THE PROPOSAL OF THE ALTERNATELY EVOLVING GENETIC ALGORITHM AND ITS APPLICATION

Zhanghui Chen, Kangrui Zhou, Yingtao Liu, Wenbin Zhan
Huazhong University of Science and Technology, China

Corresponding Author: Zhanghui Chen，Huazhong University of Science and Technology，
Center for Computational Materials Science and Measurement Simulation;
Address: ZS3#507, Huazhong University of Science and Technology, Wuhan, China;
Email: l1k2j3p4o5i6@163.com; Phone: +86 15827346521

Abstract. In view of the shortcomings of the usual genetic algorithm when solving multi-objective combinatorial optimization problems, the paper proposes an alternately evolving genetic algorithm. It adopts alternate strategy and optimizes multiple objectives one by one circularly. The solving results of the Sudoku puzzle indicate that this strategy can make the excellent patterns of different objectives all grow rapidly, and the results' comparison verify its feasibility and excellence in the general convergence. The sensitivity of the algorithm's parameter is also analyzed.

1 Introduction

Genetic algorithm (GA) is a random searching algorithm based on biological evolution and natural selection. It follows the "survival of the fittest" principles of the Darwinian evolution to implement probabilistic optimization of particular objectives. Due to its parallelism and globally searching ability, GA has been widely applied to function optimization, combinatorial optimization, artificial intelligence and some other fields. Especially in some multi-objective combinatorial optimization problems, GA is usually better than other algorithms.

When dealing with multi-objective combinatorial optimization problems, the usual genetic algorithm usually employs a comprehensive weighed objective to replace multiple objectives through constructing an evaluation function, and then uses unified genetic operators to optimize the weighted objective in a unified coding scheme. This method is effective in many cases. But for some combinational problems, the objectives may conflict with one another in the evolutionary process. That is to say, when one target gets met, others may deviate from the requirements very much. In these cases, the efficiency of the general evolution will be very low and the evolutionary process may be trapped in a state of random wander. By way of contrast, this paper puts forward an alternately evolving genetic algorithm from the opposite direction, which adopts alternate strategy to optimize multiple objectives one by one circularly. The solving results of the Sudoku puzzle verify the strategy's reasonableness and excellence.

2 The proposal of the alternately evolving genetic algorithm

2.1 Multi-objective combinatorial optimization

Multi-objective combinational optimization refers to the optimization of multiple objective functions of combinational variables in order to achieve the globally optimal solution or satisfactory solution. It is widely used in many fields, such as the university timetable, train schedules, Latin square, Sudoku and some other scheduling problems. Generally it can be modelled by the following expression:

$$\begin{aligned} &\min f_1(X),\ \min f_2(X),\cdots \min f_m(X) \\ &S.T.\begin{cases} g_1(X) \le b_1 \\ g_2(X) \le b_2 \\ \cdots \\ g_r(X) \le b_r \end{cases} \end{aligned} \tag{1}$$

Where $X=(x_1,x_2,\cdots x_n)$ are combinational variables; $f_k(X)$ is the k th objective function; and $g_i(X)$ is the i th combinational constraint condition.

2.2 The proposal of the alternately evolving genetic algorithm

It can be seen from formula 1 that multi-objective combinatorial optimization problems will become very complex with the increase of the number of objective functions and constraint conditions. It has been proved in theory that most of combinational optimization problems belong to NP complete problem, whose searching space will increase exponentially and even explode as the problem scale increases. The NP complete character of combinatorial optimization and the complexity of multi-objective optimization cause many classical algorithms not applicable any more. But GA can afford very well for its good parallelism and heuristic, and has been widely

applied to most of combinational optimization problems. When solving these problems, GA usually changes multiple objectives into a comprehensive weighed objective, namely:

$$\min\ a_1 f_1(X) + a_2 f_2(X) + \cdots + a_2 f_2(X) \tag{2}$$

Where $a_k\ (k = 1, 2, \cdots, m)$ refers to the weighted coefficient of the k th objective function.

By doing this, it then adopts a unified coding scheme to transform solution space into genetic space, and then promote individuals to a better state by a unified fitness evaluating operator, selection operator, crossover and mutation operator. This method that changes multiple objectives into a single objective is not applicable to some cases, such as problems with conflicting objectives. If we employ a comprehensive objective to evolve these problems, the evolutionary process may enter a state of random wander because of conflicts. From the point of view of Pattern Theorem, different objectives have different excellent patterns in a unified coding scheme. But the genetic operators especially the crossover operator can only consider patterns of one objective, and make the objective's excellent patterns grow exponentially, while other objectives stay in a state of random evolution. Consequently, the overall evolutionary efficiency will be very low and multiple objectives cannot get rapidly optimized at the same time.

In view of the shortcomings of the usual GA, the paper proposes an alternately evolving strategy from another standpoint. It adopts alternate method to optimize multiple objectives one by one circularly. Take the problem in formula 1 for example, and its detailed implementing steps are the following:

1. Take $f_1(X)$ as the only objective function to evolve, and turn to step 2 when the evolutional process arrives at a certain depth.
2. Take $f_2(X)$ as the only objective function to evolve, and turn to step 3 when the evolutional process arrives at a certain depth.

…

m. Take $f_m(X)$ as the only objective function to evolve, and turn to step $m+1$ when the evolutional process arrives at a certain depth.

$m+1$. Determine whether the m objective functions all achieve optimum or meet requirement. If so, terminate the procedure; if not, turn to step 1 and start the next alternate evolution.

In the above evolutionary process, every goal's evolution can have its own coding scheme and genetic operators. And it may be needed to transform coding scheme when the step turn to the next one. In order to describe each step's evolutionary depth, we can define alternate steplength S_k, which refers to the iteration times when the k th objective evolves independently in the step k. similarly, we can define the process of evolving all the m objectives one time as an alternate cycle.

The advantage in the foregoing alternate strategy is that every step is single-objective optimization, and when proper genetic operators are adopted for every objective, the convergent speed will be very fast. But every step doesn't consider other objectives, so these objectives' evolution will be in a random state in the step; that is, may converge or may diverge. In order to make the overall evolutionary trend be convergent, it is very important to control each step's evolutionary depth S_k. If evolve too deeply, the divergent trend will be greater than the convergent one; but if too shallowly, the evolutionary process will be very slow and be similar to the usual GA. Therefore, it is needed to adjust S_k enough times to search for a better one. The flow char of alternately evolving GA is shown below:

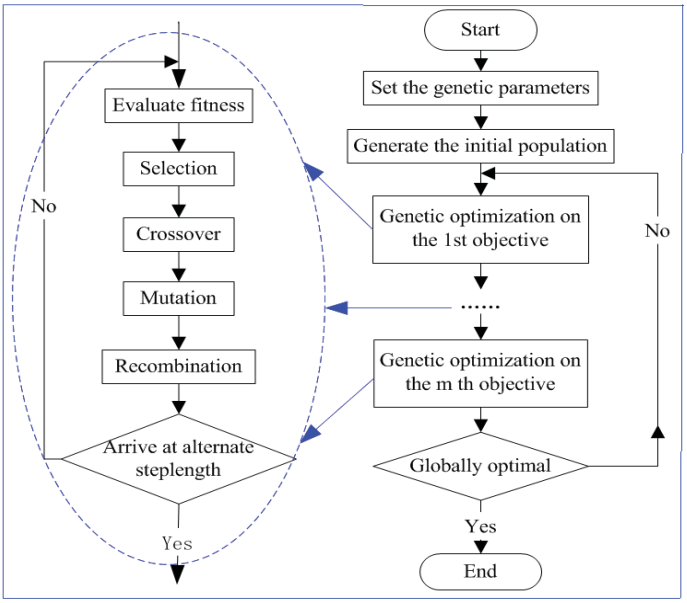

Figure 1. The flow char of alternately evolving GA.

3 Experiments

In order to verify the feasibility and high efficiency of the alternately evolving GA, take Sudoku as an example which the algorithm is used to solve. Sudoku is a numbers game popular in Japan, the UK and the USA. In essence, it is a special kind of Latin square and can be modelled by multi-objective combinational optimization. Because of Latin square's NP complete character, the solution space of Sudoku is quite huge, which causes many conventional algorithms to be ineffective. In the following paragraphs, we firstly give a brief introduction to Sudoku, and then adopt the usual GA and the alternately evolving GA to solve it respectively, and verify the excellence of the alternately evolving GA through the experimental results.

3.1 Sudoku

The most common Sudoku puzzle consists of 9×9 grid and 3×3 blocks for a total of 81 cells. One example of 9×9 Sudoku puzzle is shown in figure 2, in which different blocks are marked up by different colours. When a puzzle with a set of pre-filled numbers is designed, the task is to place the numbers 1 through 9 in each cell, such that the following rules hold:

1. Constraints on rows: each row must contain the numbers 1 - 9 one and only once;
2. Constraints on columns: each column must contain the numbers 1 - 9 one and only once;
3. Constraints on blocks: each block must contain the numbers 1 - 9 one and only once.

As can been seen, the objective of the puzzle is to make the nine groups of the numbers 1 - 9, totally 81 digits, rationally arrange on the 9×9 board, such that each row, column and block have no repeated digits.

	2	4	8	6	1			
1	7			2				5
7		1			3		8	
9				5				4
	8		9			5		1
3				8			1	7
			6	9	2	4	5	

Figure 2. 9×9 grid

3.2 The specific design of the alternately evolving GA for Sudoku

Considering Sudoku's characteristics, the following algorithm can be designed to solve it.

3.2.1 The coding scheme

Adopt the symbolic matrix coding scheme, which directly maps the 9×9 grid into a 9×9 matrix shown as follows. The matrix element x_{ij} refers to the number filled in the row i and column j of the 9×9 grid.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{19} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{29} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{91} & x_{92} & x_{93} & \cdots & x_{99} \end{bmatrix} \tag{3}$$

3.2.2 The comprehensive objective function

For each individual in the population, its fitness can be obtained in the following way:

1. Compare the nine numbers in the column i with the 1-9 series, and find the amount of the missing numbers, which is recorded as m_i .
2. Compare the nine numbers in the row j with the 1-9 series, and find the amount of the missing numbers, which is recorded as n_j .
3. Compare the nine numbers in the block k with the 1-9 series, and find the amount of the missing numbers, which is recorded as l_k .

If the total amount of the missing numbers $\sum_{i=1}^{9} m_i + \sum_{j=1}^{9} n_j + \sum_{k=1}^{9} l_k$ is regarded as the individual's comprehensive punishment, then the comprehensive fitness function can be defined as:

$$f = \frac{1}{\sum_{i=1}^{9} m_i + \sum_{j=1}^{9} n_j + \sum_{k=1}^{9} l_k} \tag{4}$$

3.2.3 The determination of the candidate set and the generating of the initial population

For each empty cell, because of the constraints from the column, row and block that the cell belongs to, it can only be filled in the number that has not existed in the column, row and block. These numbers are called candidate items, all of which make up candidate set. The candidate set of the cell in the column i and column j is recorded as c_{ij} . Naturally, the candidate set is an empty set if the cell has been pre-filled.

According to all the empty cell's candidate sets, the initial population can be generated in the following way:

1. Determine the candidate set of every empty cell on the 9×9 board.
2. For each individual, each cell's value can be determined this way: if the cell has been pre-filled, then its value is the pre-filled one; if the cell is empty, its value must be randomly chosen from its candidate set. By doing this, all the cells' values can be determined and an initial individual is generated.
3. Repeat step 2 to generate more individuals until the initial population has been yielded.

3.2.4 Optimization on rows

1) Fitness function. The optimization on rows means only considering the constraints on rows without taking into account the other two types of constraints. The optimization's goal is to meet the rows' constraints as much as possible. Therefore, the fitness function is:

$$f_1 = 1 \Big/ \sum_{i=1}^{9} m_i \tag{5}$$

Where $\sum_{i=1}^{9} m_i$ refers to the total punishment on rows.

2) Selection operator. Adopt the combined strategy of roulette wheel selection and preserving the fittest individuals to the next generation.

3) Crossover operator. Adopt one-point crossover, which treats each row of two-dimensional matrix as a point, and treats multi-row as multi-point, and then randomly determines a point to implement crossover operation.

4) Mutation operator. Adopt simple mutation and the mutated cell's new value is randomly generated from its candidate set.

3.2.5 Optimization on columns

Compared with optimization on rows, optimization on columns means only considering the constraints on columns without taking into account the other two types of constraints. The optimization's goal is to meet the columns' constraints as much as possible. Its fitness function is:

$$f_2 = 1 \Big/ \sum_{j=1}^{9} n_j \tag{6}$$

Where $\sum_{j=1}^{9} n_j$ refers to the total punishment on columns.

The genetic operators of the columns' optimization are similar to the rows'. Simply, it only needs to treat each column of two-dimensional matrix as a point when implementing crossover.

3.2.6 Optimization on blocks

Compared with the two optimizations above, the fitness function of optimization on blocks is:

$$f = 1 \Big/ \sum_{k=1}^{9} l_k \tag{7}$$

Where $\sum_{k=1}^{9} l_k$ refers to the total punishment on blocks.

The genetic operators of blocks' optimization are similar to the foregoing. Simply, it only needs to treat each block as a point and line up these points when implementing crossover.

3.2.7 Genetic parameters

All of the genetic parameters need debugging repeatedly to obtain proper values. Through many experiments, some parameters are set as follows:

The number of the population's individuals	300
The preserving probability of the fittest individuals	0.1
The crossover probability	0.85
The mutation probability	0.03
The number of possible mutated points	81
The three kinds of alternate steplength for the three objectives	10, 10, 10

3.3 Experiment results and their comparison

For the Sudoku puzzle in figure 2, adopt the usual GA and the alternately evolving GA to solve it respectively. Their solving results are shown as follows:

5	2	4	8	6	1	9	7	3
1	7	3	4	2	9	8	6	5
6	9	8	7	3	5	1	4	2
7	5	1	2	4	3	6	8	9
9	3	6	1	5	8	7	2	4
4	8	2	9	7	6	5	3	1
2	4	5	3	1	7	3	9	6
3	6	9	4	8	5	2	1	7
8	1	7	6	9	2	4	5	8

Figure 3. The result of the usual GA. The cells marked up by other colours indicate there are repeated numbers

5	2	4	8	6	1	3	7	9
1	7	3	4	2	9	8	6	5
6	9	8	7	3	5	1	4	2
7	5	1	2	4	3	9	8	6
9	3	6	1	5	8	7	2	4
4	8	2	9	7	6	5	3	1
2	4	5	3	1	7	6	9	8
3	6	9	5	8	4	2	1	7
8	1	7	6	9	2	4	5	3

Figure 4. The result of the alternately evolving GA.

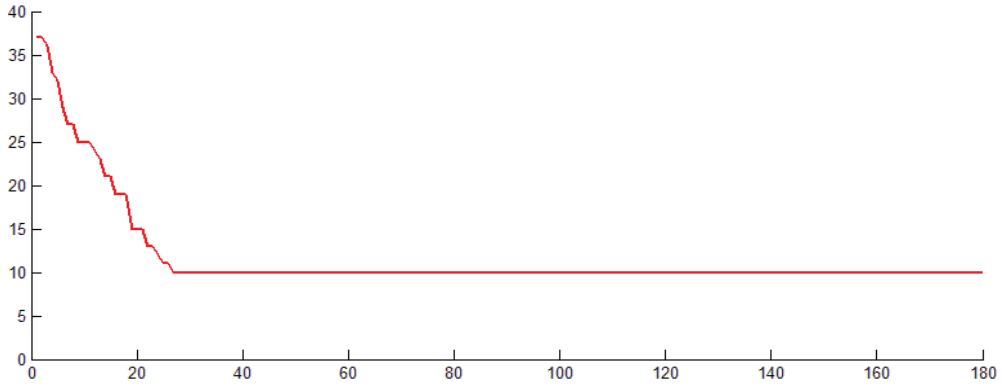

Figure 5. The comprehensive punishment's change in the solving process of the usual GA. The graph's abscissa refers to evolving generations, and the ordinate refers to comprehensive punishment. It can be seen from the graph that the population's evolution falls into local optimum at about 30 generations.

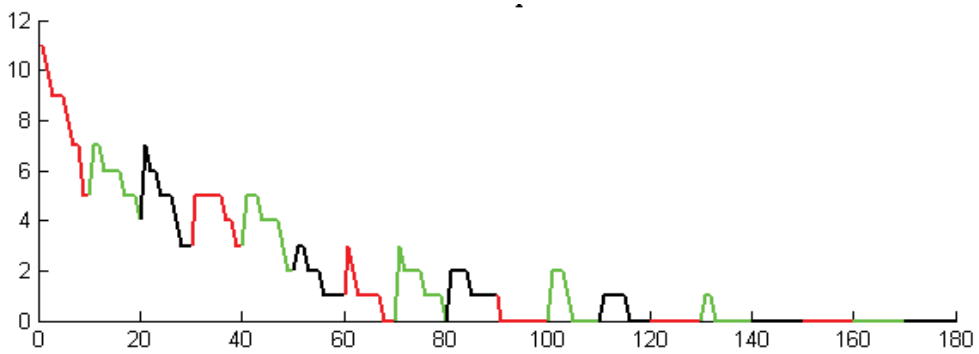

Figure 6. The change of each objective's punishment in the solving process of the alternately evolving GA. The graph's abscissa refers to evolving generations, and the ordinate refers to row objective's punishment, column objective's punishment or block objective's punishment. The curve's fluctuation in the graph is the very result of alternate evolution, in which the red curve represents the row objective's evolution and its corresponding ordinate refers to row objective's punishment; the green curve represents the column objective's evolution and its corresponding ordinate refers to column objective's punishment; the black curve represents the block objective's evolution and its corresponding ordinate refers to block objective's punishment. As can be seen from the graph, although every evolution's alternation always makes punishment fluctuate, the general trend of the three kinds of punishment is convergent. And when the three kinds of alternate steplength are all set to 10, the three kinds of punishment all converge to 0 at about 150 generations, that is to say, the population arrives at optimum.

By comparing the four figures above, it can be known that:

1. In the solving process of the alternately evolving GA, although every evolution's alternation always makes punishment fluctuate, the general trend of the three kinds of punishment is convergent and the population will evolve to optimal state in the end when all of the genetic parameters are set properly. These show that the alternately evolving GA is feasible.
2. The evolution of the usual GA falls into local optimum at about 30 generations, while the one of the alternately evolving GA can converge to global optimum. Therefore, the convergent capability of the alternately evolving GA is better than the one of the usual GA.

3. The alternately evolving GA need to evolve every objective one by one circularly, so for some simple or small-scale problems, its convergent time will be longer than the usual GA's. But for some complex and large-scale problems, the usual GA always converges slowly and even falls into local optimum, so at this moment the advantage of the alternately evolving GA in convergent time will present gradually as problems' complexity increases.
4. Sudoku is a typical multi-objective combinatorial optimization problem. It can be predicted that the alternately evolving GA is also effective for other multi-objective combinatorial optimization problems.

3.4 Sensitivity analysis

To confirm the effect of each objective's evolutionary depth to the general convergent performance, set steplength to 3 and 30 respectively and other parameters keep constant. The results of the two kinds of experiment are shown as follows:

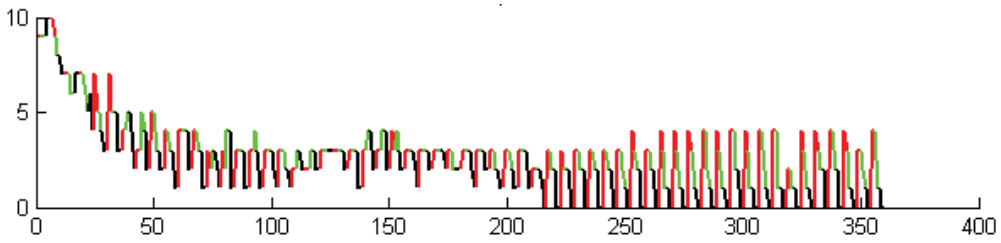

Figure 7. The change of each objective's punishment in the solving process of the alternately evolving GA when the three kinds of alternate steplength are all set to 3.

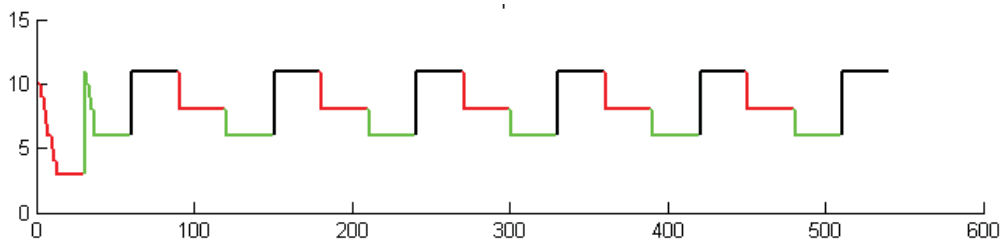

Figure 8. The change of each objective's punishment in the solving process of the alternately evolving GA when the three kinds of alternate steplength are all set to 30.

As can be seen from the two figures above, when the three kinds of alternate steplength are all set to 3, each objective's evolutionary degree is so shallow that the general evolution stays in a state of random shake, and the general performance is similar to the usual GA's; when the three kinds of alternate steplength are all set to 30, each objective's evolutionary degree is so deep that the population may fall into local optimum at the first evolution and all of individuals may be exactly the same, and the later evolution is a uniform shake. These indicate that each objective's evolutionary depth has a great effect to the general convergent performance, so it is necessary to experiment enough times to find the best steplength.

4 Conclusions

This paper considering the shortcomings of the usual genetic algorithm when solving multi-objective combinatorial optimization problems, proposes an alternately evolving strategy. The solving results of the Sudoku puzzle indicate that the alternately evolving GA's general convergent capability is better than the usual GA's, and the advantage of the former in convergent speed will present gradually as problems' complexity increases. These show that the alternately evolving GA is feasible and excellent.

The alternately evolving GA adopted in solving Sudoku, is just an easy model of the alternately evolving GA, and there is much room for improvement, such as:

1. Adopt adaptive steplengths, which can change with objectives and evolving state;
2. Adopt different coding schemes and genetic operators for different objectives.

These measures can improve general convergent capability and speed.

5 References

[1] Timo Mantere, Janne Koljonen. Solving, rating and generating Sudoku puzzles with GA. IEEE Congress on Evolutionary Computation, 2007, 1382-1389.

[2] Miguel Nicolau, Conor Ryan. Solving Sudoku with the GAuGE system. Springer Verlag, 2006, 213-224.

[3] Rhydian Lewis. On the combination of constraint programming and stochastic search: the Sudoku case. Springer-Verlag, 2007, 96-107.

[4] Alberto Moraglio, Julian Togelius. Geometric particle swarm optimization for the Sudoku puzzle. GECCO'07, July 7–11, 2007, 118-125.

[5] Zong Woo Geem. Harmony Search Algorithm for Solving Sudoku. Springer Verlag, 2007, 371-378.

[6] Zhanghui Chen, Xiaohui Huang, Wenyi Ren, Lie Kang. The diploid code genetic algorithm used to solve UTP. Unpublished.