A Chaotic Clonal Selection Algorithm and Its Application to Synthesize Multiple-Valued Logic Function

doi: 10.1002/tee.20500

IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2010; **5**: 105–114
Published online in Wiley InterScience (www.interscience.wiley.com). DOI:10.1002/tee.20500

Paper

A Chaotic Clonal Selection Algorithm and its Application to Synthesize Multiple-Valued Logic Functions

Shangce Gao[a*], Member
Qiping Cao[**], Non-member
Zhiqiang Zhang[*], Non-member
Zheng Tang[*], Non-member

In this paper, a chaotic clonal selection algorithm (CCSA) is proposed to synthesize multiple-valued logic (MVL) functions. The MVL function is realized in a multiple-valued sum-of-products expression where product is indicated by *MIN* and sum by *TSUM*. The proposed CCSA, in which chaos is incorporated into the clonal selection algorithm to initialize antibodies and maintain the population diversity, is utilized to learn a given target MVL truth table. Furthermore, an adaptive length strategy of antibodies is also introduced to reduce the computational complexity, whereas an improved affinity function enables the algorithm to find less product terms for an MVL function. Simulation results based on a large number of MVL functions demonstrate the efficiency of the proposed method when compared with other traditional methodologies.

Keywords: multiple-valued logic, clonal selection algorithm, minimization, chaos

Received 10 November 2008; Revised 13 February 2009

1. Introduction

Multiple-valued logic (MVL), whose truth values are more than two, has been used in logic circuits and systems for many years. Compared with the binary logic whose development of creating digital networks technology is physically limited, the microelectronic technology with MVL provides us with opportunities to build very complex digital circuits and systems at a relatively low cost, and provides a diversity of logic in building blocks [1,2]. MVL minimization is an important technique for reducing the area required by a programmable logic array [3]. The decision for this task is possible due to the minimization of logic function and thus the algorithms depend on the functional completeness basis in which the circuit for the given function will be realized.

Several methodologies have been proposed in the literature for synthesis of MVL functions, such as deterministic algorithms [4], direct cover-based approaches [5–7], back-propagation methods (BP) [8–10], local search methods (LS) [11,12], genetic algorithms (GA) [13–15], and so on. These methodologies can be divided into two categories: the deterministic algorithms and direct cover-based approaches belong to technology-dependent approaches, whereas BP methods, LS methods and GA belong to technology-independent approaches.

The basic idea in deterministic algorithms is to find all the prime implicants of an MVL function. However, the number of prime implicants is so large that absolute minimization of MVL functions requires computation time on the order of days [16]. Compared with deterministic algorithms, direct cover-based approaches utilize the cover selection which is integrated into the prime implicant generation process and therefore not all prime implicants are necessarily generated. Although direct cover-based approaches provide a promising alternative method to minimize MVL functions, they are still time-consuming [3]. Contrary to the technology-dependent methods, the technology-independent approaches try to find the optimal or near-optimal solutions for the problem. They need less computational time and have the merits of excellent optimization ability and robustness in various instances. In earlier days, several error BP-based algorithms were proposed to emulate MVL functions. However, during learning, many nodes and parameters such as weights and thresholds were usually necessary to approximate an MVL function and any knowledge which was available prior to training was unable to be used. Compared with the BP algorithms, the LS methods have easy hardware implementation and can widely make use of the prior knowledge we have on MVL while constructing an MVL network [11,17]. Recently, GA [18] were also proposed to constitute an important avenue for solving such a problem. They are population-based algorithms and have demonstrated their applicability and efficiency in synthesizing MVL functions [14,15]. Nevertheless, none of these approaches provides absolute optimum results for synthesizing MVL functions since the search space is too large to be explored.

Clonal selection algorithm (CSA) [19], inspired by the basic features of adaptive immune response to antigenic stimulus, can exploit and explore the solution space in parallel and effectively. CSA has been applied to pattern recognition and optimization problems, such as image classification [20], stack filters designing [21], job-shop schedules [22], traveling salesman problems [23,24], multi-objective optimization problems [25,26], and so on. However, the applications on MVL are rarely reported in the literature.

[a] Correspondence to: Shangce Gao
E-mail: cupid19831018@hotmail.com

[*] Graduate School of Innovative Life Science, University of Toyama, Gofuku 3190, Toyama-shi 930-8555, Japan
[**] Tateyama Institute of System, Toyama-shi 930-0016, Japan

In this paper, CSA is utilized to synthesize MVL functions. On the basis of the *TSUM* expression [27], any MVL function can be realized in a sum-of-products form by using *TSUM*, *MIN*, window literal operators and a set of constants. In CSA, an antibody consisting of all the tunable parameters in the operators is regarded as a solution candidate. In order to evaluate each antibody, an affinity function involving two metrics of correctness and optimality is utilized. Correctness deals with the functionality of the representation, whereas optimality deals with the quality of solution, i.e. how many product terms are required to realize the MVL function. Furthermore, since the length of the antibody is an important design parameter and mainly determines the solution space, an adaptive length (AL) strategy is proposed. During search, the antibody with the higher affinity produced by CSA is corresponding to a set of optimal parameters and based on these parameters the MVL function can be minimized. In addition, in order to maintain the diversity of antibodies and improve the searching performance of CSA, chaos is integrated to construct a chaotic clonal selection algorithm (CCSA). The numerical simulation results based on several MVL functions demonstrate the effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows: in Section 2, we provide a general description of the multiple-valued logic function minimization problem. In Section 3, the clonal selection algorithm and the chaotic system are briefly introduced, respectively. The proposed CCSA is presented in Section 4. In Section 5, we validate our model by applying it to a number of MVL functions. Finally we give some general remarks to conclude this paper.

2. Problem Representation

Before introducing the problem representation, some preliminaries of the MVL functions are interpreted. One of the functional completeness multiple-valued algebras for any radix is the *TSUM* expression [27]. On the basis of this algebra, MVL function can be formulated as follows. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n variables, where x_i takes on values from $R = \{0, 1, \ldots, r-1\}$. An n-variable r-valued function $F(X)$ is a mapping $F : R^n \rightarrow R^1$. The function is said to have n multiple-valued inputs, and variable x_i is said to take on one of r possible values. Each element in the domain of the function is called a *minterm* of the function. An enumeration of all minterms with the value of the function is called a *truth table*.

Any MVL function can be realized based on a sum-of-products expansion in which each input vector is individually selected, corresponding to the canonical sum-of-minterms realization of the binary case. The operators utilized in the realization are *TSUM*, *MIN* and window literals are defined as follows:

(1) *TSUM* and *MIN* operators:

$$\begin{aligned} x_1 \vee x_2 \vee \ldots \vee x_n &= TSUM(x_1, x_2, \ldots, x_n) \\ &= MIN(x_1 + x_2 + \ldots + x_n, r-1) \\ x_1 \wedge x_2 \wedge \ldots \wedge x_n &= MIN(x_1, x_2, \ldots, x_n) \\ &= \text{the smallest value of } (x_1, x_2, \ldots, x_n) \end{aligned}$$

(2) Window literal operators:

$$x_i(a,b) = \begin{cases} r\text{-}1 & a \le x_i \le b \\ 0 & \text{otherwise} \end{cases}$$

where x_i $(i = 1, 2, \ldots, n)$ are the n r-valued variables of the set X, $\vee$ $(\wedge)$ denotes the *TSUM* (*MIN*) operation and $x_i(a, b)$ represents literal function. The parameters a and b are called window parameters. In this condition, any n-variable r-valued function can be synthesized in a sum-of-products form:

$$\begin{aligned} F(x_1, \ldots, x_n) = \vee_{j=1}^{n^r} (c_i \wedge x_1(a_{1j}, b_{1j}) \wedge \ldots \\ \wedge\ x_i(a_{ij}, b_{ij}) \wedge \ldots \wedge x_n(a_{nj}, b_{nj})) \end{aligned} \quad (1)$$

where $x_1, x_2, \ldots, x_n$ are r-valued variables. Window literal parameters (a_{ij}, b_{ij}) belongs to $\{0, 1, 2, \ldots, r-1\}$ and the *Constant* (also referred as biasing parameters) c_i belongs to $\{0, 1, 2, \ldots, r-1\}$. In this study, the set consisting of {*TSUM*, *MIN*, Window literal, Constant} is used. A *product term* can be defined as a *MIN* operation on a set of window literals. For example, the term $c_i \wedge x_1(a_{1j}, b_{1j}) \ldots \wedge x_n(a_{nj}, b_{nj})$ is a product term. Initially, there are n^r product terms of an MVL function in the sum-of-products form.

Then, the problem can be represented as follows. Given a target truth table, the objective of the *MVL function minimization* problem is to find such an MVL function with a minimum number of product terms (optimality) that can hit all the values in the truth table successfully (correctness).

In order to solve this problem, the system parameters including a_{ij}, b_{ij} and c_j should be tuned with appropriate values. All these parameters can be involved within a vector V defined as follows:

$$\begin{aligned} V = \{&a_{11}, \ldots, a_{1m}, \ldots, a_{n1}, \ldots, a_{nm}, \\ &b_{11}, \ldots, b_{1m}, \ldots, b_{n1}, \ldots, b_{nm}, c_1, \ldots, c_m\} \end{aligned}$$

Here, m is the number of product terms in the MVL function (its value will be discussed in Section 4). For example, a target truth table of a two-variable four-valued MVL function is given in Table I. It has 11 non-zero minterms. F_1 and F_2 as shown in Eq. (2) and Eq. (3), respectively, are two of the solutions. Both of them can hit the truth table successfully. However, F_1 which is its *canonical realization* [3] having 11 product terms is regarded as a local optimum, whereas the minimized function F_2 only having 5 product terms is a global optimum solution.

Table I. Example of a quaternary 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

$$\begin{aligned} F_1(x_1, x_2) = {} & 1 \wedge x_1(0,0) \wedge x_2(0,0) \vee 1 \wedge x_1(0,0) \wedge x_2(1,1) \\ & \vee 1 \wedge x_1(1,1) \wedge x_2(1,1) \vee 1 \wedge x_1(3,3) \wedge x_2(0,0) \\ & \vee 1 \wedge x_1(3,3) \wedge x_2(1,1) \vee 1 \wedge x_1(3,3) \wedge x_2(2,2) \\ & \vee 1 \wedge x_1(1,1) \wedge x_2(3,3) \vee 2 \wedge x_1(3,3) \wedge x_2(3,3) \\ & \vee 3 \wedge x_1(0,0) \wedge x_2(2,2) \vee 3 \wedge x_1(1,1) \wedge x_2(2,2) \\ & \vee 3 \wedge x_1(2,2) \wedge x_2(2,2) \end{aligned} \quad (2)$$

$$\begin{aligned} F_2(x_1, x_2) = {} & 1 \wedge x_2(3,3) \vee 3 \wedge x_1(2,2) \wedge x_2(0,2) \\ & \vee 2 \wedge x_1(3,3) \wedge x_2(3,3) \vee 1 \wedge x_1(0,2) \wedge x_2(0,0) \\ & \vee 1 \wedge x_1(1,3) \wedge x_2(1,1) \end{aligned} \quad (3)$$

Generally speaking, for any n-variable r-valued logic function, there are n^r elements in the MVL truth table and we have to adjust $(2n+1)m$ system parameters for the synthesis of an MVL function. Nevertheless, there are maximum $r^{(2n+1)\cdot(r^n)}$ combination of

these parameters. It is impossible to enumerate all the possible combination in a reasonable time. Thus, an efficient minimization method is necessary and important.

3. CSA and Chaotic System

To make the paper self-explanatory, before actually interpreting the CSA, the basic ideas of the clonal selection principle are briefly explained.

The clonal selection theory developed by Burnet [28] interprets the essential features which contain sufficient diversity, discrimination of self and non-self and long-lasting immunological memory in the response of lymphocytes in the face of an antigenic stimulus. When a biological immune system is exposed to invading antigens, some sub-population of its bone marrow-derived cells (B lymphocytes) can recognize the antigen with a certain affinity (degree of match), the B lymphocytes will be stimulated to proliferate (divide) and eventually mature into terminal (non-dividing) antibody secreting cells, called plasma cells. Proliferation of the B lymphocytes is a mitotic process whereby the cells divide themselves, creating a set of clones identical to the parent cell. The proliferation rate is directly proportional to the affinity level, i.e. the higher affinity levels of B lymphocytes, the more of them will be readily selected for cloning and cloned in larger numbers. More specifically, during asexual reproduction, the repertoire of antigen-activated B cells is diversified by hypermutation and in addition to which, a portion of the least stimulated lymphocytes is replaced per cell generation by newcomer cells from the bone marrow and join the pool of available antigen recognizing cells to maintain the diversity of the population.

The CSA based on the clonal selection principle can be briefly interpreted as follows. From an immunological standpoint, the CSA is developed in which various immune system aspects are taken into account such as maintenance of the memory cells, selection and cloning of the most stimulated cells, death of non-stimulated cells, re-selection of the clones with higher affinity and generation of diversity. From a computational perspective, the clonal selection idea leads to algorithms that iteratively improve candidate solutions to a given problem through a process of cloning, mutation and selection.

The general steps of the CSA is described as follows. First, an antibody repertoire is generated and the affinity of each antibody in the repertoire is calculated. A few of the highest affinity antibodies from the repertoire are selected and cloned. The clonal rate is directly proportional to the affinity level. Then the repertoire of clones is submitted to a mutation process. Finally, some low affinity antibodies in the mutated clones are replaced by new introduced antibodies. In the original CSA, in addition to the initial antibody repertoire generated randomly, the low affinity antibodies have to be replaced by new random antibodies during the mutation process. No doubt these random antibodies will reduce the complexity of algorithm. Nevertheless, random antibodies have a bad influence on the diversity of repertoire. As a result, the search within affinity landscape will become premature and cannot produce better solutions.

On the other hand, chaos which exhibits bounded dynamic unstable, pseudo random, ergodic and non-period behavior enables the search system more capable of hill-climbing and escaping from local optima [29]. Furthermore, recent study revealed that chaos also exits in the process of immune response [30].

In this paper, a CCSA integrating CSA and chaos is proposed. There are several chaotic systems, such as Logistic map [31], mapping drawn from chaotic neuron [32], Tent mapping [33], Lorenz System [34], Chen System [35], Lű System [36], and so on. The first three referenced chaotic systems can show good chaotic properties. If a designed algorithm needs the candidate points to distribute in search space as much as possible, the three chaotic systems can meet this need. It has been demonstrated that these three chaotic systems display better randomness than other systems [37,38]. We adopt the Logistic map as a studying case. Nevertheless, it does not indicate that the adopted Logistic map outperforms the other two systems. Further works can be considered regarding to this issue.

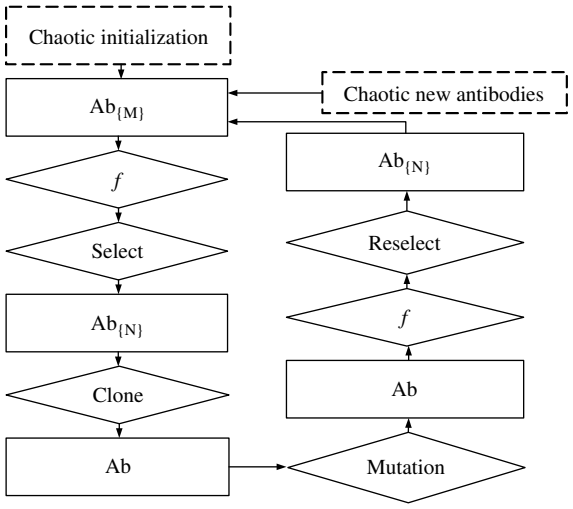


Fig. 1. Flowchart of CCSA

The Logistic map is defined by

$$\lambda^{s+1} = 4\lambda^s(1-\lambda^s),\ \ \lambda^s \in (0,1), s = 1, 2\ldots \tag{4}$$

where λ^s is the value of the variable λ at the sth iteration. When the initial value of λ (i.e., $\lambda_0 \in (0,1)$) is not equal to $\{0, 0.25, 0.5, 0.75, 1\}$, the above system enters into a chaotic state.

4. Minimization Method by CCSA

In CCSA, both antibody repertoire initialization and new introduced antibodies are generated based on chaotic systems. The flowchart of the CCSA is illustrated in Fig. 1.

4.1. Antibody and its length In our approach, for a given n-variable r-valued MVL function, an antibody Ab is expressed as the vector of **V**, that is

$$\text{Ab} = \{a_{11}, \ldots, a_{nm}, b_{11}, \ldots, b_{nm}, c_1, \ldots, c_m\}$$

The length of Ab is $L = (2n+1)m$.

The length of the antibody L is an important design parameter in the CCSA. It mainly determines the search space of the problem. Since n is a constant for a given MVL function, L is decided by m. If m is too small, the CCSA will unlikely find the best solution, i.e. the optimal solution might not be contained in the search space. On the other hand, if m is too larger, the search space is enlarged and the CCSA may waste a lot of time in finding a solution in the no-solution space. A straightforward approach is to use the length of truth table as the length of antibody. But this does not guarantee that the CCSA will find the best solution in an effective way. In this study, two different lengths of antibody are verified. One is a static length (SL) and the other is an AL. They are defined as follows.

The SL indicates the length of truth table, i.e. the number of all possible minterms of an MVL function. For example, for two-variable four-valued functions, the SL is equal to $2^4 = 16$. According to the completeness theory [39], any MVL function can be represented in its *canonical realization* form, only using the non-zero minterms of the MVLs truth table. An example is shown in (Eq. (2)). Following this observation, the AL can be set as the number of non-zero values in the truth table (NM), for the function shown in Table I is 11. Since it has been demonstrated that the maximum number of product terms to cover any two-variable four-valued function is 12 in reference [16], AL can be adapted in this special case. The two length strategies are summarized in the following.

(I) Static length (SL):

$$\mathrm{SL} = n^r, \text{ when } \mathrm{MVL}_{n,r} \tag{5}$$

(II) Adaptive length (AL):

$$\mathrm{AL} = \begin{cases} \mathrm{NM}, & \text{when } \mathrm{MVL}_{n,r} \\ & \text{and } n \neq 2 \text{ and } r \neq 4 \\ \mathrm{NM}, & \text{when } \mathrm{MVL}_{2,4} \text{ and } \mathrm{NM} < 12 \\ 12, & \text{when } \mathrm{MVL}_{2,4} \text{ and } \mathrm{NM} \geq 12 \end{cases} \tag{6}$$

Here, $\mathrm{MVL}_{n,r}$ denotes the n-variable r-valued logic functions ($n = 1, 2, \ldots; r = 1, 2, \ldots$).

4.2. Antibody population and affinity assignment An antibody population which is an M-dimensional group of antibody Ab can be defined as

$$\mathrm{Ab}_{\{M\}} = \{\mathrm{Ab}_1, \mathrm{Ab}_2, \ldots, \mathrm{Ab}_M\}$$

where the positive integer M is the size of antibody population $\mathrm{Ab}_{\{M\}}$.

In order to evaluate each antibody Ab_i, the error function is defined as:

$$\begin{aligned} E(\mathrm{Ab}_i) &= E_\mathrm{c} + \alpha E_0 \\ &= \sum_p^P (O_p - T_p)^2 + \frac{1}{r^n} E_0 \end{aligned} \tag{7}$$

where E_c denotes the correctness, whereas E_o indicates the optimality (actually the number of product terms). $\alpha = 1/r^n$ determines the relationship between the correctness and the optimality. O_p and T_p represent the pth actual output value of an MVL function and the target value in the truth table corresponding to the pth input pattern $(x_1, x_2, \ldots, x_n)_p$, respectively. P is the number of the total input patterns. As all the nodes with window literal parameters $a_{ij} > b_{ij}$ or biasing parameter $c_i = 0$ do not contribute to the output and thus can be deleted from the function, E_o is calculated by counting the number of the remaining product terms.

Furthermore, since the objective of CCSA is to maximize the affinity of the antibody, while in MVL minimization problem it is to minimize the error function. Thus, in our approach, we define the affinity of the antibody is in opposition to the error, i.e.

$$A(\mathrm{Ab}_i) = \frac{1}{E(\mathrm{Ab}_i)}$$

where $A(.)$ is the affinity function.

4.3. Solution space transformation Each antibody consists of L optimization variables, from a_{11} to c_m, and each optimization variable corresponds to a chaotic variable in (4). Since the range of optimization variables is $[0, r-1]$ and that of chaotic variables is (0, 1), the mapping relationship between them must be determined. The ergodic space of the chaos system (4) is mapped to the solution space of the MVL function minimization problem by (8), and thus the L optimization variables can be expressed by the L chaotic variables.

$$X_i = \lfloor r\lambda_i \rfloor, \ i = 1, 2, \ldots, L \tag{8}$$

where X_i denotes the ith optimization variable of the problem, i.e. the ith element in the antibody, and λ_i generated by (4) is its corresponding chaotic variable. The function $\lfloor . \rfloor$ removes the fractional of the independent variable and returns the resulting integer value. In this condition, each antibody can be represented as $\mathrm{Ab} = \{a_{11}, \ldots, c_m\} = \{\lfloor r\lambda_1 \rfloor, \ldots, \lfloor r\lambda_L \rfloor\}$.

4.4. Chaotic initialization The L chaotic variables are generated by the following mapping:

$$\lambda_i^{s+1} = 4\lambda_i^s(1 - \lambda_i^s), \ i = 1, 2, \ldots, L; s = 1, 2, \ldots \tag{9}$$

where i is the serial number of chaotic variables. Initially, the antibody population $\mathrm{Ab}_{\{M\}}$ is generated as follows. First, let $s = 1$ and given that the L chaotic variables have different initial values λ_i^1 ($i = 1, 2, \ldots, L$), the first antibody in the population can be constructed as: $\mathrm{Ab}_1 = \{\lfloor r\lambda_1^1 \rfloor, \lfloor r\lambda_2^1 \rfloor, \ldots, \lfloor r\lambda_L^1 \rfloor\}$. Then, let $s = 2, 3, \ldots, M$, the other $M - 1$ antibodies in the population are produced (9). After initialization, M antibodies ($\mathrm{Ab}_i = \{\lfloor r\lambda_1^i \rfloor, \lfloor r\lambda_2^i \rfloor \ldots, \lfloor r\lambda_L^i \rfloor\}$, $i = 1, 2, \ldots, M$) are generated.

4.5. Elitist antibodies selection: T^S According to the clonal selection principle, a portion of the antibodies ($\mathrm{Ab}_1, \mathrm{Ab}_2, \ldots, \mathrm{Ab}_N$) ($N < M$) with higher affinity (without loss of generality, we suppose that $A(\mathrm{Ab}_1) > A(\mathrm{Ab}_2) > \ldots > A(\mathrm{Ab}_M)$) are selected as elitist antibodies to undergo the following proliferation and mutation. Each antibody Ab_i is subjected into an *elite pool* where the proliferation and mutation are carried out. The other $(M - N)$ antibodies are eliminated (clonal deletion or apoptosis). This process is denoted as T^S.

4.6. Proportional cloning: T^C In immunology, cloning means asexual propagation so that a group of identical cells can be descended from a single common ancestor, such as a bacterial colony whose members arise from a single original cell as the result of mitosis. In this study, the proportional cloning T^C on the current population $\mathrm{Ab}_{\{N\}}$ is defined as

$$\begin{aligned} &T^\mathrm{C}(\mathrm{Ab}_1 + \mathrm{Ab}_2 + \cdots + \mathrm{Ab}_N) \\ &= T^\mathrm{C}(\mathrm{Ab}_1) + T^\mathrm{C}(\mathrm{Ab}_2) + \ldots + T^\mathrm{C}(\mathrm{Ab}_N) \\ &= \{\mathrm{Ab}_{11} + \ldots + \mathrm{Ab}_{1q_1}\} + \{\mathrm{Ab}_{21} + \ldots + \mathrm{Ab}_{2q_2}\} \\ &\quad + \cdots + \{\mathrm{Ab}_{N1} + \cdots + \mathrm{Ab}_{Nq_N}\} \end{aligned}$$

where $T^\mathrm{C}(\mathrm{Ab}_i) = \{\mathrm{Ab}_{i1} + \mathrm{Ab}_{i2} + \cdots + \mathrm{Ab}_{iq_i}\}$ indicates a proportional cloning in each elite pool, and $\mathrm{Ab}_{ij} = \mathrm{Ab}_i$, $i = 1, 2, \ldots, M$, $j = 1, 2 \ldots, q_i$. q_i is a self-adaptive parameter which determines the clone size of the antibody Ab_i. The representation $+$ is not the arithmetical operator, but only separates the antibodies here. $q_i = 0$ denotes that there is no cloning on antibody Ab_i.

In this study, the antibody with higher affinity is reproduced more times, viz, the antibody with higher affinity has a larger q_i. The values of q_i are calculated as

$$q_i = \lceil Q \times \frac{N - i}{N} \rceil$$

where Q is an expectant value of the size of the clone population and the ceil function $\lceil . \rceil$ returns a value representing the

smallest integer that is greater than or equal to its independent variable. After this process, all the antibodies in sub-population $\{\mathrm{Ab}_{i1}, \mathrm{Ab}_{i2}, \ldots, \mathrm{Ab}_{iq_i}\}$ are the result of the cloning on antibody Ab_i, and have the same property as Ab_i. In fact, cloning on antibody Ab_i is to make multiple identical copies of Ab_i. The aim is that the higher the affinity of an individual, the more times the individual will be reproduced. So there exist more repetitions to do search around the 'good' individuals and thus more chances to find better solutions.

4.7. Hypermutation: $T^{\mathbf{H}}$ A rapid accumulation of mutations is necessary for a fast maturation of the immune response. Hypermutation is the main mechanism. More often than not, a large proportion of the cloned population becomes dysfunctional or develops into harmful anti-self cells after the hypermutation. However, occasionally an effective change enables the offspring cell to bind better with the antigen, hence affinity is improved. Although the repertoire of antibodies in the immune system is limited; through affinity maturation, it is capable of evolving antibodies to successfully recognize and bind with known and unknown antigens, leading to their eventual elimination.

In this study, the hypermutation is realized by randomly choosing a gene position in the antibody and then changing it with a randomly generated integer in the domain of $[0, r-1]$. The lth gene of an antibody $\mathrm{Ab}_i = (a_{11}, \ldots, a_{nm}, b_{11}, \ldots, b_{nm}, c_1, \ldots, c_m)$ denotes the lth element in the antibody. The hypermutation T^{H} on the antibody Ab_i can be illustrated as follows.

$$(a_{11}, \ldots, a_{nm}, b_{11}, \ldots, \mathbf{b_{ij}}, \ldots, b_{nm}, c_1, \ldots, c_m)$$
$$\downarrow$$
$$(a_{11}, \ldots, a_{nm}, b_{11}, \ldots, \mathbf{b'_{ij}}, \ldots, b_{nm}, c_1, \ldots, c_m)$$

Without loss of generality, we suppose b_{ij} is the lth gene in the antibody Ab_i. The gene position l and mutated gene b'_{ij} are generated based on the following two equations, respectively.

$$l = \lfloor ran01() \cdot L \rfloor + 1$$
$$b'_{ij} = \lfloor ran01() \cdot r \rfloor$$

where the function $ran01()$ returns a random number uniformly generated in the range of $(0, 1)$. After the hypermutation, there are $\sum q_i$ mutated antibodies just as $(\mathrm{Ab}'_{11}, \mathrm{Ab}'_{12}, \ldots, \mathrm{Ab}'_{1q_1}; \ldots; \mathrm{Ab}'_{Nq_1}, \mathrm{Ab}'_{Nq_2}, \ldots, \mathrm{Ab}'_{Nq_N})$.

4.8. Re-selection: $T^{\mathbf{RS}}$ The fittest individual B_i $(i = 1, 2, \ldots, N)$ in each elite pool from among its mutated clones is determined by:

$$B_i = \mathrm{Ab}'_{ij}, \quad j = \arg\max_{j=1,\ldots,q_i}\{A(\mathrm{Ab}'_{ij})\}$$

Then, the parent antibody Ab_i $(i = 1, 2, \ldots, N)$ in each elite pool is updated with the fittest individual of the clones and the probability $P(\mathrm{Ab}_i \to B_i)$ is according to the role in the following.

$$P = \begin{cases} 1 & A(\mathrm{Ab}_i) < A(B_i) \\ 0 & A(\mathrm{Ab}_i) \geq A(B_i) \end{cases}$$

4.9. Chaotic new antibodies introduction: $T^{\mathbf{R}}$ In order to maintain the diversity of the population and prevent the search from being trapped in local optima, a portion of new antibodies are introduced to replace the worst c antibodies in the elite pools every generation. That is to say, the remaining antibodies after re-selection process $\{\mathrm{Ab}_{N-c+1}, \mathrm{Ab}_{N-c+2}, \ldots, \mathrm{Ab}_N\}$ are replaced by new generated antibodies $\{\mathrm{Ab}''_1, \mathrm{Ab}''_2, \ldots, \mathrm{Ab}''_c\}$. Moreover, these new antibodies are also constructed based on the chaotic system in (9) and this can be realized by setting $s = M + (k-1)c + 1, M + (k-1)c + 2, \ldots, M + kc$, respectively. And k $(k = 1, 2, \ldots)$ denotes the generation number of the algorithm.

4.10. General computational process Generally speaking, the whole process of CCSA is illustrated in Fig. 2. The termination criterion is set to be the condition that when k reaches a pre-specified large value $k_{\max}$.

Initially, each antibody is distributed in the ergodic space based on the chaotic system. Taking full advantages of ergodic and stochastic properties of chaotic variables, the parallel search mechanism of CSA is performed. The higher the affinity of an antibody is, the more clones it produces (see Fig. 3). In Fig. 3, the high-affinity antibody Ab_1 has more clones undergoing the hypermutation and thus has more chances to find the optimal solution. On the contrary, the low affinity antibody Ab_2 has less clones. As a result, the computational complexity can be reduced to some extent. Furthermore, the lowest affinity antibody is eliminated and replaced by a new antibody generated by chaos system to carry out chaotic search in the whole solution space. Both Ab_1 and Ab_2 are utilized to perform local search in the high-affinity areas of the solution space while the new introduced antibody replacing Ab_3 is employed to search in the whole solution space and maintain the population diversity to jump out of the local optima.

The characteristics (or contributions) of the proposed approach can be described as follows. First and foremost, the applicability of the clonal selection-based algorithms in synthesizing MVL functions are demonstrated. Besides, chaotic dynamics are embedded into the CSA. Both the antibody initialization and introduction of new antibodies are generated based on chaotic systems. Taking full

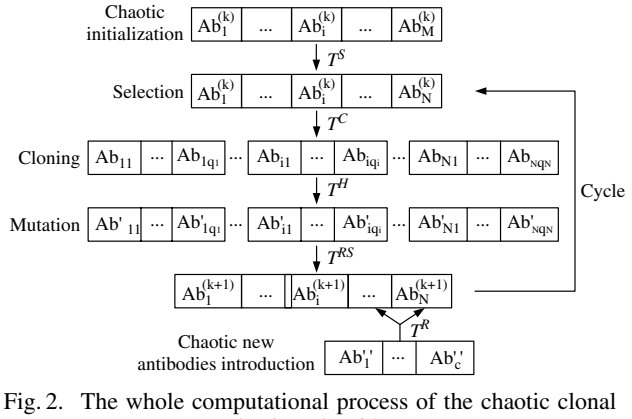


Fig. 2. The whole computational process of the chaotic clonal selection algorithm

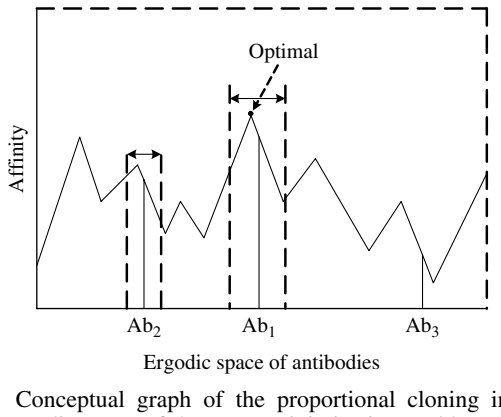


Fig. 3. Conceptual graph of the proportional cloning in the ergodic space of the MVL minimization problem

advantage of the ergodic and stochastic properties of chaotic variables, the global search ability and diversity of the population are improved. Furthermore, an adaptive length strategy of antibodies is proposed to reduce the computational complexity of the algorithm. Last but not least, an improved affinity function, in which both the correctness and the optimality of solutions are taken into consideration, enables the algorithm to find higher affinity antibodies with less product terms in the minimized MVL function.

5. Experiments and Discussions

In this paper, we applied the proposed CCSA to a series of randomly generated MVL functions as an example. All the simulations were implemented in C++ on a personal computer (Pentium 4 2.8 GHz). The characteristics of CCSA were analyzed from the following four aspects. First, the parameter sensitivity of CCSA was discussed. Especially, the two length strategies of antibody were compared. Then, compared with the error function utilized in the BP [10] and LS methods [11,12], the proposed affinity function that enabled the search to find minimized MVL functions with less product terms was verified. Moreover, the effect of the chaotic system taking on the CSA was also demonstrated. Finally, the performance of CCSA was verified by comparing with other traditional methodologies.

5.1. Parameter sensitivity analysis The performance of the CCSA was sensitive to the control parameter choices. In order to evaluate the sensitivity of the user-defined parameters in CCSA, after limited simulation experimentation, the parameters listed in Table II were adopted. All the algorithms with different parameters were separately tested on 20 two-variable four-valued MVL functions which were randomly generated. In this simulation, the initial average number of non-zero minterms of those functions was 12.15. Other simulation results are shown in Table III. The results that we recorded for each user-defined parameter were the value of the parameter, the correctness (E_c), the optimality (E_o), the affinity ($A(.)$) of the final antibody, the number of generations (G) when the correctness reached zero, and the computational time. All the values in Table III are the average results of the 20 trails.

As seen from Table III, we can easily find that the parameters k_{max}, M, N, Q, c and m possessed obvious features. For the maximum number of generation k_{max}, it can be easily concluded that the bigger the value of k_{max}, the better solution can be acquired while requiring more computational time. A trade-off scheme was setting $k_{max} = 1000$. M was the number of initial candidate solutions, and it just utilized once to produce M chaotic solutions. The computation times of different values of M were almost the same. From Table III, we can obtain that the accretion of M can rebound to the searching ability. Larger value of M indicated larger search space that the algorithm can explore. As a result, a sufficient large value of $M = 100$ was adopted. As to the parameter N which decided the antibodies whether enter into the cycle, we affirmed that the bigger value of N, the better solution can be acquired. The similar observation can also be found based on Q. Since N and Q determined the computational complexity of each generation, lower values of them significantly lowered the computational time. The optimal solution was observed when double the number of dimensions of the problem was taken as the number of them. As the current problem is $2^4 = 16$ dimensional, 32 antibodies and 32 clonal size were used. The parameter c increased the diversity of the solution through introducing new chaotic antibodies. No introduction of new antibodies ($c = 0$) might made the search easily trapped into the local optimal solutions. However, a too frequent introduction of new antibodies ($c = 5$) caused the algorithm to discard the improved solutions quickly and cannot acquire a satisfying solution. Therefore, $c = 2$ was adopted in this study.

As for the two different length strategies of antibodies, we found that better solutions can be obtained and less computational time was required when m utilized the AL strategy.

Table II. Parameters used in the simulation

Initial population number	$M = 100$
Elite pools	$N = 32$
Expectant clone population	$Q = 32$
Number of new introduced antibodies	$c = 2$
Initial number of product terms	m
Maximum number of generation	$k_{max} = 1000$

Note: m is determined according to the AL strategy in (6), i.e. m varies with different functions.

Table III. Parameter sensitivity (two-variable four-valued functions)

	Value	E_c	E_o	$A(.)$	G	T
k_{max}	100	1.35	5.85	0.78	93.6	0.95
	500	0.1	7.0	2.14	237.3	4.68
	1000	0	7.0	2.31	264.5	9.33
	2000	0	7.0	2.31	266.6	18.60
M	32	0.05	7.05	2.22	293.0	9.31
	50	0	7.15	2.26	263.95	9.32
	100	0	7.0	2.31	264.5	9.33
	200	0	7.0	2.31	251.95	9.35
N	16	0.1	7.1	2.12	435.7	5.58
	32	0	7.0	2.31	264.5	9.33
	48	0	7.0	2.31	274.85	13.12
	64	0	7.0	2.31	136.05	17.36
Q	16	0.05	7.0	2.21	360.4	5.42
	32	0	7.0	2.31	264.5	9.33
	48	0	7.0	2.31	207.2	14.12
	64	0	7.0	2.31	202.9	18.66
c	0	0.05	7.05	2.22	335.4	9.31
	1	0	7.0	2.31	252.8	9.32
	2	0	7.0	2.31	264.5	9.33
	5	0.05	7.05	2.19	292.5	9.45
m	Static	0	7.1	2.29	283.1	11.23
	Adaptive	0	7.0	2.31	264.5	9.33

5.2. Effect of the proposed affinity function The traditional error function defined in the references [11,12] was showed in the following.

$$E = E_c = \sum_{p}^{P} (O_p - T_p)^2$$

It was a part of the error function in (7). The disadvantages of the traditional error function were that there was no information incorporated in the error function and once the network outputs hit the target truth table successfully (i.e., $E_c = 0$), search would stop regardless how many product terms in final MVL function.

In practice, the objective of the MVL function design was not only the degree of the correctness which exposed how the system outputs and the target values in the truth table matched with each other but also the amount of the product terms in the final MVL function. Therefore, in this study both of them were taken into consideration.

In (7), E_c was designed to be the most important factor and E_o to be the second one. In order to realize this primary and secondary

relationship between E_c ($\in (0, (r-1)^2 r^n]$) and E_o ($\in (0, r^n]$), we set $\alpha = 1/r^n$ to make αE_o belong to the range of $(0, 1]$ in our approach. When $E_c \geq 1$ the landscape of the error function E was almost dominated by the correctness function E_c, while when $E_c < 1$, the optimality function E_o would take the main effect.

In this section, the effect of the proposed error function (or affinity function) was observed by analyzing three typical final solutions for the given MVL truth table in Table I as an example. The solutions were F_2 in Eq. (3), F_3 in Eq. (10) and F_4 in Eq. (11), respectively.

$$\begin{aligned} F_3(x_1, x_2) = {} & 3 \wedge x_1(0,1) \wedge x_2(2,2) \vee 1 \wedge x_1(1,1) \wedge x_2(2,3) \\ & \vee 1 \wedge x_1(0,1) \wedge x_2(1,2) \vee 2 \wedge x_1(3,3) \wedge x_2(3,3) \\ & \vee 1 \wedge x_1(0,0) \wedge x_2(0,2) \vee 3 \wedge x_1(0,2) \wedge x_2(2,2) \\ & \vee 1 \wedge x_1(3,3) \end{aligned} \tag{10}$$

$$\begin{aligned} F_4(x_1, x_2) = {} & 1 \wedge x_1(0,0) \wedge x_2(0,2) \vee 1 \wedge x_1(1,1) \wedge x_2(1,3) \\ & \vee 1 \wedge x_1(3,3) \vee 3 \wedge x_1(0,2) \wedge x_2(2,2) \end{aligned} \tag{11}$$

The properties of the three solutions can be summarized as follows.

(I) In (Eq. (3)), $E_c = 0$, $E_o = 5$, $E(F_2) = 5/16 = 0.3125$, $A(F_2) = 3.2$ and F_2 was a global optimal solution;

(II) In Eq. (10), $E_c = 0$, $E_o = 7$, $E(F_3) = 7/16 = 0.4375$ and $A(F_3) \approx 2.286$ and F_3 was a local optimal solution;

(III) In Eq. (11), $E_c = 1$, $E_o = 4$, $E(F_4) = 1 + 4/16 = 1.25$ and $A(F_4) = 0.8$ and F_4 was a local optimal solution;

Compared with F_4, both F_2 and F_3 were more attractive areas during searching since they had higher affinities. It was because that the correctnesses (E_c) of both F_2 and F_3 were better than that of F_4. Nevertheless, as the correctness of F_3 was zero, the search would stop by using the traditional error function. It was obvious that the minimized function F_3 was not so satisfactory because it still had seven product terms. This problem would worsen when dealing with actual large MVL circuits and systems. Fortunately, by utilizing the modified error function described above, F_2 and F_3 both of which could meet the target truth table successfully (i.e., $E_c = 0$) were separated with different affinities. In this condition, F_2 would be the only global optimal solution having the highest affinity.

Furthermore, a typical searching process based on a randomly generated MVL function is depicted in Fig. 5.2.. In Fig. 5.2., the horizontal axis indicated the generation of the algorithm whereas the vertical axis denoted the values of E_c, E_o and $A(.)$, respectively. Initially, there were 13 elements of the obtained function different from the target truth table ($E_c = 13$). Then at the 507th generation, the search found a solution hitting the truth table successfully. If using the traditional error function, the search would stop. However, at the 842th generation, a higher affinity antibody with less number of product terms was found.

Generally speaking, during all the functions with the same correctness (E_c), the less amount of the product terms (E_o) it had, the function would be synthesized more possibly after searching. As a result, less average number of product terms was required to realize a given MVL function by using the proposed affinity function.

5.3. Effect of the chaotic system taking on the CSA

In order to see how the chaotic system works on the CSA, comparison about the population diversity and the searching performance between CSA and the combined algorithm (CCSA) were observed.

A *diversity* index for the population of the solutions [40] is defined in the following. First, we define the *distance* between two solutions A and B as:

$$\mathrm{dis}(A, B) = \begin{cases} 0, & E_o(A, B) = 0 \\ \frac{E_o(A,B) - f(A,B)}{E_o(A,B)}, & \text{otherwise} \end{cases}$$

where $E_o(A, B) = \min\{E_o(A), E_o(B)\}$ indicates the minimum value of the product terms between A and B, whereas the function $f(.)$ returns the number of the common product terms between two solutions. For instance, $\mathrm{dis}(F_3, F_4) = 0.75$. In a generation, if the fittest solution is noted as A_0 and the others are noted as A_i ($i = 1, 2, \ldots, N-1$), the *diversity* of the population can be defined as:

$$\mathrm{DIV} = \frac{\sum_{i=1}^{N-1} \mathrm{dis}(A_0, A_i)}{N-1}$$

where DIV denotes the diversity of the population. It belongs to [0,1], and its main property is that the larger value of DIV, the better the diversity of the population.

We depicted the diversity versus the generation and their corresponding searching performances between CSA and CCSA in Figs 5 and 6, respectively. The difference between CSA and CCSA is the chaotic system. The initialization and new antibody introduction processes in CSA are based on random mechanisms, i.e. the candidate solutions are generated from uniform random numbers in [0,1]; while in CCSA they are based on the chaotic system. The comparison in Fig. 5 made it evident that CCSA had better diversity of the population and therefore stronger the ability of finding

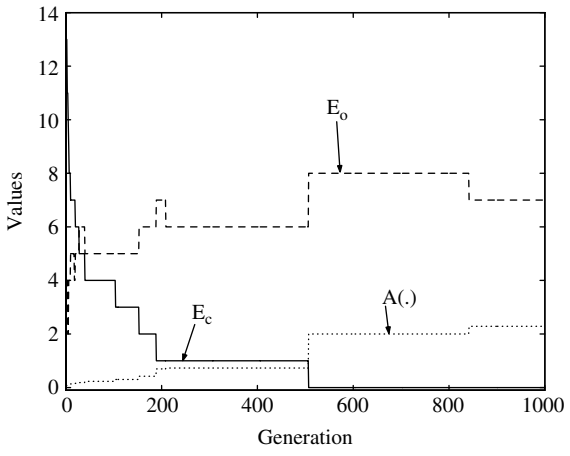


Fig. 4. A typical searching process of CCSA

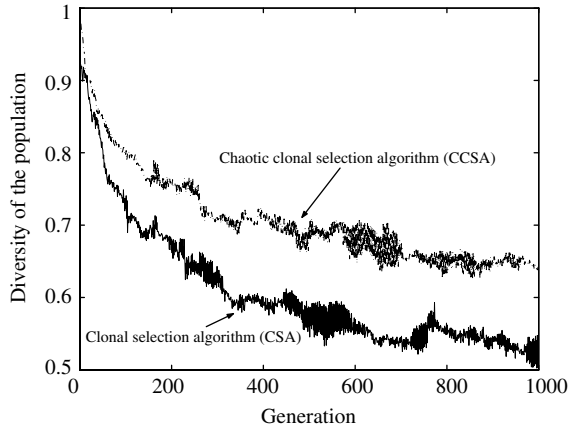


Fig. 5. Diversity of the population versus the generation between CSA and CCSA

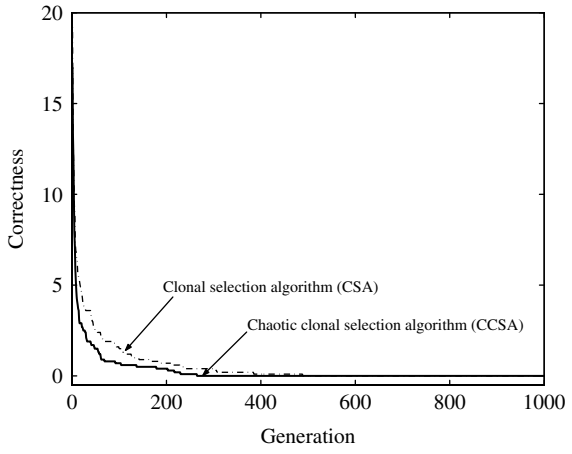


Fig. 6. Comparison of the searching performance between CSA and CCSA

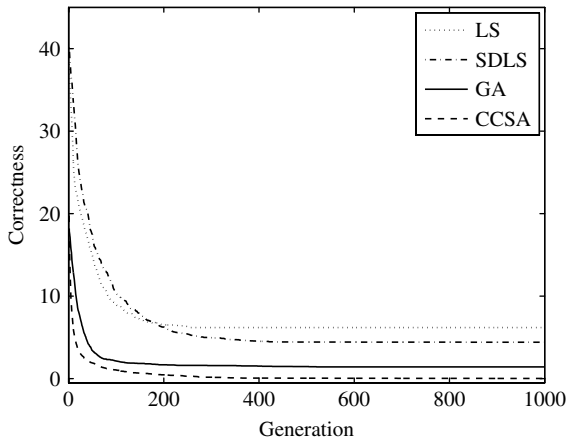


Fig. 7. Comparison of the searching performance during LS, SDLS, GA and CCSA algorithms

Table IV. Experimental results during LS, SDLS, GA and CCSA algorithms

		$MVL_{2,4}$	$MVL_{4,4}$	$MVL_{2,16}$
LS	Best E_c	1	38	417
	Average E_c	6.19	46.32	600.98
	Average E_o	—	—	—
	T(s)	3.05	135.89	89.58
SDLS	Best E_c	0 (9)	30	355
	Average E_c	4.41	45.76	593.54
	Average E_o	7.56	—	—
	T(s)	3.25	155.62	125.05
GA	Best E_c	0 (78)	9	118
	Average E_c	1.42	18.12	245.50
	Average E_o	7.17	—	—
	T(s)	16.87	577.20	385.56
CCSA	Best E_c	0 (196)	3	25
	Average E_c	0.02	12.50	80.65
	Average E_o	6.95	—	—
	T(s)	9.33	331.28	212.01

The symbol '—' means there is no value recorded for the result.

new product terms for an MVL function. On the other hand, from Fig. 6 we also found that the searching performance (convergence speed) of CCSA was better (faster) than that of CSA. As a result, we can say that the chaotic system enabled the clonal selection algorithm to better maintain the diversity of the population and improve the searching performance.

5.4. Comparison with other traditional methodologies Finally, we compared the experimental results of the four algorithms involving LS [11], SDLS [12], GA [15] and CCSA. In order to reduce the stochastic effect of the algorithms and make statistic comparisons, all the results were averaged over 20 randomly generated MVL functions and each function was run ten replications. Table IV summarized the results based on three groups of MVL functions.

In Table IV, the best and average correctness, the average number of product terms of the final solution and the computational time for each algorithm were recorded, whereas the values listed in the bracket were the times, during the 200 trails, that the correctness of the algorithm reached zero. It should be noted that the average number of product terms (optimality) was only recorded on condition that the correctness reached zero. From Table IV, we observed that, for the relatively small problems $MVL_{2,4}$, CCSA can learn the target MVL truth table successfully with a only exception of four times. For large MVL functions ($MVL_{4,4}$ and $MVL_{2,16}$), CCSA was able to perform near-optimal solutions. In general, CCSA was able to perform better solutions than the other algorithms within a reasonable computational time. The comparison results during the average E_o further confirmed the importance of the proposed affinity function.

Furthermore, in order to see how the algorithms performed solutions, we took $MVL_{2,4}$ as an example for detailed analysis of the searching performance. Similar analysis can be done for other functions. In Fig. 7, we plot the best correctness in a generation for each algorithm as a function of the number of generations. In the initial stage of searching, GA and CCSA can perform better solutions because they had a population of candidate solutions, whereas LS and SDLS only manipulated a single solution. In addition, when the difference between GA and CCSA is concerned, the two algorithms differ from the view point of inspiration, vocabulary and sequence of steps [41]. In particular, in GA the population is evolved using sexual crossover and mutation, whereas in clonal selection process the proportional cloning (T^C) operator is asexual and each child antibody produced by a cell is exact copy of its parent. It is evident that this operator can map a problem in a low dimension space to a high one, then project the results to the original space after solving, and therefore the problem can be solved more efficiently [42]. From Fig. 7, it can be easily found that CCSA outperformed the other algorithms in terms of the solution quality and convergence speed.

6. Conclusions

In this paper, a CCSA was proposed to synthesize MVL functions. On the basis of the clonal selection algorithm, chaotic dynamics were utilized to improve the global search ability of the algorithm and maintain the diversity of the population. Furthermore, two other improvements involving the affinity function and the AL strategy of antibodies enabled the algorithm to find better solutions and reduce the computational complexity, respectively. The performance of the proposed algorithm was evaluated by synthesizing a number of randomly generated multiple-valued functions and comparing the results of those of several traditional algorithms. The simulation results indicated that the proposed algorithm can synthesize the MVL function effectively and efficiently.

In future, we plan to combine the technology-dependent approach with the proposed algorithm to construct a hybrid system for synthesizing the MVL functions.

References

(1) Jain AK, Bolton RJ, Abd-El-Barr M. Cmos multiple-valued logic design. part I: circuit implementation. *IEEE Transactions on Circuits and Systems* 1993; **40(8)**:503–514.

(2) Jain AK, Bolton RJ, and Abd-El-Barr M. Cmos multiple-valued logic design. part ii: function realization. *IEEE Transactions on Circuits and Systems* 1993; **40(8)**:515–522.

(3) Brayton R, Hachtel G, Mcmullen C, Vincentelli AS. *Logic Minimization Algorithms for VLSI Synthesis*. Springer-Verlag; 1986.

(4) Mccluskey EJ. Minimization of boolean functions. *The Bell System Technical Journal* 1956; **35:** 1417–1444.

(5) Pomper G, Armstrong JR. Representation of multivalued functions using the direct cover method. *IEEE Transactions on Computers* 1981; **C–30(9)**:674–679.

(6) Dueck GW, Miller DM. A direct cover MVL minimization using the truncated sum. *Proceedings of the 17th IEEE International Symposium on Multiple-Valued Logic*, 1987, 221–226.

(7) Dueck GW. Direct cover MVL minimization with cost-tables. *Proceedings of the 22nd IEEE International Symposium on Multiple-Valued Logic*, 1992, 58–65.

(8) Watanabe T, Matsumoto M. Layered MVL neural networks capable of recognizing translated characters. *Proceedings of the 22th IEEE International Symposium on Multiple-valued Logic*, 1992, 88–95.

(9) Hata Y, Hozumi T, Yamato K. Gate model networks for minimization of multiple-valued logic functions. *Proceedings of the 23th IEEE International Symposium. on Multiple-valued Logic*, 1993, 29–34.

(10) Tang Z, Cao Q, Ishizuka O. A learning multiple-valued logic network: algorithm and applications. *IEEE Transactions on Computers* 1998; **47(2)**:247–250.

(11) Cao Q, Tang Z, Wang R, Wang W. Local search based learning method for multiple-valued logic networks. *IEICE Transactions on Fundamentals* 2003; **E86-A(7)**:1876–1884.

(12) Cao Q, Gao SC, Zhang J, Tang Z, and Kimura H. A stochastic dynamic local search method for learning multiple-valued logic networks. *IEICE Transacations on Fundamentals* 2007; **E90-A**:1085–1092.

(13) Wang W, Moraga C. Design of multivalued circuits using genetic algorithms. *Proceedings of the 26th IEEE International Symposium on Multiple-valued Logic*, 1996, 216–221.

(14) Hata Y, Hayase K, Hozumi T. Multiple-valued logic minimization by genetic algorithms. *Proceedings of the 27th IEEE International Symposium on Multiple-Valued Logic*, 1997, 97–102.

(15) Sarif B, Abd-El-Barr M. Synthesis of MVL functions—part i: The genetic algorithm approach. *International Conference on Microelectronics, ICM' 06*, 2006, 154–157.

(16) Tirumalai P, Butler J. Analysis of minimization algorithms for multiple-valued programmable logic arrays. *Proceedings of the 18th IEEE International Symposium on Multiple-Valued Logic*, 1988, 226–236.

(17) Tang Z, Ishizuka O, and Tanno K. A learning multiple-valued logic network that can explain reasoning. *Transactions of the Institute of Electrical Engineers, Japan* 1999; **119-C(8)**:970–978.

(18) Goldberg DE. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley: Boston, MA, USA; 1989.

(19) de Castro L, Zuben FJV. Learning and optimization using clonal selection principle. *IEEE Transactions on Evolutionary Computation* 2002; **6(3)**:239–251.

(20) Zhang L, Zhong Y, Li P. Application of artificial immune systems in remote sensing image classification. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2004, 397–401.

(21) Dong W, Shi G, and Zhang L. Immune memory clonal selection algorithms for designing stack filters. *Neurocomputing* 2007; **70**:777–784.

(22) Ong ZX, Tay JC, Kwoh CK. Applying the clonal selection principle to find flexible job-shop schedules. *Proceedings of the 4th International Conference on Artificial Immune Systems*, 2005, 442–455.

(23) Gao SC, Dai H, Yang G, Tang Z. A novel clonal selection algorithm and its application to traveling salesman problems. *IEICE Transactions on Fundamentals* 2007; **E90-A(10)**:2318–2325.

(24) Gao SC, Tang Z, Dai H, and Zhang J. An improved clonal algorithm and its application to traveling salesman problems. *IEICE Transactions on Fundamentals* 2007; **E90-A(12)**:2930–2938.

(25) Coello N, Cortes C. Multiobjective optimization using ideas from the clonal selection principle. *Genetic and Evolutionary Computation Conference, GECCO*, 2003, 158–170.

(26) Tan KC, Goh CK, Mamun AA, Ei EZ. An evolutionary artificial immune system for multi-objective optimization. *European Journal of Operational Research* 2008; **187**:371–392.

(27) Hata Y, Kamiura N, Yamato K. Multiple-valued product-of-sums expression with truncated sum. *Proceedings of the 27th IEEE International Symposium on Multiple-Valued Logic*, 1997, 103–107.

(28) Burnet FM. *The Clonal Selection Theory of Acquired Immunity*. Cambridge Press: Cambridge, UK; 1959.

(29) Li B, Jiang WS. Optimizing complex function by chaos search. *Cybernetics and Systems* 1990; **64**:821–824.

(30) Canabarro AA, Gleria IM, Lyra ML. Periodic solutions and chaos in a non-linear model for the delayed cellular immune response. *Physica A* 2004; **342**:234–241.

(31) May R. Simple mathematical models with very complicated dynamics. *Nature* 1976; **261**:459–497.

(32) Yang L and Chen T. Application of chaos in genetic algorithm. *Communications in Theoretical Physics* 2002; **38**:168–172.

(33) Collet P, Eckmann JP. *Iterated Maps of the Interval as Dynamical Systems*. Birkhauser: Boston; 1980.

(34) Lorenz EN. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* 1963; **20**:130–141.

(35) Chen G, Ueta T. Yet another chaotic attractor. *International Journal of Bifurcations and Chaos* 1999; **9**:1465–1466.

(36) Lű J, Chen G. A new chaotic attractor coined. *International Journal of Bifurcations and Chaos* 2002; **12(3)**:659–661.

(37) Ji MJ, Tang HW. Application of chaos in simulated annealing. *Chaos, Solitons and Fractals* 2004; **21**:933–941.

(38) Liu B, Wang L, Jin YH, Tang F, Huang DX. Improved particle swarm optimization combined with chaos. *Chaos, Solitons and Fractals* 2005; **25**:1261–1271.

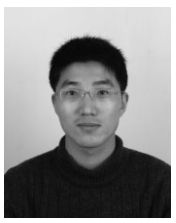
(39) Allen CM, Givone DD. A minimization technique for multiple-valued logic systems. *IEEE Transactions on Computers* 1968; **C-17(2)**:182–184.

(40) Maekawa K, Mori N, Kita H, Nishikawa H. A genetic solution for the traveling salesman problem by means of a thermodynamical selection rule. *IEEE International Conference on Evolutionary Computation* 1996, 529–534.

(41) de Castro L, Immune, swarm, and evolutionary algorithms part 2: philosophical comparisons. *Proceedings of the 9th International Conference on Neural Information Processing*, 2002:1469–1473.

(42) Gao SC, Dai H, Zhang J, Tang Z. An expanded lateral interactive clonal selection algorithm and its application. *IEICE Transactions on Fundamentals* 2008; **E91–A(8)**:2223–2231.

Shangce Gao (Member) received a B.S. degree from Southeast University, Nanjing, China in 2005 and an M.S. degree from University of Toyama, Toyama, Japan in 2008. Now, he is working toward the D.E. degree at University of Toyama, Toyama, Japan. His main research interests are multiple-valued logic, artificial immune system and artificial neural networks.



Qiping Cao (Non-member) received the B.S. degree form Zhejiang University, Zhejiang, China, an M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China and a D.E. degree from Kanazawa University, Kanazawa, Japan in 1983, 1986 and 2005, respectively. From 1987 to 1989, she was an assistant professor in the Institute of Microelectronics at Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a research student in Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a senior engineer in Sanwa Newtech Inc., Japan. In 2000, she joined Tateyama Systems Institute, Japan. Her current research interests include multiple-valued logic, neural networks and optimizations.

Zhiqiang Zhang (Non-member) received his B.S. degree and M.S. degree from Shandong University, Jinan, Shandong, China, in 2002, 2005 and in Computer Science and Management Science, respectively. He is currently working for his Ph.D. degree at University of Toyama, Japan. His main research interests are neural networks and optimizations.

Zheng Tang (Non-member) received the B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tshinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an Instructor in the Institute of Microelectronics at Tshinhua University. From 1990 to 1999, he was an associate professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined University of Toyama, Toyama, Japan, where he is currently a professor in the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks and optimizations.