoning them, we extract the excellent building block from the LC firstly, and then restructure a new chromosome based on extracted building block with immune operation, which is composed of vaccine and vaccination according to immune theory.

3.1 Active ingredients of LC

The LCs with evolutions contain some active ingredients which are similar to the parts of the optimal solution; we call them excellent building blocks. To extract excellent building block from LCs, we give a method to estimate the building block, and with it extract a better one according to estimate from \( n/2 \) to \( n \) to length of building block.

For building block of LC \( l_{1}l_{2}..l_{m} \), a binary string \( f_{1}f_{2}..f_{n} \) is used to mark genes which belongs to building block, where \( \forall i \in \{1, 2, \cdots , n\}, f_{i}=1 \) denotes \( i \) belongs to building block, \( f_{i}=0 \) denotes not. The estimate of building block is defined as following:

\[ e(f_1f_2..f_n) = \frac{\sqrt{1 - u}}{k} \]

\( u = \max \left( \frac{n}{k} \sum_{j=1}^{n} f_{j} w_{j} - c_{i}, v \right) \)

where \( v = \sum_{j=1}^{n} f_{j} v_{j} / n \), \( u = \max \left( \frac{n}{k} \sum_{j=1}^{n} f_{j} w_{j} - c_{i}, v \right) \), \( v \) is different from fitness of chromosome, which is parts of fitness, and is not bounded by \( c_{i} \) of knapsack. \( k \) is number of genes in building block marked by \( f_{1}f_{2}..f_{n} \) in LC, it is also called length of building block.

The steps of extracting the excellent building block from LC are outlined as following:

1: \( d \leftarrow 0; \)
2: for \( j=n/2 \) to \( n \) do:
3: \( e_{1}e_{2}..e_{n} \leftarrow (0, 0, 0, \cdots , 0); \)
4: select \( j \) genes randomly from \( l_{1}l_{2}..l_{m} \) and set corresponding \( e_{i} \leftarrow 1; \)
5: if \( (e_{1}e_{2}..e_{n} > d) \) then
6: \( f_{1}f_{2}..f_{n} \leftarrow e_{1}e_{2}..e_{n}; \)
7: \( d \leftarrow e_{1}e_{2}..e_{n}; \)
8: end if
9: end for
10: get \( f_{1}f_{2}..f_{n} \) to mark the excellent extracted building block in \( l_{1}l_{2}..l_{m} \).

3.2 Training vaccine

In proposed IGA, with extracting excellent building block, we introduce the immune theory into GA to
revive the LCs created by genetic operation.
In the beginning of evolution of GA, construct a schema of vaccine: \( s_1 \ldots s_n \), and set \( s_i=0 \) (\( i=1 \ldots n \)). During of evolution, when each LC \( l_1 \ldots l_n \) appears due to operation of GA, make following operation:
\[
\begin{align*}
\text{if } l_i & = 1 \text{, add 1 to } s_i; \\
& \text{(i=1,...,n)} \\
\text{otherwise, } s_i & \text{ is unchanged; (i=1,...,n)}
\end{align*}
\]
so \( s_1 \ldots s_n \) will increase with evolution process forward, and \( s_i \) \( s_i \neq f \) will usually being different each other. For convenience to state, name the schema of vaccine \( s_1 \ldots s_n \) as vaccine1.
In the other hand, we construct another schema of vaccine \( b_1 \ldots b_n \) to record excellent genes tendency. In the course of GA evolution, there is always one of the best chromosomes is retained; the old best chromosome is replaced when a new one appears. With setting \( b_1 \ldots b_n \leftarrow (0,0,\ldots,0) \), during of evolution, whenever a new best chromosome \( l_1 \ldots l_n \) appears, make following operation:
\[
\begin{align*}
\text{if } l_i & = 1 \text{, add 1 to } b_i; \\
& \text{(i=1,...,n)} \\
\text{otherwise, } b_i & \text{ is unchanged; (i=1,...,n)}
\end{align*}
\]
so \( b_1 \ldots b_n \) will increase with evolution process forward too, and \( b_1 \) \( b_1 \neq f \) will usually being different each other. We call the \( b_1 \ldots b_n \) as vaccine2.
3.3 Vaccination
To a worthwhile LC we have much available resources to deal with it, excellent building block, vaccine1 and vaccine2, to revive it, there are two operations, and we summarize them in the following.
(1) The first operation is called immune1, which sets the remaining genes excepted the genes of extracted building block from LC as 1 firstly, and then according to tendency of vaccine1, with roulette method, to select and set genes as 0 one by one until LC being revived. Explaining with pseudo-code as follows:
\begin{enumerate}
\item set the LC for vaccination as \( l_1 \ldots l_n \);
\item extract the excellent building block from LC and mark them as \( f_1 \ldots f_n \);
\item for \( i=1 \) to \( n \) do:
\item if \( f_i=0 \) \( l_i \leftarrow 1 \);
\item end for
\item while \( (l_1 \ldots l_n \text{ is still lethal}) \) do:
\item select one \( l_i \) by roulette according to the value of \( s_i \) in the vaccine1 \( s_1 \ldots s_n \), from all \( l_i = 1 \) with \( f_i=0 \);
\item Set the selected \( l_i \leftarrow 0 \);
\item end while;
\end{enumerate}
(2) Another operation, immune2, sets remaining genes of LC excepted the genes of extracted building block as 0 firstly, when the chromosome become to non-lethal, according to tendency of vaccine2, with roulette method, to select and set genes as 1 one by one until LC become to lethal again from non-lethal state, and then make revocation of the last setting of gene.
Here we explain the algorithm just by the example of MKP; to other optimal problem, ideas and methods are similarly.
4 Experiments on MKP
We performed the computer experiments of proposed genetic algorithm with immune operations (IGA) with taking instances of MKP from the OR-Library, and compared with the experimental results of SGA (GA without utilizing the LCs). Ordinarily the evolutionary curve of GA is plotted against to the number of generation of GA. In this case since IGA has larger time-complexity for a generation than SGA, we have set terminating CPU-time and plotted the evolutionary curves against to CPU-time instead of generation to compare the results of IGA and SGA and to examine their performance.

![Fig.1 evolutionary curve against to CPU-time](image)

With setting population of GA as 50, termination CPU-time=3600(seconds), running times as 10, Fig.1 shows an average of evolutionary curve for MKP that \( m=30 \), \( n=500 \), in which SGA finished 630.5 generations averagely and IGA is 566.4. Although SGA is slight faster than IGA in generations, IGA is rapid in CPU-time and obtains a better solution finally.
5 Conclusions
This paper proposes a method for reviving LCs replacing the present means based on immune operations. Applying it to MKP indicates that, to some cases, especially to large scale problems, IGA is more effective to deal with the LCs than SGA, this improves obviously the search performance of GA in solving the constraint optimization problems.

References