Multi-objective Differential Evolution Algorithm for Multivalued Logic Networks Optimization

by

Jian SUN

A dissertation

submitted to the Graduate School of Science and Engineering for Education in

Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

Gofuku 3190, Toyama-shi, Toyama 930-8555 Japan

2019

Abstract

Multivalued Logic (MVL) originates from Boolean Logic, and it is the development and generalization of the latter. MVL networks' models abstracted from a large number of practical applications need to be optimized to improve their performance. Many mathematical and physical methods have been used to solve the problems in the early stages of research. In recent years many intelligent computational methods have been introduced into research and they have made great progress so as to deal with the local minimum, run smoothly in the global field, reach some high-quality solutions in a reasonable time and show their robustness. Nevertheless they have a common flaw: their measure of the solutions is unique and incomplete. In other words, they are all the methods based on a single-objective function, while in real world optimization problems need usually be evaluated with several criteria.

We first propose a novel algorithm called bi-objective elite differential evolution (BOEDE) to optimize MVL networks in this thesis. It is a multiobjective algorithm completely different from all the existing single-objective optimization ones, and multiobjective is one of the most important innovations of it. Two objective functions, incorrectness and optimality, are put into evaluating the fitness of individuals in the evolution simultaneously. Incorrectness measures the accuracy of solutions inversely, and optimality implies the cost of solutions. Another important innovation is that a hierarchical storage structure is built for archive population and each rank has a fixed size. Moreover, a characteristic updating method corresponding to the structure is designed. Subsequently because of the particularity of MVL network problems, the performance of BOEDE to solve them is further improved by some techniques, such as distinguishing elite solutions and Pareto optimal solutions strictly, modifying the

method of dealing with illegal variables, reusing better solutions of the previous generation, and so on.

The simulations show that our algorithm can not only collect a great number of elite solutions, but also collect a great number of Pareto optimal ones with different ranks, which are sufficient to provide decision support for a variety of applications. The comparison simulation results also indicate that the solutions of the proposed algorithm have much richer phenotypes, richer genotypes and lower cost than those of the existing algorithms. In general, BOEDE is significantly better than them.

The content of the thesis is organized as follows: in Chapter 1 we outline research topic, research history and research purpose; in Chapter 2 and Chapter 3, we introduce the problem to be studied: optimization of MVL networks, and give some basic knowledge about MVL networks and optimization in order to understand the proposed algorithm; in Chapter 4 we illustrate basic DE algorithm and some techniques which are utilized to solve the above problem; in Chapter 5 we propose a novel DE algorithm to optimize MVL networks, and expatiate on its innovations; in Chapter 6 we give a lot of detailed experiments and make some contrast experiments, which all prove the proposed algorithm is more effective and more efficient than the existing algorithms; in the last chapter we summarize this research and put forward a further research plan.

Contents

Chapter 1
Introduction to Research Topic

1.1 Research background on the topic

Multivalued logic (MVL) technology evolved from binary Boolean logic technology. In traditional Boolean logic systems a variable has only two values, 1 and 0, to represent two specific states, such as right and wrong in logic, on and off in circuits, high and low in level. While in MVL systems, a variable can have many values to express many different states [1]. For examples, in the Post algebra it can express three symbols: positive, zero and negative; in computer operating systems it can express five kinds of process state: create, ready, execute, block, and exit. The MVL technology has the potential to increase the information capacity of each single line obviously [2]–[6] and consequently it has a great advantage in the logic circuit design, reliability analysis, and optimization [7]–[11]. With the expansion of circuit integration, the optimization of an MVL network can shrink the size of programmable logic arrays effectively, reduce the number of interconnections between chips in orders of magnitude [8], [12]–[14], thereby improving the circuit integration [15], speeding up hardware [16], and enhancing the ability of fault tolerance [17].

Recently, the MVL network technology has also been applied in many other ways which are promising and attractive, such as pattern recognition [18], image processing [19], associative memories [20] and data mining [21]. Especially it is also applied to multivalued neuron (MVN), which is based on the multivalued threshold logic and can match inputs and outputs arbitrarily by the partially defined multivalued function [22]–[25]. It is necessary and important to optimize MVL networks [26]–[28].

1.2 Research progress on the topic

In order to optimize MVL networks, the mathematical and physical methods, such as algebraic method [29], circuit method [30], hyper-planes method [31], have been used. Although these early methods can provide some optimized solutions, they are extremely time-consuming, even in orders of days and they can only solve the small-size problems [32]–[34]. As an alternative, direct cover (DC) [35]–[38] can generate an efficient cover directly for a given MVL network, omitting the intermediate procedure of creating prime implicants. DC requires much less computational time than those methods based on implicants. MVL networks optimized by DC are without any error as the ones before optimizing. However this technique aims at accuracy, regardless of the number of optimized gate circuits. That is to say, DC is poor in saving the cost of manufacturing and integration of circuits. Subsequently in recent years, DC is introduced into some intelligent algorithms [39] [40], and these hybrid algorithms can simplify MVL networks. Especially the ant colony optimization (ACO) algorithm combines DC technique to improve the performance of MVL networks, which is called ACO-DC [41] [42]. ACO-DC is a state-of-the-art algorithm, which first uses some ants to decompose an MVL network, and then uses DC technique to mimic each sub-network. It completes the cover of MVL networks successfully, and reduces the number of gate circuits effectively, yet not enough. Meanwhile a new learning MVL network [43]–[45] is put forward to improve the performance of the networks based on algebra [46] [47], and several kinds of technologies, such as error back-propagation [48] and non-back-propagation [49] [50], are introduced into the model. It can accumulate priori knowledge about targets and processes, so as to evolve more effectively. In fact, a learning MVL network using error backpropagation requires some presuppositions: (1) the simulated function curve must be continuous and differentiable at each point, and (2) the values of some parameters, e.g., weight and threshold are pre-assigned. It is obvious that these conditions are uncertain.

On the other hand, a learning MVL network based on non-back-propagation such as local search [51] [52] and simulated annealing [53] needs no such prior knowledge,

but it has its disadvantages: it is easy to fall into a local minimum and easy to converge prematurely. To jump out of a local minimum trap, a great deal of research has been carried out. A stochastic dynamic factor is introduced into a traditional local search method, which can permit the search to go a little farther along the error direction in order to escape from a local minimum when the search encounters a trap [54] [55]. A clonal selection algorithm incorporated with chaos is proposed to maintain the diversity of populations and speed its evolutionary efficiency [56].

The above-mentioned search methods have proved to be better than the traditional local search through a large number of experiments. They can deal with the local minimum, run smoothly in the global field, then reach some high-quality solutions in a reasonable time and show their robustness. Nevertheless they have a common flaw: their measure of the solutions is unique and incomplete. In other words, they are all the methods based on a single objective function. Real world optimization problems need usually be evaluated with several criteria, but not only one, and these criteria maybe in conflict with each other [72-77]. Specifically the optimization of MVL networks includes two objectives: accuracy and optimality. Accuracy is the most important measure of a system and optimality reflects the cost of a system. The methods mentioned above only take the accuracy into consideration, but ignore the optimality. They always try their best to find the best solutions with the least error, regardless of the cost of the solutions, which is clearly unreasonable. A modified error function combing accuracy and optimality was presented to evaluate the individuals of MVL networks by using a weighted sum [57]. It must be emphasized that these two evaluation criteria used in [57] are totally unequal in the evaluation function: the accuracy plays a dominant role, while the optimality starts to work only when the accuracy of an individual is the same or almost the same as that of the other. This algorithm is strongly interfered by the weights that are not determined objectively. Consequently it is a single-objective one in essence.

1.3 Purpose and method

In this paper, we for the first time propose a true multiobjective method to

optimize MVL networks and try to find the optimal solutions. First and foremost, two independent objective evaluation functions are applied into the problems. They are similar to the previous definitions, yet we utilize them completely equally in our research, without any subjective parameters. Second, in single-objective problems the best solution appears when the objective function is the best while in multiobjective problems there are complex internal relations among various objectives. It would not be a subjective decision whether a solution is abandoned or not in solving MVL network problems. Sufficient optimal solutions should be collected in order to provide the needed decision support for practical applications. The optimal solutions includes two kinds, the elite solutions with no error in accuracy and the Pareto optimal solutions whose accuracy and optimality are compromised with each other. The solution with both the best accuracy and the best optimality may not be turned into reality for the reason of the cost or technical limitations. Thus the best accurate solutions should have some different optimality. Furthermore, some products hope to reduce the cost by allowing some non-critical errors. Therefore, Pareto optimal solutions should also have some different ranks. Altogether, the two kinds of solutions should be uniformly distributed along the two evaluation functions.

In MVL networks, the objective functions have simple phenotypes in which their variable values are discrete and limited, while the decision vectors corresponding to the objective functions have rich genotypes in which a decision vector has a number of variables. Because of these features, a novel algorithm based on the traditional differential evolution (DE) is proposed to optimize MVL networks.

1.4 Outline

In this chapter we first elaborate why we chose MVL optimization problems as our research task, that is, the function and significance of this topic. We choose it because of wide applications of MVL networks and importance of optimize them. Secondly we summarize the existing research methods and their achievements on this topic, then put forward some ideas to improve optimization, using DE algorithm to minimize MVL networks with two objectives.

The remaining content is organized as follows: in Chapter 2 and Chapter 3, we introduce the problem to be studied: optimization of MVL networks, and give some basic knowledge about MVL networks and optimization in order to understand the proposed algorithm; in Chapter 4 we illustrate basic DE algorithm and some techniques which are utilized to solve the above problem; in Chapter 5 we propose a novel DE algorithm to optimize MVL networks, and expatiate on its innovations [78]; in Chapter 6 we give a lot of detailed experiments and make some contrast experiments, which all prove the proposed algorithm is more effective and more efficient than the existing algorithms; in the last chapter we summarize this research and put forward a further research plan.

Chapter 2
MVL Networks

2.1 Structure of MVL networks

The concept, multivalued logic, should be described ahead of the concept of MVL network. Generally, a variable of traditional two-valued systems has only two values -1 and 0 - representing two specific states. While in the multivalued systems, a variable can have many values and so it can express many states. MVL systems evolve from two-valued logic system, and apparently they have more information storage on single wire than traditional systems. Without loss of generality, we define it as an R- valued system.

An R-valued logic is used to define a MVL system specifically, in which a variable can take values from 0 to R-1 (R∈N and R≥2) to express R kinds of states.

An MVL network [43] [44] is a three-layer feed-forward network, shown in Fig. 2.1. This network consists of five parts: input, three intermediate layers, and output. The network is multivalued because its variables are all multivalued. X is an input vector, which includes n input variables, i.e.,

$$x = \{x_1, x_2, x_3, \dots, x_n\} \tag{2.1}$$

where $x_i \in \{0,1,2,\dots,R-1\}$ ($i = 1,2,\dots,$ n and R ∈ N and R ≥ 2) .

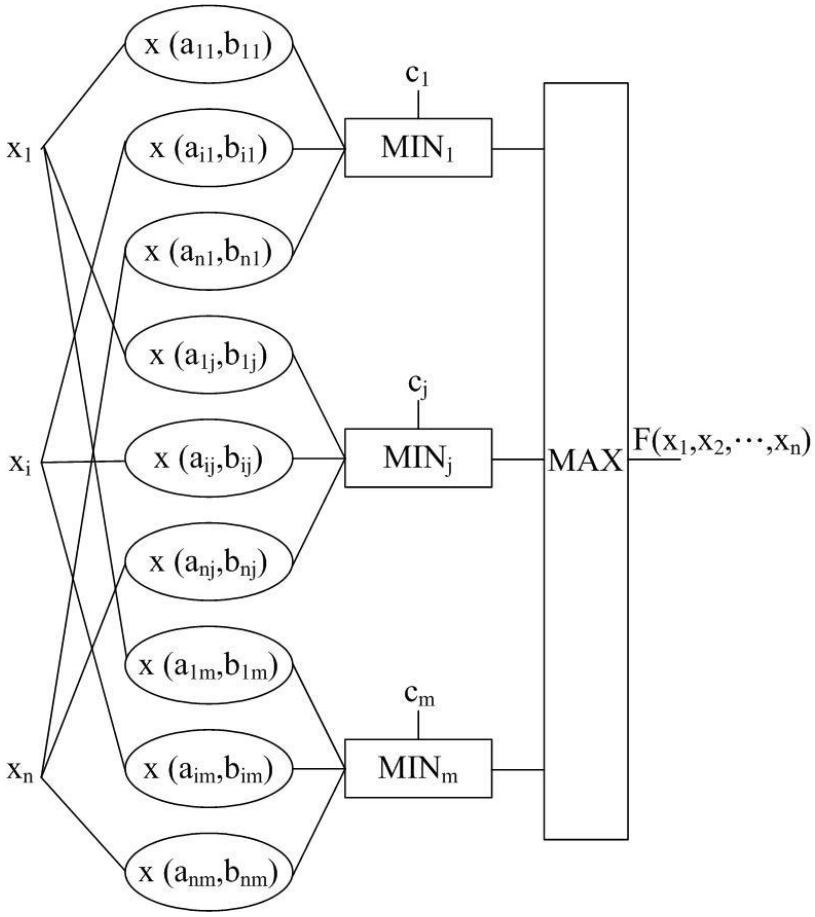

Fig. 2.1. Structure of MVL networks.

The first layer in an MVL network is considered to be the Literal operation, and $x_i(a_{ij}, b_{ij})$ is called a Literal operation node, represented as

$$x_i(a_{ij}, b_{ij}) = \begin{cases} R-1 & a_{ij} \le b_{ij} \\ 0 & otherwise \end{cases} \quad (2.2)$$

$where\ i = 1,2,\dots,n\ and\ \ j = 1,2,\dots,m\ (m = R^n - 1)$. If $a_{ij} \le b_{ij}$, the value of the node is R − 1; otherwise the value is 0. Here a_{ij} and b_{ij} are both window parameters and they are certainly multivalued ($a_{ij}, b_{ij} \in \{0,1,2,\dots,R-1\}$); the subscript i indicates that the input comes from the i^{th} x and the subscript j indicates that the value of the node goes to the j^{th} MIN gate. The Literal operation is shown in

Fig. 2.2.

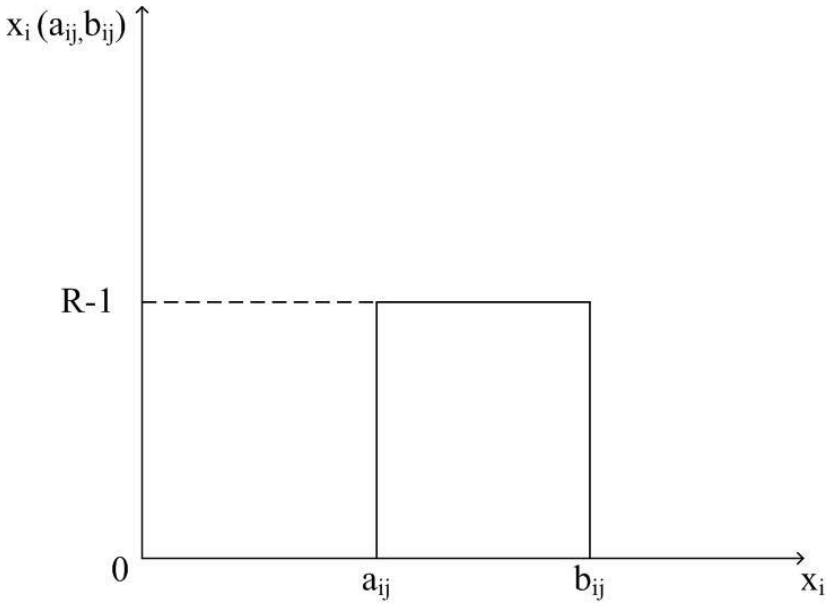

Fig. 2.2 Definition of literal operation.

The second layer is MIN gates. As the name implies, the operation is to get the smallest values of all the members including all the input values from Layer 1 and a biasing parameter c which is also multivalued ($c_j \in \{0,1,2,\dots,R-1\}$). The MIN operation is shown as follows:

$$MIN_j = the\ smallest\ value\ of$$

$$\left(c_j, x(a_{1j}, b_{1j}), \dots, x(a_{ij}, b_{ij}), \dots, x(a_{nj}, b_{nj})\right) \qquad (2.3)$$

The third layer MAX describes the MAX operation, i.e.,

$$MAX = the\ largest\ value\ of\left(MIN_1, \dots, MIN_j, \dots, MIN_m\right) \qquad (2.4)$$

MAX is the largest value of all the MIN operations. $F(x_1, x_2, \dots, x_n)$ is the output of the network, a value assigned by MAX. $(x_1, x_2, \dots, x_n)$ represents input variables corresponding to the value of MAX.

The MVL network with MAX operation, MIN operation and literal operation can simulate any multivalued function based on canonical expansion in a sum-of-products form [58] which is given as follows:

$$F(x_1, x_2, \dots, x_n) = \sum_{e_1, e_2, \dots, e_n} F(e_1, e_2, \dots, e_n) \cdot x_1(e_1, e_1) \cdot x_2(e_2, e_2) \cdot \dots \cdot x_n(e_n, e_n) \quad (2.5)$$

where $x_1, x_2, \dots, x_n$ are R-valued variables, $e_i \in \{0,1,2, \dots, R-1\}$,

i $= 1,2, \dots,$ n and $F(e_1, e_2, \dots, e_n) \in \{0,1,2, \dots, R-1\}$.

2.2 A simple optimization example

The above formula is very abstract and it will be illustrated with a classical example. Table 2.1 presents a truth table of a 2-variable 4-valued function. X is the input vector including two variables which can take a value from a limited set {0, 1, 2, 3}. Thus there are 16 different input combinations and 16 corresponding output results. Then the following canonical expansion is acquired according to the Eq. (2.5) and the MVL network, seen Fig. 2.3, is constructed to simulate the MVL function.

Table 2.1 Truth table of a 2-variable 4-valued function.

x_1 \ x_2	0	1	2	3
0	1	0	0	1
1	1	1	0	1
2	3	3	3	1
3	0	1	0	2

$$\begin{aligned} F(x_1, x_2) = \; & 1 \cdot x_1(0,0)x_2(0,0) + 1 \cdot x_1(0,0)x_2(3,3) \\ & + 1 \cdot x_1(1,1)x_2(0,0) + 1 \cdot x_1(1,1)x_2(1,1) \\ & + 1 \cdot x_1(1,1)x_2(3,3) + 3 \cdot x_1(2,2)x_2(0,0) \\ & + 3 \cdot x_1(2,2)x_2(1,1) + 3 \cdot x_1(2,2)x_2(2,2) \\ & + 1 \cdot x_1(2,2)x_2(3,3) + 1 \cdot x_1(3,3)x_2(1,1) \\ & + 2 \cdot x_1(3,3)x_2(3,3) \end{aligned} \tag{2.6}$$

The number of the MIN gates usually reflects the cost of hardware in production. The network described by Eq. (2.6) has been simplified by mathematical and physical method and the result can be given as follows:

$$\begin{aligned} F(x_1, x_2) = \; & 1 \cdot x_1(0,2)x_2(0,0) + 1 \cdot x_1(0,3)x_2(3,3) \\ & + 1 \cdot x_1(1,3)x_2(1,1) + 2 \cdot x_1(3,3)x_2(3,3) \\ & + 3 \cdot x_1(2,2)x_2(0,2) \end{aligned} \tag{2.7}$$

The simplified structure after optimization is shown in Fig. 2.4. This method has been verified to be correct [59].

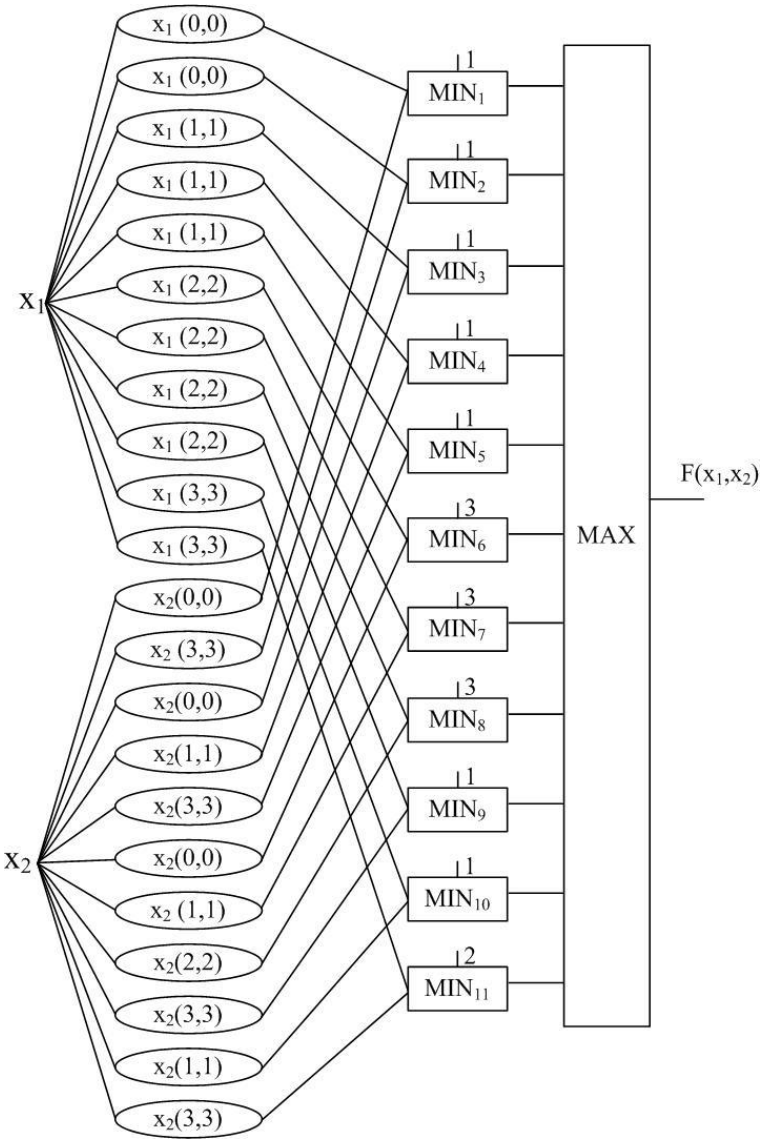

Fig 2.3 MVL network of Table 2.1 before optimization.

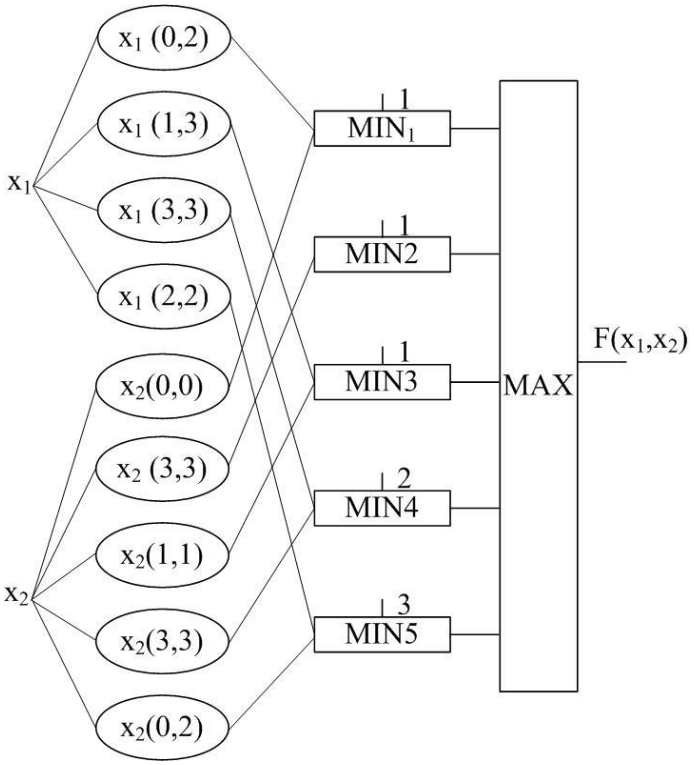

Fig 2.4 MVL network of Table 2.1 after optimization.

Comparing the two figures which are before and after the simplification respectively, we can know the network before optimization has 11 MIN gates and the one after optimization has only 5 MIN gates, and the latter is simpler and less costly than the former. Generally, the number of MIN gate represents the manufacturing cost. Our task is using some algorithm to simulate the initial inputs of a network as the ones of Fig. 2.3 at the beginning and to obtain the ideal structure of the network like the one of Fig. 2.4 in the end.

Furthermore, we maybe obtain several optimizing results like (2.7) and we can select one which is cheaper in cost and easier in realization. Although the number of MIN gate represents the manufacturing cost, at the same time the high degree of integration also brings huge technical problems, that is to say, its technical cost maybe

higher. Take another truth table 2.2 as an example.

Table 2.2 Another Truth table of a 2-variable 4-valued function.

x_1 \ x_2	0	1	2	3
0	2	1	1	3
1	1	2	1	3
2	1	1	1	3
3	1	3	1	0

Its canonical functions is shown as follows:

$$\begin{aligned} F(x_1, x_2) = {} & 2 \cdot x_1(0,0)x_2(0,0) + 1 \cdot x_1(0,0)x_2(1,1) + 1 \cdot x_1(0,0)x_2(2,2) \\ & + 3 \cdot x_1(0,0)x_2(3,3) + 1 \cdot x_1(1,1)x_2(0,0) + 2 \cdot x_1(1,1)x_2(1,1) \\ & + 1 \cdot x_1(1,1)x_2(2,2) + 3 \cdot x_1(1,1)x_2(3,3) + 1 \cdot x_1(2,2)x_2(0,0) \\ & + 1 \cdot x_1(2,2)x_2(1,1) + 1 \cdot x_1(2,2)x_2(2,2) + 3 \cdot x_1(2,2)x_2(3,3) \\ & + 1 \cdot x_1(3,3)x_2(0,0) + 3 \cdot x_1(3,3)x_2(1,1) + 1 \cdot x_1(3,3)x_2(2,2) \end{aligned} \tag{2.8}$$

It has 15 MIN gates. After optimization, we can get results given as follows:

$$\begin{aligned} F(x_1, x_2) = {} & 1 \cdot x_1(0,2)x_2(0,3) + 2 \cdot x_1(0,0)x_2(0,0) \\ & + 2 \cdot x_1(1,1)x_2(1,1) + 3 \cdot x_1(1,1)x_2(1,1) \\ & + 3 \cdot x_1(3,3)x_2(0,2) \end{aligned} \tag{2.9}$$

$$\begin{aligned} \mathrm{F}(x_1, x_2) \;=\;& 1 \cdot x_1(0,2)x_2(0,3) \;+\; 1 \cdot x_1(0,2)x_2(2,3) \\ &+\; 2 \cdot x_1(1,1)x_2(1,1) \;+\; 2 \cdot x_1(0,0)x_2(0,0) \\ &+\; 3 \cdot x_1(1,1)x_2(3,3) \;+\; 3 \cdot x_1(3,3)x_2(0,2) \end{aligned} \tag{2.10}$$

$$\begin{aligned} \mathrm{F}(x_1, x_2) \;=\;& 1 \cdot x_1(0,2)x_2(0,3) \;+\; 2 \cdot x_1(1,1)x_2(1,1) \\ &+\; 3 \cdot x_1(1,1)x_2(3,3) \;+\; 3 \cdot x_1(3,3)x_2(0,1) \end{aligned} \tag{2.11}$$

Functions (2.9), (2.10) and (2.11) have 5, 6 and 4 MIN gates, respectively. But functions (2.9) and (2.10) have no error with the canonical function (2.8), while function (2.11) only has a little error which is 1 in value. We their implementations are different. We don't decide which one is the easiest to implement and some certain degree of error can be compatible by different systems. Therefore we try to get as many optimized solutions as possible. They are divided into two categories: ones are solutions with no error, the others are solutions with few errors which can be tolerant. They are chosen by different application systems according to different demands. Generally, an MVL network is a special kind of artificial neural networks [79-88].

Chapter 3
Multiobjective Optimization

3.1 Multiobjective optimization

Multi-objective optimization problems (MOPs) are ubiquitous in the real world. For example, during the period of producing a soft system such as mobile phones, cars, the characteristics such as performance, reliability, stability, compatibility, cost and so on should be taken into considering. Often the cost of such systems is to be minimized, while maximum performance is desired. We can take all of these as evaluation functions or we can choose some important characteristics as the evaluation functions and translate the others into constraints. Formally, a minimum optimization problem is defined as follows:

$$\begin{aligned} &\text{minimize} && \mathrm{y} = \mathrm{F}(\mathrm{x}) = (f_1(x), f_2(x), \dots, f_k(x)) \\ &\text{subject to} && \mathrm{E}(\mathrm{x}) = (e_1(x), e_2(x), \dots, e_m(x)) \leq 0 \\ &\text{where} && \mathrm{x} = (x_1, x_2, \dots, x_n) \in X \\ & && \mathrm{y} = (y_1, y_2, \dots, y_k) \in \mathrm{Y} \end{aligned} \tag{3.1}$$

An MOP includes a set of k objective functions, a set of m constraints functions. Here

x is the decision vector, X is the decision space, y is the objective vector, Y is the objective space, F(x) is the total objective function consist of k sub-objective functions, E(x) is the total constraint functions consist of m sub-constraint functions.

When k=1, the optimization problem has only one objective function and it is called a Single-objective Optimization Problem (SOP); otherwise it is called a Multiobjective Optimization Problem (MOP). When m=0, the optimization problem is unconstrained optimization; otherwise it is constrained optimization [60].

3.2 Pareto optimal solutions

The optimization goal is to find a set of decision variables in the decision space which can satisfy the constraints and make the objective functions appear optimal. In SOP, the set of corresponding decision variables is optimal when the unique objective function is optimal. But in MOP there are more than one objective functions and the ideal case is that when all the objective functions obtain the minimum at the same time, the corresponding decision variables are optimal. In fact it is almost impossible. Almost all the problems in the real world involves simultaneously several evaluation objectives. These objectives are incommensurable and often competitive with each other. The improvements to some functions often result in decreases in the performance of others.

In Fig. 2.3, assume that f_1 and f_2 are the two objective functions and their values are as small as necessary. Compare points A and point C: $f_1(A) < f_1(C)$ and $f_2(A) < f_2(C)$, which is formally expressed that the decision vector to which point A corresponds dominates (symbol $\succ$) that to which point C corresponds. This is Pareto dominance, defined as follows [60]:

Pareto dominance (minimize)

$$\exists \mathrm{u} = (u_1, u_2, \dots, u_n) \in X$$

$$\exists v = (v_1, v_2, \dots, v_n) \in X$$

u $\succ$ y(u dominate y)

$$\forall i = \{1,2,\dots,n\}: f_p(u_i) \le f_p(v_i) \ \ (p = 1,2,\dots,k)$$

$$\text{and } \exists j = 1,2,\dots,n: f_p(u_j) \le f_p(v_j) \ \ (p = 1,2,\dots,k) \qquad (3.2)$$

Compare points A and B: $f_1(A) < f_1(B)$ and $f_2(A) < f_2(B)$, thus we cannot judge which is superior and which is inferior. The decision vectors to which points A and B correspond to respectively don't dominate (symbol $\nsucc$) with each other, which is indifferent, defined as follows [60]:

For any two decision vectors u and v

$$\exists u = (u_1, u_2, \dots, u_n) \in X$$

$$\exists v = (v_1, v_2, \dots, v_n) \in X$$

u $\backsim$ v (u is indifferent to v)

$$u \nsucc v \quad \text{and} \quad v \nsucc u \qquad (3.3)$$

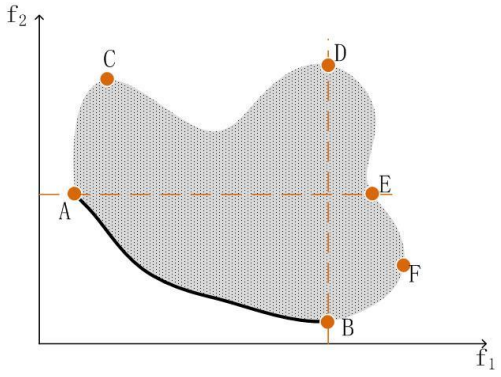

Fig. 3.1 Schematic diagram of Pareto dominance.

3.3 Rank of solutions

As mentioned above, in this study optimal solutions include elite solutions with different optimality and Pareto optimal solutions with different ranks. For the former, rank indicates their optimality. For the latter, rank indicates hierarchy of solutions, similar to the concept “level” of non-dominated sorting in NSGA II [61]. Fig. 2 illustrates the concept of rank. In this figure, it is obvious that the decision vectors to which points A and B correspond respectively are indifferent with each other, and they dominate all the other vectors. Therefore, they are of course Pareto optimal solutions with the first rank. Another decision vectors to which points C, D and E correspond respectively are also indifferent with each other. And if points A and B are removed, the decision vectors to which points C, D and E correspond are Pareto optimal solutions. Thus they are of the second rank. And so on, all Pareto optimal solutions with different ranks are obtained. Taking into account the actual situations such as allowing some non-critical errors to reduce production costs, the existing manufacturing technology which cannot achieve the ideal high-precision solutions, more optimal solutions with different ranks have to be collected for the decision maker [60] [62].

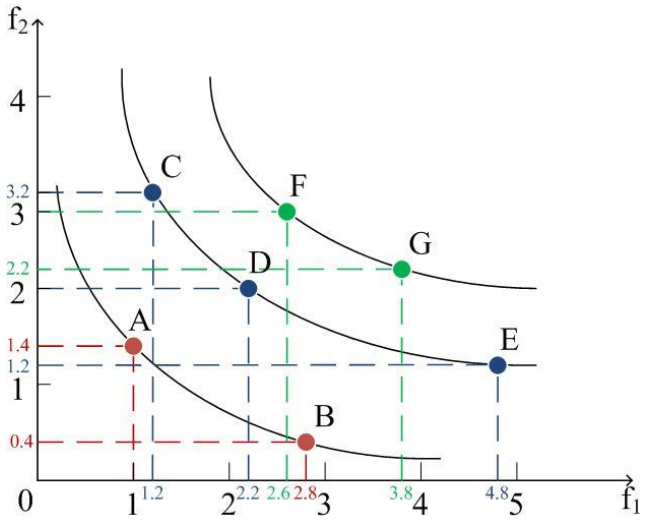

Fig. 3.2. Pareto optimal solutions with different ranks.

Chapter 4

Differential Evolution

4.1 Development and characteristics of DE

Optimization problems are ubiquitous in science and engineering. What shape gives an airfoil maximum lift? Which polynomial best fits the given data? Which configuration of lenses yields the sharpest image? Without question, very many researchers need a robust optimization algorithm for solving the problems that are fundamental to their daily work.

Ideally, solving a difficult optimization problem should not itself be difficult, e.g., a structural engineer with an expert knowledge of mechanical principles should not also have to be an expert in optimization theory just to improve his designs. In addition to being easy to use, a global optimization algorithm should also be powerful

enough to reliably converge to the true optimum. Furthermore, the computer time spent searching for a solution should not be excessive. Thus, a genuinely useful global optimization method should be simple to implement, easy to use, reliable and fast.

Since the 1980s, evolutionary algorithms have become a hotspot in the field of computer science, which basically satisfies the above requirements. Evolutionary algorithms include Genetic Algorithm (GA), Evolution Strategies (ES), Evolutionary Programming (EP), Genetic Programming(GP), Gene Expreceion Programming (GEP), and so on.

DE is a simple, yet very efficient, global optimization algorithm which originated with evolutionary algorithms. It was proposed by Storn and Prince in 1995 [64] to solve the Chebyshev polynomials. As a further improved version of the evolutionary algorithm, DE has a wide applicability to optimization problems of on which the functions bases do not have the characteristics of being differentiable, continuous, unimodal, unlike the classical optimization algorithms. Thus it also has been successful in many fields such as data mining, pattern recognition, digital filter design, artificial neural networks, and electromagnetics [65]–[68]. It can solve not only SOPs but also MOPs [69]–[71].

DE has have the following characteristics:

- It uses the code of genetic variables after optimization as the object of being searching, which make the algorithm effective not only in numerical optimization problems but also in non-numerical optimization ones;

- It only utilizes fitness functions and does not need any other information, such as the specific values of objective functions, which widens the applications to a variety of optimization problems;

- It has a population search strategy rather than a single search one, which allows the algorithm to search in parallel, then obtain many optimized results, especially for multiobjective decision-making problems;

- It uses a stochastic search mechanism, which enhance the robustness of the

algorithm.

4.2 Basic DE

The basic idea of evolutionary algorithms to solve problems has been inspired by biological evolution which involves reproduction, mutation, recombination and selection. The algorithms generate the first population randomly, of which the individuals as the candidate solutions of the problem. Then mutation operator, recombination operator work on the population, which produce a new solution population. At the last it applies selection operator into the population and select the solutions which can fit the surroundings well according to their values of fitness functions. GA is the most famous and classic one of the evolutionary algorithms and its basic flowchart is shown in Fig. 4.1.

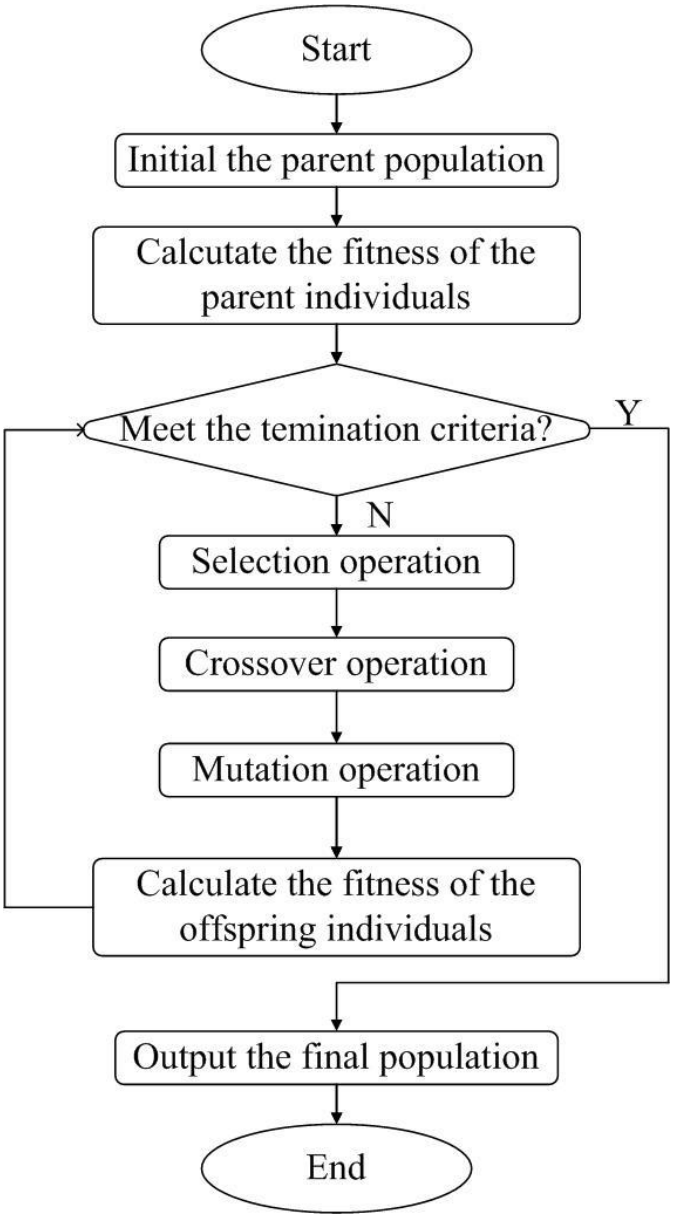

Fig 4.1 Flowchart of basic GA.

DE is a self-adaptive global algorithm based populations, which has a similar process with evolutionary algorithms, such as initialization of populations, evaluation of the fitness value of individuals, operating of genetic operators, and so on, shown in Fig.4.2.

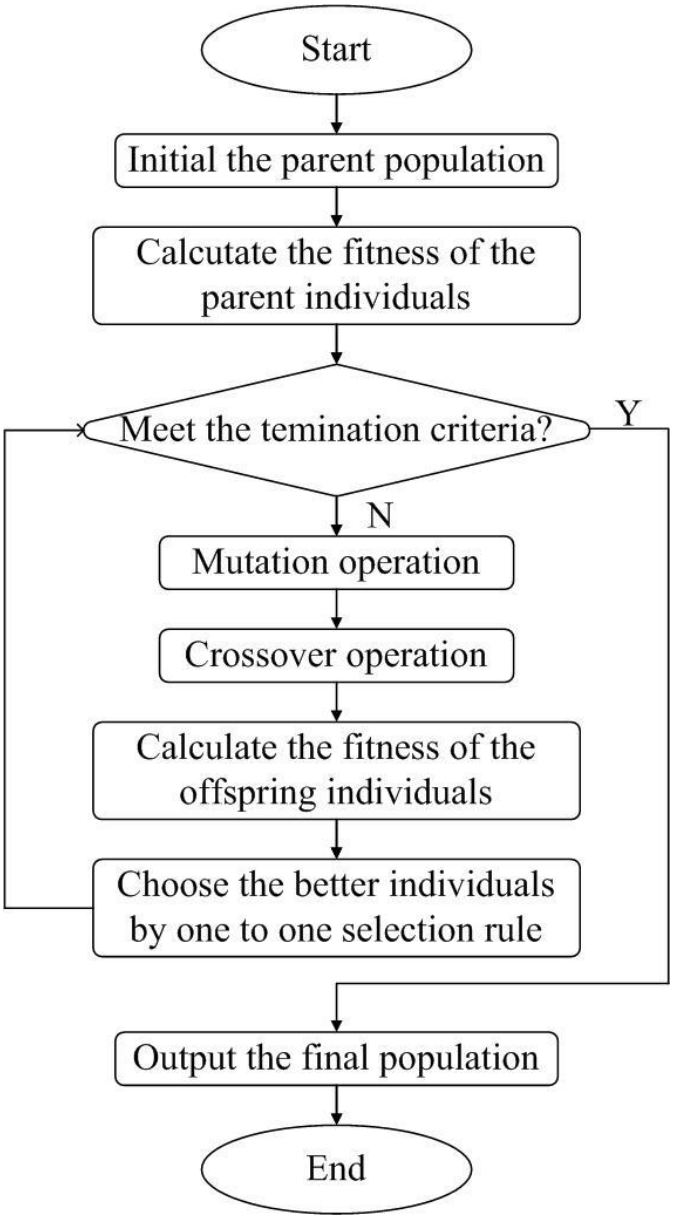

Fig 4.2 Flowchart of basic DE.

Fig. 4.3 demonstrates the pseudo code of DE based on DE/rand/1/bin strategy, where

N_P is the size of a population,

X_i is a decision vector,

D is the dimension of the decision vector,

j_{rand} is a random integer, $j_{rand} \in [1, D]$,

r_j is a real number generated randomly and uniformly from 0 to 1,

C_R is a crossover probability, $C_R \in [0, 1)$,

F is a scale factor, $F \in (0,1+)$,

U_i is a trial vector combining three different vectors.

1: For i = 1 to NP do
2: Randomly select $X_{r_1}, X_{r_2}, X_{r_3}$ (r1 ≠ r2 ≠ r3 ≠ i)
3: j_{rand} = rndint (1,D)
4: For j = 1 to D do
5: If $(rndreal_j < CR \text{ or } j = j_{rand})$ Then
6: $U_i(j) = X_{r_1}(j) + F\left(X_{r_2}(j) - X_{r_3}(j)\right)$
7: Else
8: $U_i(j) = X_i(j)$
9: End For
10: End For
11: End For

Fig 4.3 Pseudo code of DE/rand/1/bin.

The termination condition of this algorithm can be the maximum number of evolutionary generations or fitness evaluations, which is generally given by user in advance. As we can see from the figure, the basic differential evolution algorithm is simple and easy to implement.

There are two characters in the classical DE: the differential mutation operation and the selection operation which are so remarkable as to it is significant difference

from other evolutionary algorithms. The vectors of two different individuals in the same population execute differential and scaling operations, seen Line 6 of Fig 4.3, to get a temporary result. The temporary result adds the vectors of the third individual, which creates a new individual. The statement $j = j_{rand}$ ensures that the trial vector U_i doesn't duplicate X_i totally, and there is one dimension at least coming from the vector V_i, which maintains the diversity of the population. Fig. 4.4 plots the possible trial vectors in two-dimensional space, where X_i denotes the target vector, V_i denotes the mutation vector and U_i denotes the trial vector. The hollow circle in the figure represents the possible trial vectors.

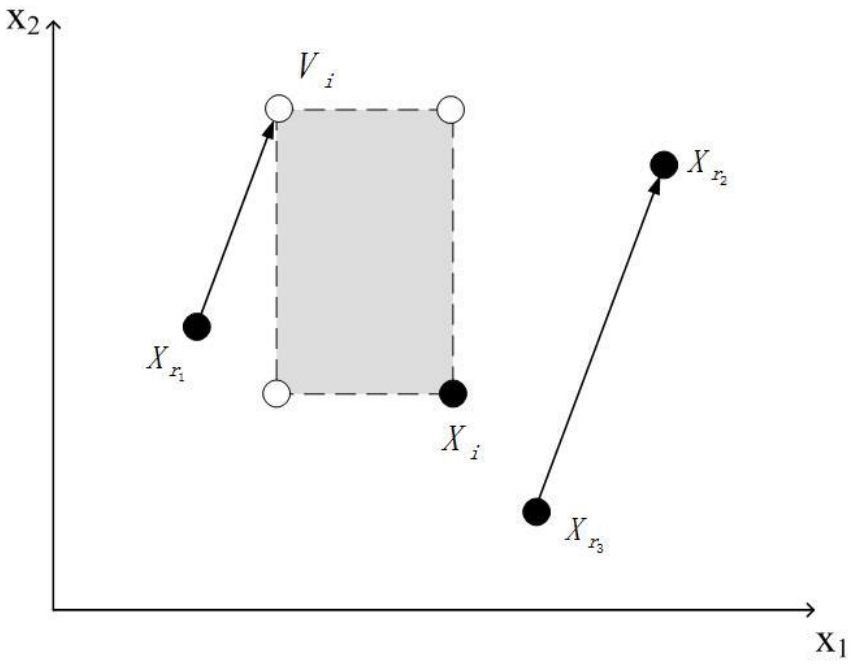

Fig. 4.4 2D Schematic of Differential mutation.

Finally, the algorithm uses one to one choice method, denoted as follows:

$$X_i = \begin{cases} U_i, & \text{if}(\, f(U_i) \le f(\, X_i)\,) \\ X_i, & \text{otherwise} \end{cases} \qquad (4.1)$$

Here $f(U_i)$ means an objective function of the trial vector and $f(X_i)$ means that of the target vector. If $f(U_i)$ has an equal or lower value than that of $f(X_i)$, it replaces the target vector in the next generation; otherwise, the target is retained in the population. This choice method can ensure that the elite individuals cannot be lost in the evolution.

The differential mutation operations can maintain the diversity of populations, and the one to one selection ensures the better individuals are not lost during evolutions. Therefore, these two operations are remained in our new algorithm for solving MVL network problems. Moreover the new one also uses DE/rand/1/bin strategy to produce new individuals.

Chapter 5
BOEDE for MVL Optimization

5.1 Introduction to BOEDE

We design a novel algorithm to optimize the MVL network, called Bi-objective Elite Differential Evolution (BOEDE). The algorithm is based on the classical Differential Evolutionary (DE) algorithm, and preserves the mutation and crossover operators of DE. The essential difference between our algorithm and the traditional algorithm is that it uses two evaluation functions, so it belongs to multiobjective optimization, while the previous algorithms all have a single objective function. The highlight of BOEDE lies in its archive population with fixed-size ranks which is used to store the optimal solutions with different ranks. At the same time, the method of updating the archive population is very unique. The spark in BOEDE is that it defines the concept of the elite solutions clearly and gives priority to save the elite solutions and not allowed to be replaced. The elitist solutions are the most precise solutions, that is, there is no error between the instruction data and the actual data.

The flowchart of BOEDE is shown in Fig. 5.1. The algorithm first generates an initialized parent population and calculates the fitness of its individuals, and then enters the cycle of evolutionary iteration. In each iteration, it firstly produces the offspring population by changing the parent population according to the principle of DE and calculates the fitness of its individuals. Secondly, it does some operations which include ranking the individuals according to their fitness, labelling the individuals by their differences. Thirdly, it updates the archive population using the offspring population, then takes the archive population after processing as the parent population of next generation.

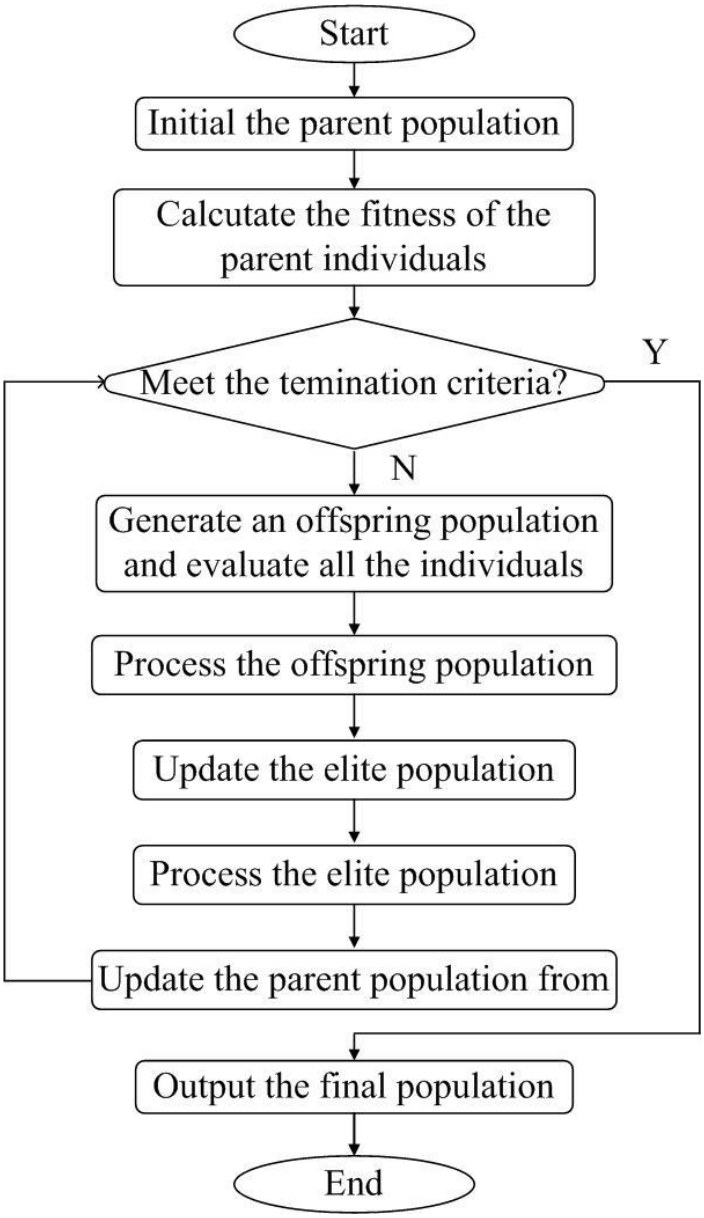

Fig. 5.1 Flowchart of BOEDE.

As the name, BOEDE, implies, the proposed algorithm contains three meanings:

- It bases on DE, follows the structure of basic DE, especially in producing new individuals;

- It puts forward the idea of optimizing the MVL networks with two objectives: incorrectness and optimality. Incorrectness reflects the accuracy of

optimization and optimality represents the complexity after optimization. In all the former algorithms there are only one evaluation: accuracy.

- It has a new storage structure of final solutions, which includes not only the most accurate solutions without any error, but also suboptimal ones with a little error but much lower complexity.

5.2 Coding and generating new individuals

The classical DE algorithm uses real number coding, which makes it is better for solving real number optimization problems. Assume that the variable of the problem to be solved has D dimensions, then the i^{th} individual, denoted as X_i, is expressed as follows:

$$X_i = \{x_i(j), x_i(j), \dots, x_i(j),\} \qquad (5.1)$$

where i=1,…, NP, j=1,…,D, $x_i(j) \in [L_j, U_j]$. NP indicates the size of the population. $L_j \; and \; U_j$ indicate the minimum and maximum values of variables, respectively.

BOEDE follows the coding method although it is for MVL networks which have discrete variables. In order to produce an integer $x_i(j) \in [B_L, B_U]$ ($B_L \; and \; B_U$ indicate the minimum and maximum values of variables, respectively), it firstly produces a real number between 0 and $(B_U - B_L + 1)$, then truncates the integer part to obtain a discrete value, which is according to

$$x_i = B_L + (int)\big(\, random\, real() * ((B_U - B_L + 1)\big) \qquad (5.2)$$

The variables is randomly generated when the population is initialized, which does have little effect on evolution of solutions because the evolution is based on the population and there are a lot of random unpredictable transformations.

5.3 Bi-objectives

We first put forward the idea of optimizing MVL networks with multiple objectives. In this specific algorithm, incorrectness and optimality are proposed as the two evaluation functions. Incorrectness indicates the error between the actual output and the instruction output, i.e., the output value in the given truth table corresponding to the actual output, and it is denoted as E in symbol and calculated as

$$E_c = \sum_p^P (O_p - T_p)^2 \tag{5.3}$$

where O_p represents the p[th] actual output value of a network, Tp represents the pth instruction output, and p represents the pth input vector $(x_1, x_2, \dots, x_n)_p$. The incorrectness function is the most important metric. The smaller the incorrectness value is, the better the performance of the network is. If the incorrectness value of a solution is 0, the solution is defined as an elite solution. Elite solutions are the best results with 0 error and should be collected as many as possible. Optimality indicates the number of valid MIN gates in the network, and it is denoted as G in symbol and is formally defined as

$$\mathrm{G} = \left|\{MIN_j : c_j \neq 0 \wedge a_{ij} \leq b_{ij}\}\right| \tag{5.4}$$

The unit component of an MIN gate is {cj; ai j; bi j}, and when cj , 0 and ai j $\leqslant$ bi j, the item of the MIN gate is valid, and the value of optimality increases by 1. The optimality function shows the manufacturing cost of a network. The smaller the optimality value, the lower the manufacturing cost. Although this equation only reflects the hardware cost, the implementation technology and other software costs must be taken into account.

Bi-objectives are applied in the process of selecting excellent individuals from the population. As mentioned in Section 2, the multiobjective evaluation is completely different from single-objective evaluation. In this study, the values of two objective

functions are compared, as shown in Table 5.1. There are two individuals labelled as Ind_i and Ind_j. Their incorrectness and optimality are denoted as E_i, E_j, G_i and G_j, respectively. The symbol k means the selection probability. When the incorrectness of Ind_i is less than the one of Ind_j and the optimality of Ind_i are less than the one of Ind_j at the same time, Ind_i dominates Ind_j, hence it will be selected into the next population. The cases can be seen from Lines 2 to 5 of Table 5.1. When IndiInd_i is indifferent to Ind_j, which is expressed from Lines 6 to 8 of Table 5.1, they have 50% chance of being selected into the offspring population.

Table 5.1 Comparison strategy of bi-objective functions.

E_i and E_j	G_i and G_j	Decision
<	< or ≤	Ind_i
=	<	Ind_i
=	>	Ind_j
>	> or ≥	Ind_j
<	>	$k = 0.5$
=	=	$k = 0.5$
>	<	$k = 0.5$

It is worth noticing that the bi-objective evaluation method runs both in parent population and offspring one through the whole of the algorithm. It is the biggest difference between BOEDE and the previous single-objective algorithms and it is one of the main contributions of the proposed method.

5.4 Archive with ranks

Archive population, similar to the traditional multiobjective optimization algorithm, is an effective structure to collect optimal solutions during the evolution, and it is also the place where the final output of a network corresponds to the optimal decision vectors.

An individual from offspring population and one from archive population are compared with their objective functions, and the better one is saved into the latter. This strategy is mainly suitable for MOPs with continuous decision variables, because the number of the decision variables is infinite, and a tiny change of a decision variable may change the value of an objective function. Specifically speaking, the different genotypes of individuals can lead to different phenotypes. If the phenotypes of two individuals are almost the same, their genotypes must be the same, and one of them can thus be abandoned. If they are indifferent with each other, one of them can be abandoned at once because there are infinite solutions. But this strategy is not suitable for an MVL network learning problem, because 1) the number of its solutions is rather limited, and 2) a solution with a kind of phenotype may have several or more corresponding solutions with different genotypes.

1) Structure of Archive Population:

The structure is the same as the one of parent and offspring populations in most of multiobjective algorithms. The comparison strategy of multiobjective functions like Table II has been adopted to save optimal solutions. But they are not suitable for the archive in an MVL network problem.

In most of MOPs, both decision variables and objective functions are continuous and a tiny change of a decision variables may change the values of objective functions. That is to say, they can have numerous solutions with different phenotypes and different genotypes. While in MVL network problems, the situation is sharply different. Specifically for a 2-variable 4-valued MVL function, each decision variable can only take 4 integer values: 0,1,2 and 3, but a decision vector has 80 decision variables expressed as

$$\mathrm{V} = \{a_{1,1}, b_{1,1}, a_{2,1}, b_{2,1}, c_1, a_{1,2}, b_{1,2}, a_{2,2}, b_{2,2}, c_2, \dots\dots, a_{1,j}, b_{1,j}, a_{2,j}, b_{2,j}, c_j, \dots\dots, a_{1,m}, b_{1,m}, a_{2,m}, b_{2,m}, c_m\} \qquad (\mathrm{m} = 4^2 = 16) \qquad (5.5)$$

Thus we have 4^{80} combinations of parameters. Consequently the solutions of this

problem have an extremely number of individual genotypes although the variables are rather limited. Then the two fitness functions are analyzed. G is an integer number to indicate the number of valid MIN gates. There must be many individuals with the same G in the population because its range from 1 to 16 is very limited. E is a simple function whose result is a positive integer with a range. It has a certain probability of repetition. Taken together, many solutions of MVL network problems are probably of different genotypes but of the same phenotype. Furthermore, because the parameters are randomly and uniformly initialized in the early stage of evolution, the individuals usually have a larger E and a smaller G. If the above selection strategy is applied, according to the definition of Pareto optimal solutions, the individuals of the archive population inevitably would bias toward small G, which is bad for the convergence of E.

Therefore, a new architecture of archive population is required because of the particularity of MVL network problems. The archive population is divided into 16 ranks, each of which has a fixed size. The structure is shown in Fig. 5.2. The two objective functions of an MVL network are both combined into the structure. The archive population ranks by G since there are at most 16 MIN gates in the network. Each rank can store excellent individuals including the elite solutions with 0 error and the Pareto solutions. The maximum capacity of storage is expressed as R. The structure is simple but novel. It is very effective for the collection of the excellent solutions of MVL networks with discrete variables.

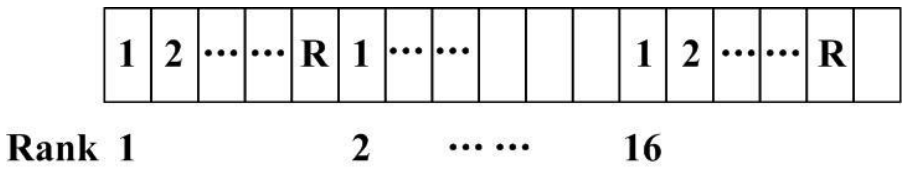

Fig. 5.2. Structure of the archive population.

2) Method of Updating Archive Population:

A new method of updating the archive population which corresponds to the new structure is now proposed. This is a partly update method that fully bases on the bi-objective functions. First offspring population and archive population are ranked by the optimality and then sorted by the incorrectness, and the elite individuals and the ones with the same variables are labelled, respectively. Each step of the following update operations is carried out in the range of each rank, which ensures the diversity of optimality during the period of optimization. Second collecting excellent solutions as many as possible ensures the diversity of incorrectness.

In the first generation, the archive is empty without any individual. If the number of offspring individuals is less than or equal to R, all the individuals are copied to the archive population; otherwise the R ones are copied and the remaining are cut off. In every subsequent generation, if the sum of offspring individuals and archive ones is less than or equal to R, all are copied to the archive one of course; otherwise the archive population must be partly updated according to Fig. 5.3.

The following parameters are used: r is the current rank; R is the fixed size; $L \in \{1,2,\dots,R/2\}$ is a randomly generated integer that indicates one position of the updated individual of archive population; There are N individuals with rank r in archive population and there are M individuals with rank r in offspring population, and the updating position j is the first one. The rule of partly update is described as follows:

- Case 1: Theoretically, if $M \leq R - N\ (N \leq R)$, all individuals of offspring population are copied to archive population starting from the location N + 1; Otherwise,

- Case 2: the update starts from position L and ends when it gets to the last position of archive population or offspring population with rank r. Actually, elite solutions must be paid much attention not to be overwritten by the updating individuals because the number of elite solutions is very small, and individuals with the same genotypes cannot be used repeatedly. Thus it is necessary that elite solutions and the individuals with the same genotypes must be labelled in pretreatment.

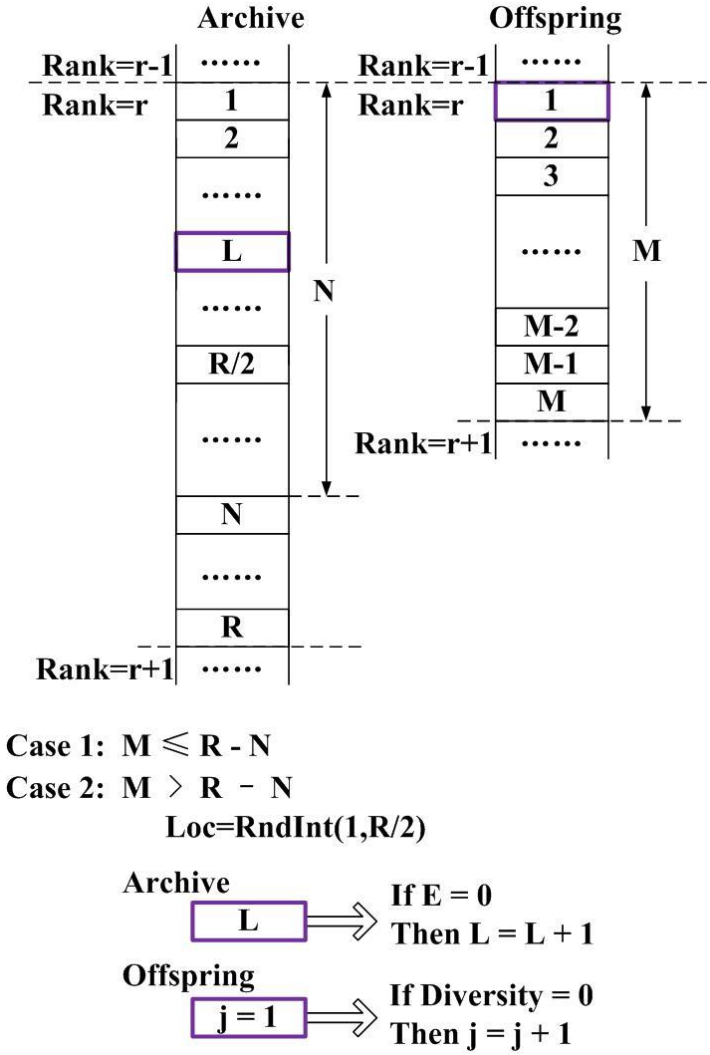

Fig. 5.3. Illustration of partly update.

3) Characteristics of updating:

In order to solve an MVL network problem in which there are two objective functions and individuals with many genotypes but singular phenotypes, archive population with a novel structure and updating method is innovatively proposed to maintain the diversity of archive population, and to improve the accuracy during evolution. Its characteristics are summed up as follows:

a) The updated location of each rank of archive population is set by a random function L, which ensures that the best solutions in archive population are not replaced and the flag in pretreatment ensures that the elite ones are not replaced, either;

b) The updating location of each rank of offspring population begins from the first one, which ensures the best solutions of offspring population are able to enter archive population;

c) The number of updated individuals is limited through parameters L, R, M and N, which ensures the updated rate of archive population. In general, individuals cannot be immutable unless all the individuals are elite ones;

d) The update at a probability of 50% can keep the individuals with poor fitness in archive population, slow down the evolution speed and reduce the possibility of premature convergence;

e) The sort operation works only by the incorrectness of individuals because the ranks of archive population are sorted themselves, so it does not bring too much computational burden;

f) We do not use the one-to-one comparison (which is a common operator in DE) in the period of update, but instead carry out the update based on a random location generated with 50% probability in archive population. Compared to the one-to-one comparison operator, the proposed update strategy is straightforward and also effective to be shown in the experiments.

5.5 Other operations in detail

1) Production of parent population:

In evolutionary algorithms, parent population is the foundation of other populations. Offspring population based on it is generated by some kinds of rules, and

then usually acts as the next parent population. While in our algorithm the parent population after the first generation is not the offspring but the archive one after being updated because the average fitness of its individuals is better than the other's. Making the archive population as the parent can improve the efficiency of evolution.

2) Handling of illegal variables:

Initially, all the variables of V in (5.5) are generated randomly and uniformly distributed. Most of them are legal. But some of them may overflow the decision space and become illegal after differential mutation operations. These illegal variables have to be dealt with in order to return to the feasible space. The classical approach in DE resorts to symmetric mapping as shown in Fig. 5.4(a), while our algorithm employs an arbitrary maximum values, shown in Fig. 5.4(b). B_L and B_U represent the lower and upper bounds respectively. In Fig. 5.4(a), variables x_1 is outside of the upper bound. The symmetric mapping takes point B_U as a symmetric center, gets the point x_1' that is on the line from point B_U to point B_L and has the same distance from itself to B_U as the one from x_1 to B_U. Point x_1' is a legal variable corresponding to x_1.

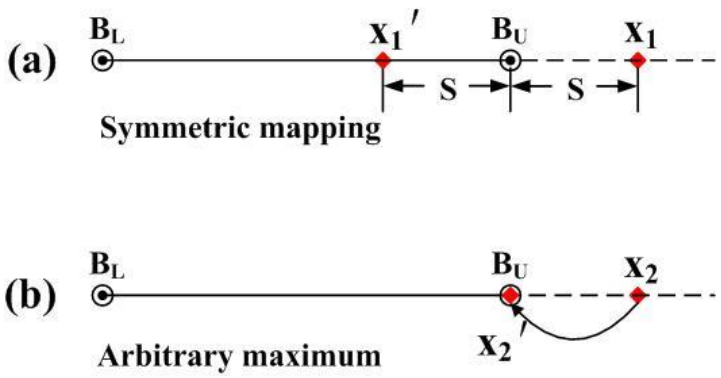

Fig. 5.4. Handling of illegal variables.

Although the symmetric mapping is effective in DE, it usually takes effect in continuous optimization rather than discrete one [65]. Especially for MVL networks,

the symmetric mapping makes the algorithm generate many illegal solutions. In Fig. 5.4(b), once a variable is illegal, it is directly assigned with the maximum value. According to the validity of MIN gates, in the five-member group $\{a_{1,j}, b_{1,j}, a_{2,j}, b_{2,j}, c_j\}$, the probability of three variables $b_{1,j}, b_{2,j}\ and\ c_j$ that can acquire the maximum value are greatly increased, thus increasing the number of valid MIN gates, and improving the performance of the evolution.

Chapter 6
Simulation

6.1 Parameters

Our algorithm BOEDE is applied to solve the MVL network problems with 2 variables and 4 values. All the simulations are implemented in Microsoft Visual Studio 2017 on a personal computer with Intel i5-4460 3.20G CPU and 4GB memory. The parameters used in the simulations are listed in Table 6.1.

Table 6.1 Description of the experimental parameters.

Name	Value	Description
popsize	300	the size of populations
m_F	0.5	the scale factor
m_CR	0.3	the crossing probability
R	30	the size of each rank in archive population
ngen	5000	the number of iteration times
Runno	10	the number of running times

6.2 Searching for elite solutions

The classical truth table in Table I are used as the first test instance of our algorithm and a number of elite solutions are found.

The simulations iterate from 1000 to 20000, and each kind of iteration is run 10 times. The average number of elite solutions with 0 error is shown in Fig. 6.1. As can be seen from the figure: when the number of iterations is 1000, the average number of elite solutions is 0.1; when the number of iterations is 20000, the number is 245; when it is less than 5000, the evolution of the MVL network is not sufficient; when it reaches 5000, the number of the solutions increases greatly; when it reaches about 12000, the diversity of solutions tends to be stable.

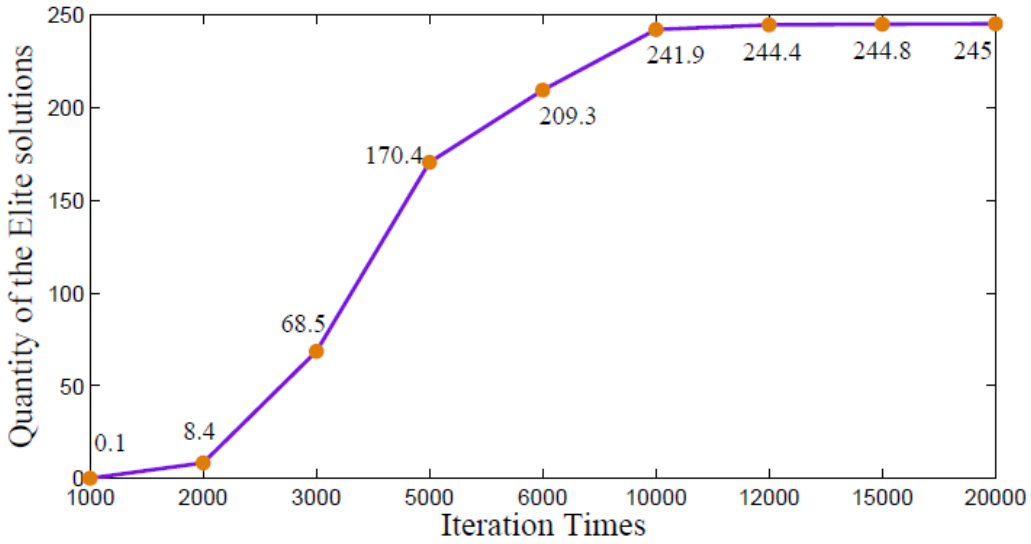

Fig. 6.1. Trend of the average quantity of the elite solutions on the instance Table 2.1 when the parameter ngen varies from 1000 to 20000 and Runno = 10.

A case of 5000 iterations is analyzed carefully. After 10 run times, the average number of elite solutions with optimality 5 (i.e., 5 valid MIN gates after optimization)

is founded to be 4, and the number with optimality 6 is founded to be 30, and so on, as shown in Fig. 6.2. The number 30 is the maximum storage capacity of each rank in the archive population. In other words, the archive population of ranks 6, 7 and 8 are full of the elite solutions with error of zero. The size of the archive population is 480 (16R), and the number of all elite solutions is 170.4 on average in each independent run. Hence the ratio of elitist solutions to total solutions is 37.87%.

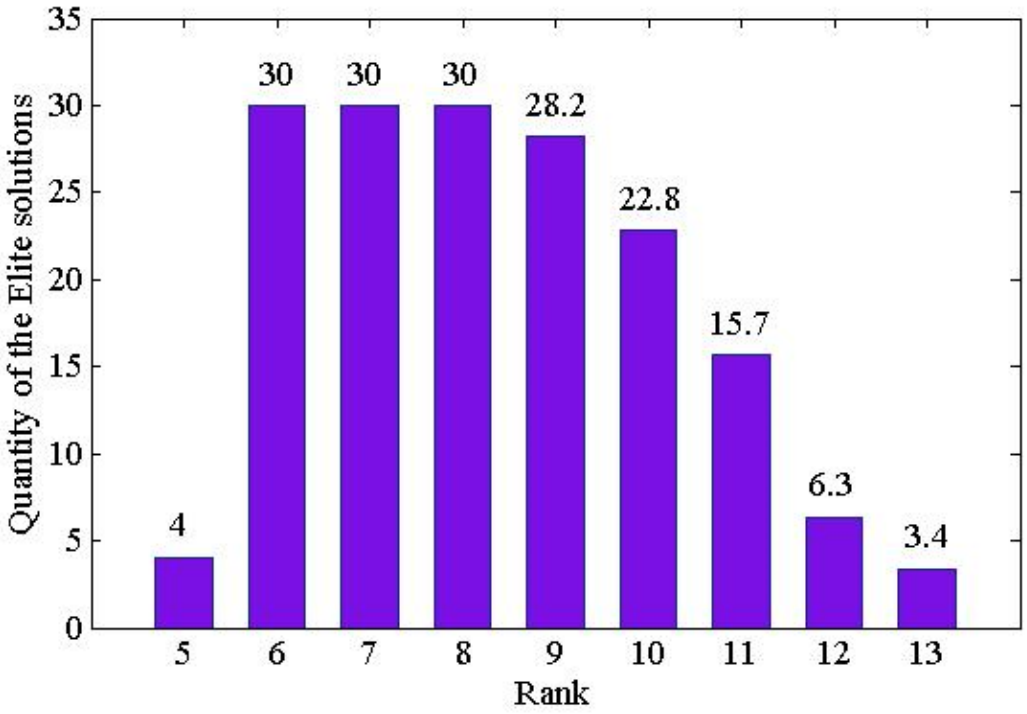

Fig. 6.2. Average number of the elite solutions (i.e., E = 0) with different optimality on the instance shown in Table I using ngen = 5000 and Runno = 10.

To verify whether all the elite solutions with the same optimality are really different, we further examine the genotype of every solution. Four elite solutions with optimality 5 obtained by BOEDE are listed as follows:

$$F(x_1, x_2) = 1 \cdot x_1(0,2)x_2(0,0) + 1 \cdot x_1(0,3)x_2(3,3) + 1 \cdot x_1(1,3)x_2(1,1) + 2 \cdot x_1(3,3)x_2(3,3) + 3 \cdot x_1(2,2)x_2(0,2) \quad (6.1)$$

$$F(x_1, x_2) = 1 \cdot x_1(0,1)x_2(0,0) + 1 \cdot x_1(0,3)x_2(3,3) + 1 \cdot x_1(1,3)x_2(1,1) + 2 \cdot x_1(3,3)x_2(3,3) + 3 \cdot x_1(2,2)x_2(0,2) \quad (6.2)$$

$$F(x_1, x_2) = 1 \cdot x_1(0,2)x_2(0,0) + 1 \cdot x_1(0,2)x_2(3,3) + 1 \cdot x_1(1,3)x_2(1,1) + 2 \cdot x_1(3,3)x_2(3,3) + 3 \cdot x_1(2,2)x_2(0,2) \quad (6.3)$$

$$F(x_1, x_2) = 1 \cdot x_1(0,1)x_2(0,0) + 1 \cdot x_1(0,2)x_2(3,3) + 1 \cdot x_1(1,3)x_2(1,1) + 2 \cdot x_1(3,3)x_2(3,3) + 3 \cdot x_1(2,2)x_2(0,2) \quad (6.4)$$

The four sets of solutions obviously have the same phenotype (E = 0 and G = 5), yet different genotypes. This means that there are different input vectors in the MVL networks and there are different line connections and different thresholds in real logic circuits. Eq. (6.1) is exactly the same as (2.7), which is the optimization result by the mathematical method mentioned. It can be also obtained by the single objective optimization methods, i.e., SDLS [54] and IMVL [57].

As shown in Fig. 6.2, there are plenty of elite solutions (i.e., E = 0) with optimality 6, 7 and 8 respectively. To enumerate and compare these solutions, the following equation is used to judge whether two elite solutions x and y ($\in$ V) with the same optimality are of the same genotype solutions or not.

$$\text{Div} = \sum_{i=1}^{D}(x_i - y_i)^2 \qquad (6.5)$$

From (6.5), it is clear that Div = 0 indicates two solutions are the same in the genotypic representation. By using (6.5), it is verified that the elite solutions summarized in Fig. 9 with the same or different optimality differ from each other in the genotypic representation, which suggests that BOEDE is more capable of maintaining the diversity of obtained solutions.

In addition to the instance shown in Table 2.1, other 29 truth table instances are tested and the results based on all these 30 instances are summarized in Table 6.2. In Table 6.2, e.g., the instance Test01_11_18 indicates that the number of non-zero items of the truth table is 11, and the sum of the values of all non-zero items is 18. In addition, the symbol * indicates that the elite solutions are not found when the iterations reaches 5000, but a small number of elite ones can be found when it reaches 10000. From Table IV, it is clear that BOEDE can find elite solutions with no error in accuracy for 18 out of 30 instances when ngen=5000. Based on the above results, we conclude that BOEDE can obtain a good number of different elite solutions that can definitely provide desired decision supports for practical applications of MVL networks.

Table 6.2 Average number of elite solutions for 10 runnings on 30 Truth Tables (ngen=5000, Runno=10).

Instance	Number	Instance	Number
Test01_11_18	170.4	Test16_14_26	0
Test02_15_25	127.5	Test17_12_20	0
Test03_12_25	100.1	Test18_09_24	202.4
Test04_10_21	87.9	Test19_16_27	264.2
Test05_12_20	26.1	Test20_13_24	0
Test06_10_17	51.6	Test21_09_24	11.6
Test07_10_19	77	Test22_12_24	0
*Test08_09_16	0	Test23_11_25	0
Test09_08_15	211.5	*Test24_09_16	0
Test10_07_14	176.8	Test25_10_22	82.8
Test11_06_11	11.9	Test26_11_20	0
Test12_11_23	101.5	Test27_10_13	17.9
*Test13_13_20	0	*Test28_16_25	0
Test14_10_19	4.9	Test29_13_21	0
Test15_13_23	1.1	Test30_14_27	0

6.3 Searching for Pareto optimal solutions

The elite solutions have perfect requirements on accuracy, but sometimes there is no need to reach such perfect accuracy. We can reduce accuracy within some allowable range to achieve additional improvements of other aspects. For example, an ideal simplified elite solution with optimality 5 (i.e., E = 0 and G = 5) can be replaced by the two solutions: one, denoted as S_A, is with incorrectness 2 and optimality 4 (i.e., E = 2 and G = 4), and the other, denoted as S_B, is with incorrectness 1 and optimality 6 (i.e., E = 1 and G = 6). They are indifferent with each other and they are the Pareto solutions with the same rank. Although S_A and S_B have a small error compared to the ideal solution, S_A has fewer valid MIN gates which can reduce the manufacturing cost, while S_B has relatively more valid MIN gates which can save the technical cost of the matching control circuits [8] [9].

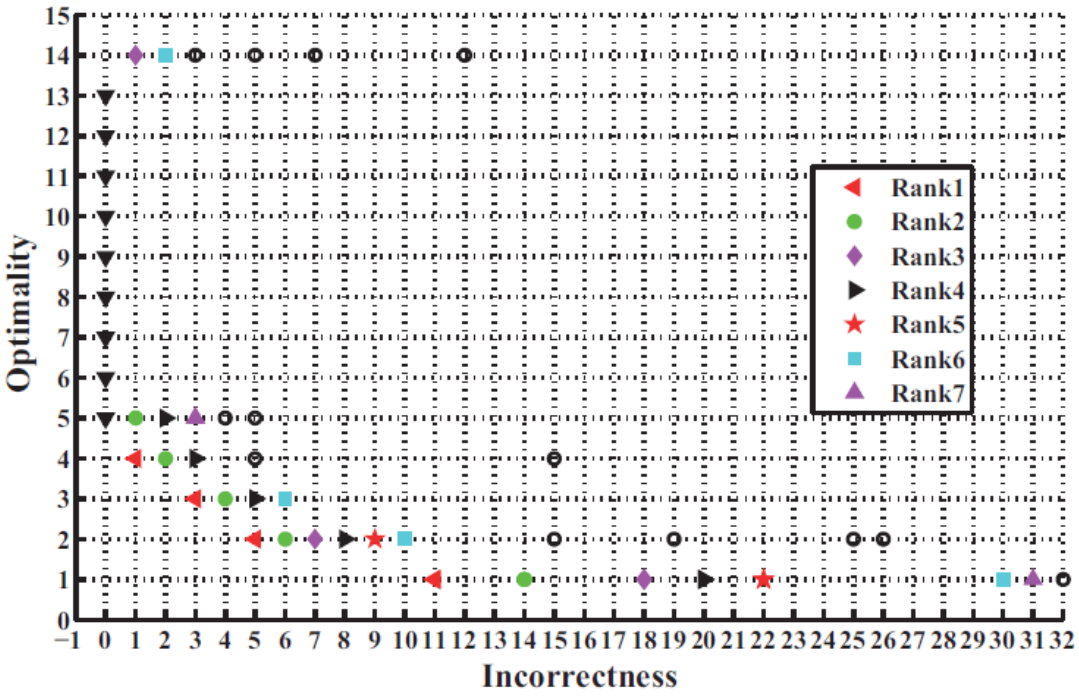

Fig. 6.3. Illustration of all the elite solutions with different optimality and all the Pareto optimal solution with different ranks, according to the sixth running results on Table I with ngen=5000.

Fig. 6.3 shows a typical running result on Table 2.1 with 5000 iterations among 10 runs. The horizontal axis denotes the incorrectness function (i.e., E in (11)) and the vertical axis denotes the optimality (i.e., G in (12)) after optimization. This figure shows the phenotypes of all the final solutions in the archive population, according to E and G, regardless of the genotype. For example, those dots symbolled ▼ indicate the elite solutions (i.e., E = 0) with different optimality. There are 9 solutions and their optimality varies from 5 to 13. Except for ▼, in this figure, marks of different shapes are used to label the Pareto optimal solutions with different ranks, and all the solutions with the same icons have the same rank. The rank of Pareto optimal solutions indicates a sorted list described in Section II-B3. In Fig. 6.3, together with the elitist solutions, Pareto solutions with the first 7 ranks are illustrated by different marks, while those with lower ranks are all drawn using hollow circles.

In order to show the ranks more clearly, the elite and Pareto optimal solutions

with low quality are removed from the statistical data. Fig. 6.4 focuses on Pareto optimal solutions with the first 7 ranks. The upper limit of incorrectness is set to 22. The discrete points presented by the two objective functions are obtained Pareto solutions, while the connections between the two points are virtual to express the ranks of the Pareto optimal solutions intuitively. In Fig. 11, there are 21 solutions totally, distributed uniformly in the Pareto front, thus suggesting that BOEDE is able to generate a sufficient number of uniformly distributed non-dominated solutions for MVL networks.

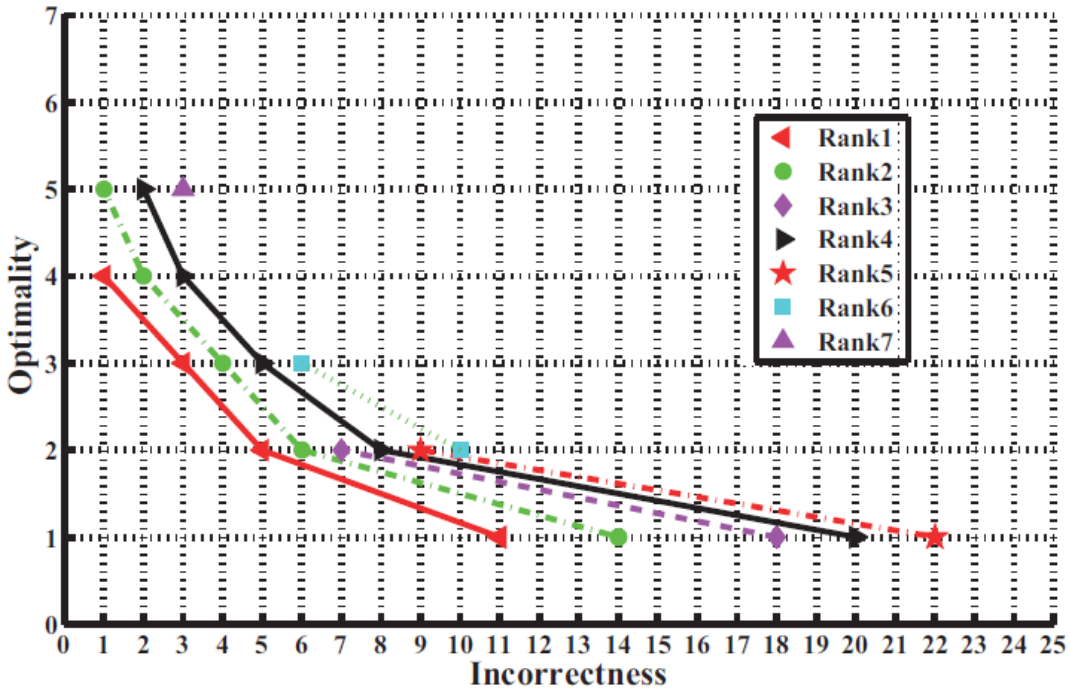

Fig. 6.4. Pareto optimal solutions with different ranks.

6.4 Comparison with other algorithms

In this paper, BOEDE is compared with two typical algorithms from two aspects: the quality of solutions and the average number of MIN gates after MVL networks are optimized.

1) The quality of solutions

The improved local search leaning method [57], IMVL in brief, is one of the most competitive algorithms for learning MVL networks. It utilizes a single-objective function to optimize MVL networks. It combines the local search and chaos to evaluate the correctness and optimality in a form of a weighted sum. IMVL is proved to be the best in comparison with local search (LS) [52] and stochastic dynamic local search (SDLS) [54]. Therefore, BOEDE is compared with IMVL first in this paper.

IMVL is a single-objective algorithm and it has an explicit function of collecting the best solutions. It collects solutions in an ascending order of accuracy. While BOEDE has two fitness functions and is completely different from single-objective algorithms. It not only distinguishes elite solutions and Pareto optimal ones, but also takes into account the distribution of these solutions. Thus we concentrate on the comparison of the optimal solutions collected by the two algorithms.

Take Table 2.1 as the test instance first. IMVL is based on the individual evolution and the parameters are set to values suggested in [57]. The algorithm generates 100 initial individuals to take part in the evolution and each individual has 1000 iterations. The process is repeated for 10 times. 1000 optimal solutions are thus collected. On the other hand, BOEDE is an evolutionary algorithm based on population, and the population size is set to 300, the iteration number is set to 5000, and 450 optimal solutions are thus collected. Because of the need to analyze the genotypes of the solutions, we choose the first running results that are similar to the average one as the compared objective. In addition, it is necessary to note that the iteration count 1000 in IMVL is sufficient as the evolution has been convergent due to its inherent local search property, while the iteration count 5000 in BOEDE is a starting line of reaching the best solutions.

Fig. 6.5 shows the comparison results of the elite solutions obtained by IMVL and BOEDE. IMVL can find only 2 elite solutions, and their phenotypes are the same

(E = 5). BOEDE can find 170 solutions with different optimality and there are 7 phenotypes E ∈ {5, 6, 7, 8; 9, 10, 11}. More importantly, those solutions with the same G obtained by BOEDE can possess several different genotypes. For example, the solutions with E = 0 and G = 5 have 4 genotypes, the ones with E = 0 and G = 6 have 30 genotypes, and the ones with E = 0 and G = 11 have 17 genotypes. From Fig. 6.5, it is clear that BOEDE can obtain more fruitful solutions than IMVL, thus providing more decision supports for practical applications of MVL.

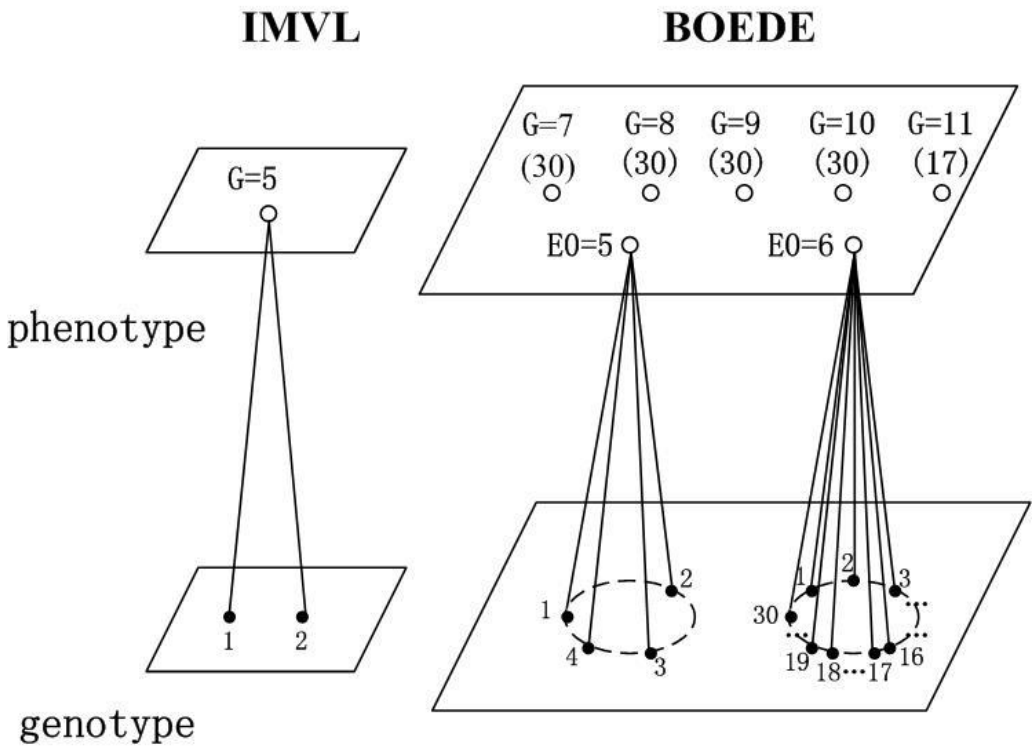

Fig. 6.5. Comparison of the elite solutions obtained by IMVL and BOEDE.

Moreover, Table 6.3 illustrates the comparison of all the valid solutions obtained by IMVL and BOEDE. It is sorted by E in an ascending order. The valid solutions means that the solutions have different genotypes. In IMVL there are 8 valid solutions with E = 1 and their corresponding G are 4 and 5. In BOEDE there are 16 valid solutions with E = 1 and their corresponding G are 4, 5, 12, 13 and 14. It is clear that

BOEDE can obtain much more valid and diverse solutions than IMVL for instance Test01_11 _18. This table also clearly shows that the solutions of IMVL with different E values have a few types of G while the solutions of BOEDE, not only the elite but also the Pareto optimal ones, have a wide range of G, and their quantity is enough, their distribution is balanced, and they can meet the need for various applications. In addition, it is worth noting that only 65 out of 1000 solutions are valid in IMVL, while 296 out of 450 solutions are valid in BOEDE. There are a large number of repeated solutions in IMVL and additional analysis on the same genotype must be performed in order to delete all the repeated solutions, which will consume additional computational time.

Table 6.3 Comparison of all the solutions with different Ec.

	Number of Valid Solutions		**Range of *G***	
E	**IMVL**	**BOEDE**	**IMVL**	**BOEDE**
0	2	170	5	5,6,7,8,9,10,11
1	8	16	4,5	4,5,12,13,14
2	10	25	4,5	4,5,11,12,13,14
3	15	20	3,4	3,4,5,12,13,14
4	9	9	3,4	3,4,12,13
5	9	8	2,3	2,3,4,13
6	4	6	2,3	2,12,13
Others	8	41	1,2,3	1,2,4,12,13,14
Total	65	296		

Finally, for all tested 30 MVL instances, the elite solutions obtained by IMVL and BOEDE are compared as shown in Fig. 6.6 and Table 6.4. In IMVL the elite solutions can be found for 15 instances and the success rate is 50%, while BOEDE can find elite solutions for 18 instances and its success rate is 60%. Averagely, BOEDE can find 57.57 elite solutions which is much much more than those (1.47) found by IMVL.

Fig. 6.6 shows the number of elite solutions of 30 instances in a box-and-whisker diagram, which clearly illustrates that BOEDE can find more elite solutions for MVL networks than IMVL.

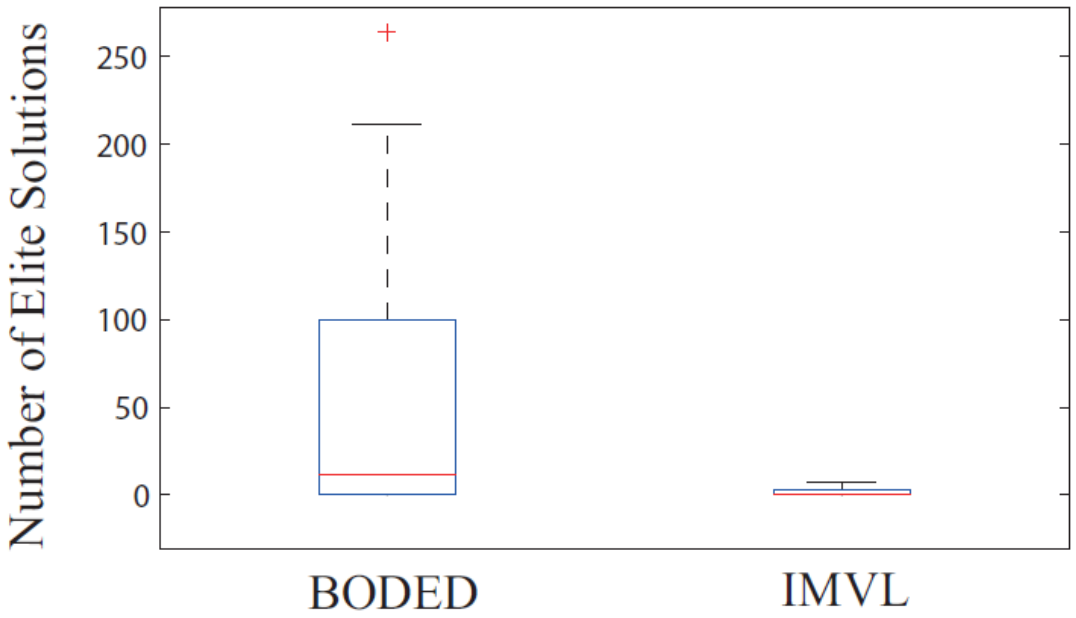

Fig. 6.6. Comparison of the solutions with E = 0 obtained by IMVL and BOEDE.

Table 6.4 Comparison of the number of elite solutions found by IMVL and BOEDE for all tested MVL instances.

	Quantity of the elite solutions	
	IMVL	**BOEDE**
Total elite solutions	44	1727.2
Average elite solutions	1.47	57.57
Success rate	50%	60%

2) The average number of MIN gates

DC technique is used to obtain optimal or near-optimal covers of MVL networks.

ACO-DC is a hybrid algorithm which utilizes both the greedy search feature in large solution space of ACO and the exact search feature in local area of DC. It can optimize MVL networks effectively, aiming at the accuracy metric in essential, and at the same time concentrating on the optimality of MVL networks, i.e., the number of valid MIN gates. In [41], it is tested against 50000 randomly generated 2-variable 4-valued truth tables, and calculates the average number of MIN gates after these networks are optimized, and then compares the average number to other algorithms. Those simulations show that ACO-DC is better than the fuzzy-based direct cover algorithm [39] and the multiple connected pseudo minterms algorithm [40]. Therefore, BOEDE is compared with ACODC only in this study. It is noted again that BOEDE is a bi-objective algorithm, which focuses on the accuracy and optimality metrics during the process of optimizing MVL networks. It collects many solutions, including elite and Pareto optimal ones. Our algorithm also generates 50000 2-variable 4-valued truth tables randomly and optimizes them, then calculates the average optimality G to compare with that obtained by ACO-DC. The result is shown in Table 6.5, which indicates that the average number of MIN gates obtained by BOEDE is less than the number obtained by ACO-DC.

Table 6.5 Comparison of the average number of MIN gates.

Algorithm	Average number of MIN gates
ACO-DC	7.02906
BOEDE	5.784

Chapter 7
Conclusion

In this paper we propose a novel bi-objective algorithm BOEDE to optimize MVL networks. The two objective functions, incorrectness and optimality, are considered as the optimization targets independently and equally. To effectively maintain the diversity of non-dominated solutions, we design a unique structure and a novel updating method for archive population that is used to store elite solutions with different optimality and Pareto optimal ones with different ranks. Simulations based on 30 truth tables with 2 variables and 4 values show that BOEDE outperforms the traditional algorithms in terms of the number of elite solutions found and the diversity of solutions. Simulations based on 50000 truth tables with 2 variables and 4 values show that BOEDE also superior to the traditional algorithms in terms of optimality. These merits of BOEDE can provide better decision supports for real MVL applications than any existing methods.

In the future we plan to study the influence of user-defined parameters of

BOEDE and try to incorporate the effective direct cover method as a local search technology into BOEDE to further improve its performance. We also plan to use other computational intelligence methods, such as artificial immune algorithms [89-102], gravitational search algorithms [103-107], ant colony optimization [108], imperialist competition algorithms [109-111], and brain storm optimization methods [112-114], to learn MVL networks. Also, the multivalued dendritic neuron models [115-123] will be designed to verify the effectiveness of a single neuron model.

Acknowledgements

I sincerely thank my Professor Zheng Tang and my Associate Professor Shangce Gao, whose careful and able research is of great help. They guide me to know about artificial intelligence step by step and establish my subject of research eventually. They give me plenty of help, advice, experience, encouragement and never-ending support during my studying for degree. I also thank my paper collaborators deeply, with whom I discuss our research, explore some methods to solve problems. I couldn’t finish my dissertation without their help and support.

I am grateful for all the members of Tang Laboratory, for all the teachers and employees in University of Toyama, for all the friends I know here. They help me know about our university, be familiar with it and love it gradually. Life is beautiful because they are here.

I sincerely thank my family, my parents, my husband and my lovely son. Thank them for their support, encouragement, patience and love which enable me to complete my research.

Thank all the people and wish you healthy and happy.

References

[1] T. Sasao, “Multiple-valued decomposition of generalized boolean functions and the complexity of programmable logic arrays,” IEEE Transactions on Computers, vol. 30, no. 9, pp. 635–643, 1981.

[2] Z. G. Vranesic and Z. Zilic, “A multiple-valued reed-muller transform for incompletely specified functions,” IEEE Transactions on Computers, vol. 44, no. 8, pp. 1012–1020, 1995.

[3] J. T. Butler, “Multiple-valued logic examining its use in ultra-high speed computation,” IEEE Potentials, vol. 14, no. 2, pp. 11–14, 1995.

[4] A. Raychowdhury and K. Roy, “Carbon-nanotube-based voltage-mode multiple-valued logic design,” IEEE Transactions on Nanotechnology, vol. 4, no. 2, pp. 168–179, 2005.

[5] M. E. Romero, E. M. Martins, and R. R. Santos, “Multiple valued logic algebra for the synthesis of digital circuits,” in Proc. of the 39th International Symposium on Multiple-Valued Logic, 2009. ISMVL’09. IEEE, 2009, pp. 262–267.

[6] J. Liang, L. Chen, J. Han, and F. Lombardi, “Design and evaluation of multiple valued logic gates using pseudo n-type carbon nanotube fets,” IEEE Transactions on Nanotechnology, vol. 13, no. 4, pp. 695–708, 2014.

[7] G. Epstein, G. Frieder, and D. C. Rine, “The development of multiplevalued logic as related to computer science,” Computer, vol. 7, no. 9, pp. 20–32, 1974.

[8] S. L. Hurst, “Multiple-valued logic its status and its future,” IEEE Transactions on Computers, vol. 33, no. 12, pp. 1160–1179, 1984.

[9] H. Inokawa, A. Fujiwara, and Y. Takahashi, “A multiple-valued logic and memory with combined single-electron and metal-oxide-semiconductor transistors,” IEEE Transactions on Electron Devices, vol. 50, no. 2, pp. 462–470, 2003.

[10] Y. Mo, L. Xing, and S. V. Amari, “A multiple-valued decision diagram based method for efficient reliability analysis of non-repairable phasedmission systems,” IEEE Transactions on Reliability, vol. 63, no. 1, pp. 320–330, 2014.

[11] Y. Iijima, Y. Takada, and Y. Yuminaka, “High-speed interconnection for vlsi systems using multiple-valued signaling with tomlinson-harashima precoding,” IEICE Transactions on Information and Systems, vol. 97, no. 9, pp. 2296–2303, 2014.

[12] K. C. Smith, “The prospects for multivalued logic: A technology and applications view,” IEEE Transaction Computers, vol. 30, no. 9, pp. 619–634, 1981.

[13] A. K. Jain, R. J. Bolton, and M. H. Abd-El-Barr, “Cmos multiple-valued logic design. i. circuit implementation,” IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 40, no. 8, pp. 503–514, 1993.

[14] S. Mahapatra and A. M. Ionescu, “Realization of multiple valued logic and memory by hybrid setmos architecture,” IEEE Transactions on Nanotechnology, vol. 4, no. 6, pp. 705–714, 2005.

[15] S. Lin, Y.-B. Kim, and F. Lombardi, “Cntfet-based design of ternary logic gates and arithmetic circuits,” IEEE Transactions on Nanotechnology, vol. 10, no. 2, pp. 217–225, 2011.

[16] M. H. Abd-El-Barr, Z. G. Vranesic, and S. G. Zaky, “Algorithmic synthesis of MVL functions for CCD implementation,” IEEE Transactions on Computers, vol. 40, no. 8, pp. 977–986, 1991.

[17] A. Ngom, I. Stojmenovic, and R. Tosic, “The computing capacity of three-input multiple-valued one-threshold perceptrons,” in Proc. of the 30th International Symposium on Multiple-Valued Logic. IEEE, 2000, pp. 33–38.

[18] I. Aizenberg and N. Aizenberg, “Pattern recognition using neural network based on multivalued neurons,” Engineering Applications of Bio- Inspired Artificial Neural Networks, pp. 383–392, 1999.

[19] I. Aizenberg, E. Myasnikova, M. Samsonova, and J. Reinitz, “Application of the

neural networks based on multivalued neurons to classification of the images of gene expression patterns," in International Conference on Computational Intelligence. Springer, 2001, pp. 291– 304.

[20] T. Akiduki, Z. Zhang, T. Imamura, and T. Miyake, "Design of multivalued cellular neural networks for associative memories," Int. J. Innov. Comput. Inf. Control, vol. 8, no. 3, pp. 1575–1589, 2012.

[21] Y. Zhang, Z. Pei, and P. Shi, "Association rule mining based on topology for attributes of multi-valued information systems," International Journal of Innovative Computing, Information and Control, vol. 9, no. 4, pp. 1679–1690, 2013.

[22] I. Aizenberg, "Neural networks based on multi-valued and universal binary neurons: theory, application to image processing and recognition," Computational Intelligence, pp. 306–316, 1999.

[23] I. N. Aizenberg, N. N. Aizenberg, and J. Vandewalle, Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications. Kluwer Academic Publishers, 2000.

[24] C. Tang, F. Chen, and X. Li, "Perceptron implementation of triple-valued logic operations," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 58, no. 9, pp. 590–594, 2011.

[25] C. Luo and X. Wang, "Algebraic representation of asynchronous multiple-valued networks and its dynamics," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 10, no. 4, pp. 927–938, 2013.

[26] A. Ngom, I. Stojmenovic, and V. Milutinovic, "Strip-a strip-based neural-network growth algorithm for learning multiple-valued functions," IEEE Transactions on Neural Networks, vol. 12, no. 2, pp. 212–227, 2001.

[27] K. Adams, D. Bell, L. P. Maguire, and J. McGregor, "Knowledge discovery from decision tables by the use of multiple-valued logic," Artificial Intelligence Review, vol. 19, no. 2, pp. 153–176, 2003.

[28] A. Ngom, I. Stojmenovic, and J. Zunic, “On the number of multilinear partitions and the computing capacity of multiple-valued multiple threshold perceptrons,” IEEE Transactions on Neural Networks, vol. 14, no. 3, pp. 469–477, 2003.

[29] C. Lee and W. Chen, “Several-valued combinational switching circuits,” Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics, vol. 75, no. 3, pp. 278–283, 1956.

[30] Q. L. Zheng and G. G. Huang, “Logic design of multiple valued logic dyl circuits,” Acta Electronica Sinica, vol. 3, p. 002, 1982.

[31] T. Watanabe and M. Matsumoto, “Design of multiple-valued logic asynchronous sequential circuits using hyperplanes,” in Proc. 14th Int. Symp. Multiple-Valued Logic, 1984, pp. 251–256.

[32] M. Perkowski, M. Marek-Sadowska, L. Jozwiak, T. Luba, S. Grygiel, M. Nowicka, R. Malvi, Z. Wang, and J. S. Zhang, “Decomposition of multiple-valued relations,” in Multiple-Valued Logic, 1997. Proceedings., 1997 27th International Symposium on. IEEE, 1997, pp. 13–18.

[33] C. M. Files and M. A. Perkowski, “New multivalued functional decomposition algorithms based on mdds,” IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems, vol. 19, no. 9, pp. 1081–1086, 2000.

[34] M. Gao, J.-H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, S. Sinha, T. Villa, and R. Brayton, “Optimization of multi-valued multi-level networks,” in Multiple-Valued Logic, 2002. ISMVL 2002. Proceedings 32nd IEEE International Symposium on. IEEE, 2002, pp. 168–177.

[35] G. Pomper and J. R. Armstrong, “Representation of multivalued functions using the direct cover method,” IEEE Transaction on Computers, vol. 30, no. 9, pp. 674–679, 1981.

[36] P. P. Tirumalai and J. T. Butler, “Minimization algorithms for multiplevalued programmable logic arrays,” IEEE Transactions on Computers, vol. 40, no. 2, pp. 167–177, 1991.

[37] G. Caruso, “A local cover technique for the minimization of multiplevalued input binary-valued output functions,” IEICE Transcations on Fundamentals of Electronics, Communications and Computer Sciences, vol. 79, no. 1, pp. 110–117, 1996.

[38] B. Fraser and G. W. Dueck, “Multiple-valued logic minimization using universal literals and cost tables,” in Proc. of the 28th International Symposium on Multiple-Valued Logic. IEEE, 1998, pp. 239–244.

[39] B. Sarif and M. Abd-El-Barr, “Fuzzy-based direct cover algorithm for systhesis of multi-valued logic functions,” in Proc. of IASTED International Conference on Circuits and System, 2008, pp. 625–630.

[40] ——, “The use of multiple connected pseudo minterms in the synthesis of MVL functions,” in Proc. of the 39th International Symposium on Multiple-Valued Logic. IEEE, 2009, pp. 145–150.

[41] M. Abd-El-Barr, “Ant colony optimisation–direct cover: a hybrid ant colony direct cover technique for multi-level synthesis of multiplevalued logic functions,” International Journal of Electronics, vol. 97, no. 12, pp. 1391–1404, 2010.

[42] M. Abd-El-Barr and E. A. Khan, “Improved direct cover heuristic algorithms for synthesis of multiple-valued logic functions,” International Journal of Electronics, vol. 101, no. 2, pp. 271–286, 2014.

[43] Z. Tang, Q. P. Cao, and O. Ishizuka, “A learning multiple-valued logic network: Algebra, algorithm, and applications,” IEEE Transactions on Computers, vol. 47, no. 2, pp. 247–251, 1998.

[44] Z. Tang, O. Ishizuka, and K. Tanno, “A learning multiple-valued logic network that can explain reasoning,” IEEJ Transactions on Electronics, Information and Systems, vol. 119, no. 8-9, pp. 970–978, 1999.

[45] A. Chowdhury and A. Singh, “Synthesis and reduced logic gate realization of multi-valued logic function using neural network deployment algorithm,” Journal of Engineering Science and Technology, vol. 11, no. 2, pp. 177–192, 2016.

[46] T. Watanabe and M. Matsumoto, “Layered MVL neural networks capable of recognizing translated characters,” in Proc. of the 22th International Symposium on Multiple-Valued Logic. IEEE, 1992, pp. 88–95.

[47] Q. P. Cao, O. Ishizuka, Z. Tang, and H. Matsumoto, “Algorithm and implementation of a learning multiple-valued logic network,” in Proc. of the 23th International Symposium on Multiple-Valued Logic. IEEE, 1993, pp. 202–207.

[48] D. Williams and G. Hinton, “Learning representations by backpropagating errors,” Nature, vol. 323, no. 6088, pp. 533–538, 1986.

[49] F.Glover and G. Kochenberger, Eds., Handbook of Metaheuristics. Kluwer, 2003.

[50] A. Ngom, D. A. Simovici, and I. Stojmenovic, “Evolutionary strategy for learning multiple-valued logic functions,” Journal of Multi-Valued Logic and Soft Computing, vol. 12, pp. 459–489, 2006.

[51] J. K. Lenstra, Local search in combinatorial optimization. Princeton University Press, 2003.

[52] Q. P. Cao, Z. Tang, R. L. Wang, and X. G. Wang, “A local search based learning method for multiple-valued logic networks,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 86, no. 7, pp. 1876–1884, 2003.

[53] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” Science, vol. 220, no. 4598, pp. 671–680, 1983.

[54] Q. P. Cao, S. C. Gao, J. C. Zhang, Z. Tang, and H. Kimura, “A stochastic dynamic local search method for learning multiple-valued logic networks,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 90, no. 5, pp. 1085–1092, 2007.

[55] S. C. Gao, Q. P. Cao, M. Ishii, and Z. Tang, “Local search with probabilistic modeling for learning multiple-valued logic networks,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 94, no. 2,

pp. 795–805, 2011.

[56] S. C. Gao, Q. P. Cao, Z. Q. Zhang, and Z. Tang, “A chaotic clonal selection algorithm and its application to synthesize multiple-valued logic functions,” IEEJ Transactions on Electrical and Electronic Engineering, vol. 5, no. 1, pp. 105–114, 2010.

[57] S. C. Gao, Q. P. Cao, C. Vairappan, J. C. Zhang, and Z. Tang, “An improved local search learning method for multiple-valued logic network minimization with bi-objectives,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 92, no. 2, pp. 594–603, 2009.

[58] C. M. Allen and D. D. Givone, “A minimization technique for multiplevalued logic systems,” IEEE Transaction on Computers, vol. C-17, no. 2, pp. 182–184, 1968.

[59] Z. Luo, M.Hu, and T.H.Chen, The Theory of Multiple-Valued Logic and Its Applications. Bei Jing: Science Press, 1992.

[60] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach,” IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, pp. 257–271, 1999.

[61] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-II,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[62] K. Deb and A. Srinivasan, “Innovization: Innovating design principles through optimization,” in Proc. of the 8th Annual Conference on Genetic and Evolutionary Computation. ACM, 2006, pp. 1629–1636.

[63] H. Q. Chen, S. Li, Q. Shi, D. M. Shen, and S. C. Gao, “Multivalued neural network trained by differential evolution for synthesizing multiple-valued functions,” in 2015 2nd International Conference on Information Science and Control Engineering (ICISCE). IEEE, 2015, pp. 332–335.

[64] K. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution: A Practical

Approach to Global Optimization. Springer Science & Business Media, 2006.

[65] S. Das and P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,” IEEE Transactions on Evolutionary Computation, vol. 15, no. 1, pp. 4–31, 2011.

[66] J. Wang, W. Zhang, and J. Zhang, “Cooperative differential evolution with multiple populations for multiobjective optimization,” IEEE Transactions on Cybernetics, vol. 46, no. 12, pp. 2848–2861, 2016.

[67] Y. Cai and J. Wang, “Differential evolution with neighborhood and direction information for numerical optimization,” IEEE Transactions on Cybernetics, vol. 43, no. 6, pp. 2202–2215, 2013.

[68] G. Tian, Y. Ren, and M. Zhou, “Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 11, pp. 3009–3021, 2016.

[69] J. H. Wang, Y. Zhou, Y. Wang, J. Zhang, C. P. Chen, and Z. B. Zheng, “Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms,” IEEE Transactions on Cybernetics, vol. 46, no. 3, pp. 582–594, 2016.

[70] J. H. Wang, J. J. Liao, Y. Zhou, and Y. Q. Cai, “Differential evolution enhanced with multiobjective sorting-based mutation operators,” IEEE Transactions on Cybernetics, vol. 44, no. 12, pp. 2792–2805, 2014.

[71] Y. Zhou and J. Wang, “A local search-based multiobjective optimization algorithm for multiobjective vehicle routing problem with time windows,” IEEE Systems Journal, vol. 9, no. 3, pp. 1100–1113, 2015.

[72] Yalan Zhou, Jiahai Wang, Jian Chen, Shangce Gao, Luyao Teng, “Ensemble of many-objective evolutionary algorithms for many-objective problems,” Soft Computing, Vol.21, No.9, pp.2407-2419, May 2017. (SCI:, EI: 20155001672081)

[73] Shuangbao Song, Shangce Gao, Xingqian Cheng, Dongbao Jia, Xiaoxiao Qian, and Yuki Todo, “AIMOES: Archive Information Assisted Multi-objective Evolutionary Strategy for Ab Initio Protein Structure Prediction,” Knowledge-Based Systems, vol. 146, pp.58-72, April 2018.

[74] Shangce Gao, Shuangbao Song, Jiujun Cheng, Yuki Todo, and MengChu Zhou, “Incorporation of Solvent Effect into Multi-objective Evolutionary Algorithm for Improved Protein Structure Prediction,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 15, no.4, pp.1365-1378, July-Aug. 2018.

[75] Jiahai Wang, Binzhong Cen, Shangce Gao, Zizhen Zhang, and Yuren Zhou, “Cooperative Evolutionary Framework With Focused Search for Many-Objective Optimization,” IEEE Transactions on Emerging Topics in Computational Intelligence, DOI: 10.1109/TETCI.2018.2849380, 2018.

[76] Shuangbao Song, Junkai Ji, Xingqian Chen, Shangce Gao, Zheng Tang, and Yuki Todo, “Adoption of an improved PSO to explore a compound multi-objective energy function in protein structure prediction,” Applied Soft Computing, vol.11, pp.539-551, Nov. 2018.

[77] Jiahai Wang, Liangsheng Yuan, Zizhen Zhang, Shangce Gao, Yuyan Sun, and Yalan Zhou, “Multiobjective Multiple Neighborhood Search Algorithms for Multiobjective Fleet Size and Mix Location-Routing Problem With Time Windows,” IEEE Transactions on Systems, Man and Cybernetics: Systems, DOI: 10.1109/TSMC.2019.2912194, 2019.

[78] Jian Sun, Shangce Gao, Hongwei Dai, Jiujun Cheng, MengChu Zhou, and Jiahai Wang, “Bi-objective Elite Differential Evolution for Multivalued Logic Networks,” IEEE Transactions on Cybernetics, DOI: 10.1109/TCYB.2018.2868493, 2018.

[79] Shangce Gao, Jianchen Zhang, Xugang Wang, and Zheng Tang, “Multi-layer Neural Network Learning Algorithm based on Random Pattern Search Method,” International Journal of Innovative Computing, Information and Control, vol.5, no.2, pp.489-502, February, 2009.

[80] Catherine Vairappan, Hiroki Tamura, Shangce Gao, and Zheng Tang, “Batch Type Local Search-based Adaptive Neuro-Fuzzy Inference System (ANFIS) with Self-Feedbacks to Time Series Prediction,” Neurocomputing, vol.72, pp.1870-1877, March 2009.

[81] Catherine Vairappan, Shangce Gao, Hiroki Tamura, and Zheng Tang, “An Improved Learning of Local Search for Fuzzy Controller Network,” International Journal of Innovative Computing, Information and Control, vol.5, no.4, pp.1101-1113, April 2009.

[82] Junyan Yi, Gang Yang, Shangce Gao and Zheng Tang, “A Shrinking Chaotic Maximum Neural Network for Maximum Clique Problem,” International Journal of Innovative Computing, Information and Control, vol.5, no.5, pp.1513-1229, May, 2009.

[83] Junyan Yi, Gang Yang, Shangce Gao and Zheng Tang, “Transiently chaotic neural network based on switched cooling and its application to maximum clique problem”, International Journal of Innovative Computing, Information and Control, vol.5, no.6, pp.1569-1586, June, 2009.

[84] Zhiqiang Zhang, Shangce Gao, Gang Yang, Fangjia Li and Zheng Tang，“An Algorithm of Supervised Learning for Elman Neural Network, ” International Journal of Innovative Computing, Information and Control, vol. 5, no.10 (A), pp.2997-3011, October 2009.

[85] Zhiqiang Zhang, Zheng Tang, Shangce Gao, and Gang Yang, “Training Elman Neural Network for Dynamical System Identification Using Stochastic Dynamic Batch Local Search Algorithm,” International Journal of Innovative Computing, Information and Control, vol.6, no.4, pp.1883-1892, April, 2010.

[86] Zhiqiang Zhang, Zheng Tang, Shangce Gao, and Gang Yang, “Training Elman Neural Network for Dynamic System Identification Using an Adaptive Local Search Algorithm,” International Journal of Innovative Computing, Information and Control, vol.6, no.5, pp.2233-2244, May, 2010.

[87] Rong-Long Wang, Shangce Gao, and Zheng Tang, “Solving the Maximum Cut Problem Using Two-Phase Hopfield Neural Network,” International Journal of Innovative Computing, Information and Control, vol.6, no.8, pp.3573-3583, August, 2010.

[88] Zhiqiang Zhang, Zheng Tang, Shangce Gao, and Gang Yang, “An Algorithm of Chaotic Dynamic Adaptive Local Search Method for Elman Neural Network,” International Journal of Innovative Computing, Information and Control, vol.7, no.2, pp.647-656, February, 2011.

[89] Shangce Gao, Hongwei Dai, Gang Yang, and Zheng Tang, “ A Novel Clonal Selection Algorithm and Its Application to Traveling Salesman Problems,” IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol.E90-A, no.10, pp.2318-2325, October 2007.

[90] Shangce Gao, Zheng Tang, Hongwei Dai, and Jianchen Zhang, “An Improved Clonal Selection Algorithm and Its Application to Traveling Salesman Problems,” IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol.E90-A, no.12, pp.2930-2938, December 2007.

[91] Shangce Gao, Zheng Tang, Hongwei Dai and Jianchen Zhang, “A Hybrid Clonal Selection Algorithm,” International Journal of Innovative Computing, Information and Control, vol.4, no.4, pp.995-1008, April, 2008.

[92] Shangce Gao, Wei Wang, Hongwei Dai, Fangjia Li, and Zheng Tang, “Improved clonal selection algorithm combined with ant colony optimization,” IEICE Transactions on Information and Systems, vol.E91-D, no.6, pp.1813-1823, June 2008.

[93] Shangce Gao, Hongwei Dai, Jianchen Zhang, and Zheng Tang, “An Expanded Lateral Interactive Clonal Selection Algorithm and Its Application,” IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol.E91-A, no.8, pp.2223-2231, August, 2008.

[94] Hongwei Dai, Yu Yang, Cunhua Li, Jun Shi, Shangce Gao, and Zheng Tang, “Quantum Interference Crossover-Based Clonal Selection Algorithm and Its

Application to Traveling Salesman Problem," IEICE Transactions on Information and Systems, vol.E92-D, no.1, pp.78-85, January, 2009.

[95] Shangce Gao, Zheng Tang and Catherine Vairappan, "An Effective Immune Algorithm for Multiple-Valued Logic Minimization Problems," International Journal of Innovative Computing, Information and Control, vol.5, no.11(A), pp.3961-3969, November, 2009.

[96] Wei Wang, Shangce Gao, and Zheng Tang, "Improved Pattern Recognition with Complex Artificial Immune System," Soft Computing, vol.13, no.12, pp.1209-1217, December 2009.

[97] Shangce Gao, Rong-Long Wang, Hiroki Tamura, and Zheng Tang, "A Multi-layered Immune System for Graph Planarization Problem," IEICE Transactions on Information and Systems, vol.E92-D, no.12, pp.2498-2507, December 2009.

[98] Shangce Gao, Zhiqiang Zhang, Qiping Cao, and Zheng Tang, "A Chaotic Clonal Selection Algorithm and Its Application to Synthesize Multiple-Valued Logic Function," IEEJ Transactions on Electrical and Electronic Engineering, vol.5, no.1, pp.105-114, January, 2010.

[99] Shangce Gao, Rong-Long Wang, Masahiro Ishii, and Zheng Tang, "An Artificial Immune System with Feedback Mechanisms for Effective Handling of Population Size," IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, vol.E93-A, no.2, pp.532-541, February 2010.

[100] Shuaiqun Wang, Shangce Gao, Aorigele, Yuki Todo, Zheng Tang, "A Multi-learning Immune Algorithm for Numerical Optimization," IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, Vol.E98-A,No.1,pp.362-377, January 2015.

[101] Zhe Xu, Yirui Wang, Sheng Li, Yanting Liu, Yuki Todo, and Shangce Gao, "Immune Algorithm Combined with Estimation of Distribution for Traveling Salesman Problem," IEEJ Transactions on Electrical and Electronic Engineering, vol. 11, no. S1, pp.S142-S154, June 17, 2016.

[102] Yu Yang, Hongwei Dai, Shangce Gao , Yirui Wang, Dongbao Jia, and Zheng Tang, “Complete Receptor Editing Operation Based Quantum Clonal Selection Algorithm for Optimization Problems,” IEEJ Transactions on Electrical and Electronic Engineering, vol.14, no.3, pp.411-421, 2019.

[103] Shangce Gao, Yuki Todo, Tao Gong, Gang Yang, and Zheng Tang, “Graph Planarization Problem Optimization based on Triple-valued Gravitational Search Algorithm,” IEEJ Transactions on Electrical and Electronic Engineering, vol.9, no.1, pp.39-48, January 2014.

[104] Shangce Gao, Catherine Vairappan, Yan Wang, Qiping Cao, and Zheng Tang, “Gravitational Search Algorithm Combined with Chaos for Unconstrained Numerical Optimization,” Applied Mathematics and Computation, vol.231, pp.48-62, March 2014.

[105] Zhenyu Song, Shangce Gao, Yang Yu, Jian Sun, and Yuki Todo, “Multiple Chaos Embedded Gravitational Search Algorithm,” IEICE Transactions on Information and Systems, vol.E100-D, No.4, pp.888-900, Apr. 2017.

[106] Junkai Ji, Shangce Gao, Shuaiqun Wang, Yajiao Tang, Hang Yu, and Yuki Todo, “Self-adaptive Gravitational Search Algorithm with A Modified Chaotic Local Search,” IEEE Access, vol. 5, pp. 17881-17895, September 2017.

[107] Yirui Wang, Yang Yu, Shangce Gao, Haiyu Pan, and Gang Yang, “A hierarchical gravitational search algorithm with an effective gravitational constant,” Swarm and Evolutionary Computation, vol.46, pp.118-139, 2019.

[108] Shangce Gao, Yirui Wang, Jiujun Cheng, and Yasuhiro Inazumi, and Zheng Tang, “Ant colony optimization with clustering for solving the dynamic location routing problem,” Applied Mathematics and Computation, vol.285, pp. 149–173, July 2016.

[109] Shuaiqun Wang, Aorigele, Guanjun Liu, and Shangce Gao, “A hybrid discrete imperialist competition algorithm for fuzzy job-shop scheduling problems,” IEEE Access, vol.4, pp.9320-9331, November 2016.

[110] Aorigele, Zheng Tang, Yuki Todo, and Shangce Gao, “A hybrid discrete imperialist competition algorithm for gene selection for microarray data,” Current Proteomics, vol. 15, no.2, pp.99-110, 2018.

[111] Shuaiqun Wang, Wei Kong, Aorigele, Jin Deng, Shangce Gao, and Weiming Zeng, “Hybrid Feature Selection Algorithm mRMR-ICA for Cancer Classification from Microarray Gene Expression Data,” Combinatorial Chemistry & High Throughput Screening, vol.21, no.6, pp.420-430, July 2018.

[112] Yirui Wang, Shangce Gao, Yang Yu, and Zhe Xu, “The discovery of population interaction with a power law distribution in brain storm optimization,” Memetic Computing, vol. 11, pp. 65-87, 2019.

[113] Yang Yu, Shangce Gao, Shi Cheng, Yirui Wang, Shuangyu Song, and Fenggang Yuan, “CBSO: A Memetic Brain Storm Optimization with Chaotic Local Search,” Memetic Computing, vol.10, no.4, pp.353-367, December 2018.

[114] Yang Yu, Shangce Gao, Yirui Wang, Jiujun Cheng, and Yuki Todo, “ASBSO: An Improved Brain Storm Optimization with Flexible Search Length and Memory-based Selection,” IEEE Access, vol.6, no.1, pp. 36977-36994, December 2018.

[115] Zijun Sha, Lin Hu, Yuki Todo, Junkai Ji, Shangce Gao, and Zheng Tang, “The Breast Cancer Classifier using Neuron Model with Dendritic Nonlinearity,” IEICE Transactions on Information and Systems, Vol.E98-D, No.7, pp.1365-1376, Jul. 2015.

[116] Junkai Ji, Shangce Gao, Jiujun Cheng, Yuki Todo, and Zheng Tang, “An approximate logic neuron model with a dendritic structure,” Neurocomputing, vol.173, no.3, pp.1775-1783, 15 January, 2016.

[117] Tianle Zhou, Shangce Gao, Jiahai Wang, Chaoyi Chu, Yuki Todo and Zheng Tang, “Financial time series prediction using a dendritic neuron model,” Knowledge-Based Systems, vol.105, pp.214-224, August 1, 2016.

[118] Tao Jiang, Shangce Gao, Dizhou Wang, Junkai Ji, Yuki Todo, and Zheng

Tang, “A Neuron Model with Synaptic Nonlinearities in a Dendritic Tree for Liver Disorders,” IEEJ Transactions on Electrical and Electronic Engineering, vol.12, no.1, pp.105-115, 2017.

[119] Wei Chen, Jian Sun, Shangce Gao, Jiu-jun Cheng, Jiahai Wang, and Yuki Todo, “Using A Single Dendritic Neuron to Forecast Tourist Arrivals to Japan,” IEICE Transactions on Information and Systems, vol.E100-D, No.1, pp.190-202, Jan. 2017.

[120] Ying Yu, Yirui Wang, Shangce Gao, and Zheng Tang, “Statistical Modeling and Prediction for Tourism Economy Using Dendritic Neural Network,” Computational Intelligence and Neuroscience, vol. 2017, Article ID 7436948, 9 pages, Jan. 2017.

[121] Yajiao Tang, Junkai Ji, Shangce Gao, Hongwei Dai, Yang Yu, and Yuki Todo, “A pruning neural network model for credit classification analysis,” Computational Intelligence and Neuroscience, vol.2018, Article ID 9390410, 22 pages, Feb. 2018.

[122] Shangce Gao, Mengchu Zhou, Yirui Wang, Jiujun Cheng, Hanaki Yachi and, Jiahai Wang, “Dendritic neural model with global learning algorithms for classification, approximation and prediction,” IEEE Transactions on Neural Networks and Learning Systems, vol.30, no.2, pp. 601-614, 2019.

[123] Junkai Ji, Shuangbao Song, Yajiao Tang, Shangce Gao, Zheng Tang, and Yuki Todo, “Approximate logic neuron model trained by states of matter search algorithm,” Knowledge-based Systems, vol. 163, pp. 120-130, January 2019.