Structural design of the danger model immune algorithm

Qingyang Xu *, Song Wang, Caixia Zhang

School of Mechanical, Electrical & Information Engineering, Shandong University at Weihai, Shandong 264209, China

A R T I C L E   I N F O

Article history:
Received 15 May 2009
Received in revised form 31 March 2012
Accepted 5 April 2012
Available online 17 April 2012

Keywords:
Danger model
Immune algorithm
Danger signal
Danger area

A B S T R A C T

The traditional immune algorithm (IA) is based on a self–nonself biological immunity mechanism. Recently, a novel immune theory called the danger model theory has provided more suitable biological information for data handling compared with the self–nonself mechanism. According to the danger model theory and based on past experiences of the genetic and artificial IA, we present the Danger Model Immune Algorithm (DMIA) that differs from the traditional IA in terms of the self–nonself biological immunity mechanism. We define a danger area and a danger signal in DMIA. We use the selection, mutation, and specific danger operators to update the population. The algorithm can achieve complex problem optimization. Simulation studies demonstrate that DMIA exhibits a higher efficiency than traditional genetic algorithms and other algorithms when considering a number of complicated functions.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

An Artificial Immune System (AIS) exhibits a highly parallel, distributed, self-adapted, and self-constructed system based on the biological immune system [6]. AIS can learn, recognize, memorize, and extract various characteristics [27]. In 1986, Farmer, a Nobel Prize winning biologist and immunologist, wrote a paper on the application of an artificial immune system, where the relationship between the immune system and machine learning was discussed [15]. Research on AIS application has gained considerable attention since then. Medical researchers discovered that the “self–nonself” biological immunity mechanism cannot resolve a number of problems appropriately [29], which serves as the foundation of conventional AIS. For example, humans do not have a specific immune response to bacteria in food, in the same way that pregnant women do not exhibit immune response to their fetus [28]. However, the “self–nonself” model [4] suggests that humans should respond to these substances outside their body. Therefore, Matzinger [29] proposed the danger model theory in 1994. The danger model theory centers on the biological immune system that reacts to danger signals released during unnatural cell death [10,28–30]. The danger signals are used instead of self- and non-self discrimination because they provide a different method of “grounding” the immune response [2]. Currently, a large number of experts and scholars have studied the danger model theory [1–3,16,17,20–22,40]. Aickelin and Greensmith introduced the danger model theory into AIS in 2002 and proposed an algorithm based on the behavior of dendritic cells, wherein the algorithm is applied to solve intrusion–detection problems [3]. Furthermore, the immune algorithm is used in network security and other applications [1,2]. Secker et al. [21] introduced the danger model theory in web mining. Smith et al. [22] described the immune memory, emphasizing that it is an in compact distributing memory with association and robustness characteristics. Therefore, Hart and Ross [16] constructed a Self-Organizing SDM (SOSDM) based on the immune memory. Later, Hart and Ross [17] applied the danger theory to improve the SOSDM algorithm, which is suitable for data handling in a dynamic environment. Swimmer [25] described the mechanism
involved in the immune response against defense problems based on the danger theory. The research findings on the artificial
immune algorithm based on the danger model theory have mostly focused on the realization idea of algorithms and the
application in intrusion–detection problems, among others. However, a concrete description of the algorithm has not yet been
fully elucidated. Therefore, we propose the Danger Model Immune Algorithm (DMIA) according to the characteristics of the
danger model theory using the genetic algorithm (GA) \[9,15,24,36,41\] and the conventional artificial immune algorithms
\[8,19,31–34,42\]. The simulation results demonstrate the validity and advantage of this algorithm in terms of optimization.

2. Danger model theory

The biological immune system protects the body from infection \[26\]. However, the immune system does not respond to
nonself signals; instead, it responds to the danger signal according to the danger model theory \[11\]. The danger signal essen-
tially establishes a danger zone. The antibodies in the danger zone become activated or stimulated. However, activation of
the lymphocytes does not occur even if an antigen matches with an antibody outside the danger zone. According to the two-
signal model \[3\], antigen recognition is facilitated by two signals, particularly a recognition signal (signal 1) and a co-stim-
ulation signal (signal 2), both of which recognize the antigen as the real danger.

Fig. 1 describes the immune response mechanism involved in the danger model theory. The immune system commonly
functions at three levels, specifically in the external barriers (skin, mucus), innate immunity, and the acquired or adaptive
immune system. Infectious agents are recognized based on the characteristics of protein fragments. The B-cell takes up
and internalizes the antigen. The antigen subsequently undergoes different processes and becomes re-expressed. The B-cell
then sends signal 1 and waits for the help signal from the T-helper (Th) cell. The B-cell lives for approximately 24 h and dies
if the Th signal is not released. The Th cells are not always active like B cells. Thus, the antigen recognition signal is insuf-
icient to activate the Th cells completely. T-cells need a specific co-stimulatory signal from the antigen-presenting cell
(APC). The danger model theory states that the APC must initially be activated depending on the health of its neighboring
cells \[28\]. Healthy cells do not send danger signals to local APCs. Only the damage and death attributed to viruses or other
entities generate such distress signals. In other words, a cell sends the danger signal 0 to activate the local APC when a cell
dies in an unnatural way (cell stress or lytic cell death, contrary to programmed cell death or apoptosis) \[30\]. The APC sends
the co-stimulated signal 2 to the Th cell when it becomes activated. The Th cell becomes completely activated. Finally, the Th
cell sends the antigen recognition signal 2 to the B-cell to initiate adaptive immune responses. B-cells then produce antibod-
ies to kill the antigen. By contrast, the cells that die through a programmed cell death send a signal that stimulates a neigh-
bor phagocyte. However, the dead cells should not induce the expression of co-stimulation.

We believe that the danger model theory provides more suitable biological information for data handling than the
traditional self–nonself viewpoint, regardless of the debatable ideas presented in the theory \[2\]. AIS algorithm based on

![Fig. 1. Schematic diagram of the danger model theory chart.](image-url)
the self–nonself mechanism describes the process involved in the recognition of antigens by the antibodies. We must establish a perfect antibody set for antigen matching to increase recognition accuracy. This process requires a large number of matching calculations, which is unsuitable for practical applications. Hence, DMIA provides an easy way of solving this problem. The danger theory does not focus on the way AIS represents the data. Instead, the danger theory provides information on the data that AIS should represent and deal with. Dangerous data, i.e., interesting data, should be the focus of the danger theory. Antigens and antibodies that do not match or are located far from each other do not become stimulated. The process can reduce the operation time and enhance the efficiency of the algorithm. Fig. 2 shows the mechanism involved in DMIA optimization.

3. Danger model immune algorithm

The variables are defined as conventional AIS in DMIA. The antigen corresponds to the objective function, and different kinds of constraint are present during optimization, thereby resulting in the formation of an antibody. The affinity of antibodies refers to the matching degree between optimization results and the objective function [13]. In this paper, we use the algorithm to optimize complex functions for testing. Thus, considering the maximization optimization problem:

\[ f^* = \max \{ f(\text{de}(Ab)) : Ab \in I \}, \]

where \( Ab = b_1, b_2, \ldots, b_l \) is the encoding of antibody \( x \), \( I \) is the antibody set, and \( b_i \) is the genetic factor of an antibody \( Ab \). In this paper, we adopt binary encoding. We divide the antibody string into \( M \) segments for a multi-variable problem, where \( l_m \) refers to the length of each segment and the variable \( a_m \in [ul_m, uh_m], i = 1, 2, \ldots, M \). The binary decoding of the antibody is shown in the following equation:

\[ a_m = ul_m + \frac{uh_m - ul_m}{2^{l_m} - 1} \left( \sum_{j=1}^{l_m} b_j 2^{j-1} \right), \]

where \( ul_m \) and \( uh_m \) are the bounds of variable \( a_m \), \( l_m \) is the encoding length of variable \( a_m \), \( j \) is the \( j \)th bit of \( a_m \), and \( b_j \) is the binary value.

3.1. Danger signal

The danger signal plays a key role in the immune response in the danger model theory [12]. Thus, defining the danger signal is the initial problem. We can define the danger signal as an indication of user interest [5]. Therefore, we define the danger signal as follows:

**Definition 1.** Danger signals 0 and 1. The system sends danger signals 0 and 1 when the number of the antibody (having an affinity of antibodies larger than \( C_1 \)) is larger than \( L_1 \).

\[ \text{count}(\text{aff}_{abi, a bj} > C_1) > L_1, \quad i \neq j \]

where \( C_1 \) is the affinity limit value of antibodies with other antibodies, \( L_1 \) is the limit number, and \( \text{count} \ (x) \) is a count function.

The system always sends danger signals 0 and 1 at the beginning of application because of the variety of antibodies. The similarity among antibodies increases by several iterations. The antibodies move closer to the optimal individual gradually. The danger signal stops when the end requirements are met.

![Optimized mechanism of DMIA](image-url)
The following equation is used to measure the affinity between any pair of antibodies [18]:

\[
\text{aff}_{ab-ab}(\text{Ab}_i, \text{Ab}_j) = \sum_{l=1}^{I} \phi(\text{Ab}_{il}, \text{Ab}_{jl}), \quad 1 \leq i, j \leq N
\]  

(2)

\[
\phi(x, y) = \begin{cases} 
1, & x = y \\
0, & x \neq y 
\end{cases}
\]

(3)

where \( N \) is the population size, \( \text{Ab}_{il} \) is the \( t \) bit of the \( i \)th Ab, \( I \) is the encoding length of each Ab, \( \text{aff}_{ab-ab} \) is the affinity of the antibody, and \( \phi(x, y) \) is the measurement function.

Affinity between the antibody and the antigen can be calculated using Eq. (4) [35]:

\[
\text{aff}_{ab-ag} = \frac{F(\text{Ab}_i)}{\sum_{i=1}^{N} F(\text{Ab}_i)}, \quad 1 \leq i \leq N
\]

(4)

\[
F(\text{Ab}_i) = f(\text{Ab}_i) + \text{MAX}
\]

(5)

where \( f(x) \) is the object function, and MAX is a larger value (maximize problem) to ensure that \( F(\text{Ab}_i) \) is a positive real number. An antibody with a higher \( \text{aff}_{ab-ag} \) value performs better than other antibodies.

**Definition 2.** Danger signal 2. The system selects the optimal individual and sends danger signal 2 to confirm the danger after it sends the danger signal 0 and 1.

We define the optimal individual for optimization problem as Eq. (6):

\[
\text{Ab}^*_{op} = \{f(\text{Ab}^*) = \max(f(\text{Ab})), \text{Ab}, \text{Ab}^* \in I\}
\]

(6)

where \( \text{Ab}^*_{op} \) is the optimal antibody, \( f(x) \) is the object function, and \( \max() \) is the maximum function. \( \text{Ab} \) is the antibody, and \( \text{Ab}^* \) is the semi-optimum antibody.

**3.2. Danger area**

**Definition 3.** Danger area. The system defines the environment of the optimal antibody as the danger area, which is the hyper sphere with radius \( R \), after it confirms a danger.

The antibody can be denoted as \( x_i = (a_{i1}, a_{i2}, \ldots, a_{iM}) \); the distances between the central antibody \( x_c \) and the antibodies \( x_i \) are given in Eq. (7):

\[
\text{dist}(x_c, x_i) = \sqrt{(a_{i1} - a_{c1})^2 + (a_{i2} - a_{c2})^2 + \cdots + (a_{im} - a_{cm})^2 + \cdots (a_{iM} - a_{cM})^2}
\]

(7)

where \( 1 \leq m \leq M, 1 \leq i \leq N, a_{im} \) is one of the variables of antibody \( x_i \), \( a_{cm} \) is one of the variables of antibody \( x_c \), \( M \) is the variable number of each antibody string, \( N \) is the population size, \( \text{dist}(x_c, x_i) \) denotes the distance between the central antibody \( x_c \) and the antibody \( x_i \). The antibody is in the danger area if \( \text{dist}(x_c, x_i) \) is smaller than \( R \).

**Definition 4.** Danger operator. The operator that establishes a mapping between the antibodies and the danger area is defined as the danger operator.

According to the definition, we know that \( a_{im} \in [a_{op}^{m*} - R, a_{op}^{m*} + R] \); thus, the decoding of \( a_{im} \) is as follows:

\[
a_{im} = a_{op}^{m*} - R + \frac{2R}{2^{l_m} - 1} \left( \sum_{j=1}^{l_m} b_j 2^{j-1} \right)
\]

(8)

where \( R \) is the radius of the danger area, \( op^* \) indicates the optimal individual, \( l_m \) is the \( m \)th variable encoding the length of the optimal antibody, and \( b_j \) is a binary value of the \( j \)th bit.

**Definition 5.** Danger antibody. The antibodies in the danger area are referred to as danger antibodies.

\[
\text{AbD}(k) = [\text{Ab}_{D1}(k) \text{Ab}_{D2}(k) \ldots \text{Ab}_{Dq}(k) \ldots \text{Ab}_{DP}(k)]\text{, }\text{whereAbD is the danger antibody set, andP is the number of danger antibodies.}
\]

**Definition 6.** Safe antibody. The antibodies outside the danger area are defined as safe antibodies.

\[
\text{AbS}(k) = [\text{Ab}_{S1}(k) \text{Ab}_{S2}(k) \ldots \text{Ab}_{SQ}(k) \ldots \text{Ab}_{SQ}(k)]\text{, whereAbS is the danger antibody set, andQ refers to the number of safe antibodies. We defineQ = N - P, whereN is population size, andP is the number of danger antibodies.}
\]
3.3. Flow of DMIA

Intelligent computations based on the gene exhibit similar characteristics. The selection operator is generally the preferred operator used in updating the population; crossover, mutation, and other operators are optional. The crossover operator is the main operator for population updating in GA; the mutation operator is the main operator for the immune algorithm (IA) depending on the characteristics of the immune system [6]. Therefore, the selection operator, mutation operator, and another special danger operator are used in population updating. The flowchart or the basic framework of the algorithm is shown in Fig. 3. The antibody population is divided into two populations, specifically the danger antibody set and the safe antibody set, during optimization. The safe antibody set is then operated using the danger operator. Finally, the mutation and selection operators are used in updating the population.

The flow of DMIA is described as follows:

Step 1: Initialization. The max iteration, termination condition, mutation probability $P_m$, and other parameters are set. Iteration $k = 0$, and the initial population is generated as follows:
$$Ab(0) = [Ab_1(0) \ Ab_2(0) \cdots Ab_N(0)] \in \mathbb{R}^N,$$
where $N$ is the population size.

Step 2: Affinity calculations. The antibodies are decoded. The affinity of antibodies with other antibodies and the antibodies with antigens are calculated.

Step 3: Acquisition of danger signals 0 and 1. Optimization ends when the danger signal is no longer present. Otherwise, the program sends danger signals 0 and 1. We can also use the iteration steps as the termination condition [7]
$$count(\text{aff}_{ab} > C_1) > L_1$$
where $C_1$ refers to the affinity limit value of antibodies and other antibodies, $L_1$ refers to the limit number of count for danger signal sending, and $count(x)$ refers to the count function. The equation indicates that the program sends the danger signal if sufficient antibodies ($L_1$) having affinity larger than $C_1$ are present.

Step 4: Sending of danger signal 2. The program obtains the optimal individual $Ab_{OP}$, and establishes the danger area according to $Ab_{OP}(k)$. Antibody $Ab(k)$ can be denoted as $Ab(k) = (a_1(k), a_2(k), \ldots, a_M(k))$, where $Ab_{OP}(k)$ is the central antibody. The distance between the central antibody and another antibody can be calculated using the following equation:
$$\text{dist}(\text{Ab}_{OP}(k), \text{Ab}_i(k)) = \sqrt{(a_1 - a_{OP1})^2 + (a_2 - a_{OP2})^2 + \cdots + (a_M - a_{OPM})^2}$$
where $1 \leq m \leq M$, $1 \leq i \leq N$, $M$ is the variable number, and $N$ is the population size. The danger area is a hypersphere with the center $(a_{OP1}, a_{OP2}, \ldots, a_{OPM})$.

Step 5: The program divides the population into two parts according to the danger area, danger antibody $Ab_D(k)$, and safe antibody $Ab_S(k)$.

$Ab(k) = Ab_D(k) \cup Ab_S(k)$, where $Ab_D(k)$ is the danger antibody in $\text{dist}(Ab_{OP}(k), Ab_S(k)) \leq R, Ab_S(k)$ is the safe antibody in $\text{dist}(Ab_{OP}(k), Ab_S(k)) > R$, and $R$ is the radius of the danger area.

Step 6: Danger operator. The danger operator indicates that the safe antibody is mapped toward the danger area directly. The individual safe antibody is as follows:
\[
\begin{align*}
\text{\( A_{S1}(k) = [A_{S1}(k)A_{S2}(k) \ldots A_{S9}(k) \ldots A_{SQ}(k)]\)} \\
\text{\( A_{S9}(k) = [b_1 b_2 \ldots b_l] = [a_{sq1}(k), a_{sq2}(k), \ldots, a_{sqm}(k), \ldots, a_{sqM}(k)]\)}
\end{align*}
\]

The program decodes \( a_{sqm}(k) \) in \([u_{hm}, u_{hm}]\) at step 2. \( m = 1, 2, \ldots, M \) is the variable number; however, \( M \) lies outside the danger area and moves in the danger area after treatment by the danger operator. \( a \in [Dl, Dh] \), \( Dl \) and \( Dh \) are the bounds of the danger area and variable \( a \). According to the optimal antibody, we can obtain \( Dl \) and \( Dh \) using the following equation:

\[
\begin{align*}
\text{\( Dl, Dh \) = \{de(A_{p}(k)) - R \text{ \( de(A_{q}(k)) \) + \( R \) \} \in I\)} \\
\text{\( Dl = (Dl_1, Dl_2, \ldots, Dl_m, Dl_M)\)} \\
\text{\( Dh = (Dh_1, Dh_2, \ldots, Dh_m, Dh_M)\)}
\end{align*}
\]

where \( de(x) \) is a decoding function, and \( M \) is the variable number. The variables of the safe antibody are decoded in the danger area, as shown in the following equation:

\[
\text{\( a_{sqm} = DO(a_{sqm}) = Dl_{sqm} + \frac{Dh_{sqm} - Dl_{sqm}}{2^m - 1} \left( \sum_{j=1}^{2^m} b_j 2^{j-1} \right) \)}
\]

where \( DO(x) \) is the danger operator. All the safe antibodies are transformed into danger antibodies after the operation. Finally, the danger operator is mapped toward the variable domain via inversion of transformation; the safe antibodies are described as follows:

\[
\begin{align*}
\text{\( Ab_S'(k) = [b_1' b_2' \ldots b_l'] = [a_{sq1}'(k), a_{sqm}'(k), \ldots, a_{sqM}'(k)]\)}
\end{align*}
\]

If \( Ab_S'(k) = [Ab_{S1}'(k) \ Ab_{S2}'(k) \ldots \ Ab_{S9}'(k) \ldots \ Ab_{SQ}'(k)] \), then the antibody population becomes:

\[
\begin{align*}
\text{\( Ab'(k) = Ab_{SQ}'(k) \cup Ab'_S(k) = [Ab_{D1}'(k) \ldots Ab_{DP}'(k), Ab_{S1}'(k) \ldots Ab_{SQ}'(k)]\)}
\end{align*}
\]

**Step 7: Mutation and selection operation.** We use the conventional proportion selection operator \( SO \) and basic mutation operator \( MO \) [23] to update the population.

Mutation of individual \( Ab^i \)

\[
\text{\( Ab^j = \{ M0 \ Ab^i(k) \ Ab^j(k) = [b_1^j b_2^j \ldots b_l] \}, \quad 1 \leq i \leq l \)}
\]

where \( l \) is the encoding length of the antibody string.

\[
\begin{align*}
\text{\( Ab^j(k) = MO(\text{\( Ab^i(k) \)}) = [\text{\( MO(\text{\( Ab^i_1(k) \)) \ldots MO(\text{\( Ab^i_2(k) \)) \ldots MO(\text{\( Ab^i_M(k) \)) \]}\)} \\
\text{\( = [Ab^j_1(k) \ Ab^j_2(k) \ldots \ Ab^j_M(k) \]}\)}
\end{align*}
\]

\[
\text{\( Ab^p(k) = [b_1^p b_2^p \ldots b_l^p] \)}
\]

\[
\text{\( b_i^p = \{ b_i^j \oplus 1, \quad \text{if } PM_i < Pm \) \\
\text{\( b_i^j, \quad \text{if } PM_i \geq Pm \) \}}
\]

where \( \oplus \) is exclusive or operation. \( MO(x) \) is the mutation function, and \( PM_i \in [0, 1] \) is a random number that refers to the mutation probability of the \( i \)th bit. \( Pm \) is the setting value of mutation probability. Individual \( Ab^p_i \) is selected in the next generation according to probability \( PS_i \).

\[
\begin{align*}
\text{\( Ab_S'(k) \rightarrow SO \ Ab^p_S(k) \)} \\
\text{\( PS_i = \text{aff}^{i}_{ab-\text{ag}} \sum_{k=1}^{N} \text{aff}^k_{ab-\text{ag}} \}, \quad 1 \leq i \leq N \)}
\end{align*}
\]

\[
\begin{align*}
\text{\( Ab^w(k) = SO(\text{\( Ab^p(k) \)}) = [\text{\( Ab^w_1(k) \) \ Ab^w_2(k) \ldots \ Ab^w_M(k) \]}\)}
\end{align*}
\]

where \( SO(x) \) is a selection function that uses the conventional roulette wheel selection.

**Step 8:** \( Ab(k + 1) = Ab^w(k), k = k + 1, \) return to Step 2.

### 4. Simulation

The following complex functions are used to verify the validity of the algorithm.
Example 1.

\[ f_1(x, y) = 0.5 - \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001 \times (x^2 + y^2))^2} \]

where \( f_1 \) is Schaffer's function [38], wherein \( x, y \in [-5.12, 5.12] \) with an infinite maximum value, and the global maximum value 1 is at point (0,0). A circuit ridge surrounds the global maximum value. Hence, obtaining a local maximum value during optimization is easy. The function is used to test the global searching capability of the algorithm. Fig. 4 shows the function \( f_1 \).
The algorithm parameters include the following: Binary string is adopted to encode the antibody with the binary string length measuring 20. Each variable encoding length measures 10 in an antibody. The population size is 50, and the maximum number of iterations is 100. The algorithm degrades into a simplified genetic algorithm with the selection and mutation operators if the danger area spreads all over the variable domain. Thus, we can use this algorithm to compare DMIA with GA under the same conditions.

Different danger areas ($R = 3$ and $R = 5$) are used in testing the affinity of the danger area and in comparing the DMIA with GA under different mutation probabilities. Fig. 5 shows the average optimization process curve of $f_1$, where DMIA adopts different danger areas after the 100rd optimization. The variable domain of $f_1$ is $[-5, 12]$. The algorithm almost degrades into a GA if the radius of the danger area $R$ is 5. Fig. 5 shows that GA ($R = 5$) initially exhibits the same optimal individual as DMIA ($R = 3$). However, GA’s optimization capability is not very good, whereas DMIA ($R = 3$) has a better optimization characteristic. The optimization capability of GA is better when $P_m = 0.3$ (Fig. 5b); however, DMIA is superior to GA. The optimization value is 0.99 in 30 iteration steps in DMIA when $P_m = 0.03$. The optimization value is only 0.965 in GA when iteration steps are set at 100. The optimization value for DMIA is 0.99 in 10 iterations when $P_m = 0.3$, faster than the previous step. For GA at 50 iteration steps, the optimization value is 0.99. DMIA thus has a better global searching capability than GA.

Fig. 6 shows the optimal result after the 100-times optimization with different $P_m$ and danger areas. In Fig. 6a, $P_m$ is 0.03, the dotted line refers to $R = 3$, and the solid line represents $R = 5$. $R = 3$ (DMIA) shows a better optimization result than $R = 5$ (GA). In Fig. 6b, $P_m = 0.3$, indicating that the mutation probability is 10 times larger than the previous simulation. Both GA and DMIA have better optimization capability than other algorithms. The optimization result in GA is steadier than that in Fig. 6a. The mutation operator is the basic operator for the IA; thus, the algorithm with higher mutation probability exhibits better optimization capabilities.

Fig. 7 shows the contrast of optimization under the same danger area at different $P_m$ conditions. A higher $P_m$ value is useful for global searching, and the convergence speed is faster than that at smaller $P_m$ values.
Example 2.

\[ f_2(x, y) = \left( \frac{3}{0.05 + (x^2 + y^2)^2} \right)^2 + x^2 + y^2 \]

where \( x, y \in [-5, 12, 5, 12] \), \( f_2 \) is a local peak function, and the maximum value is 3600 at point (0, 0) [8]. This function can be used in determining the local searching capability of the algorithm. Fig. 8 shows \( f_2 \).

We adopt different danger areas and mutation probabilities to confirm the algorithm during the optimization.
The parameters of the algorithm are similar to those in Example 1. Fig. 9 shows the optimization result curves with distinct danger areas. The algorithm degrades into a GA if the radius of the danger area equals 5 when $x, y \in [-5.12, 5.12]$. Fig. 9 also shows that DMIA exhibits a better local searching capability than GA; the convergence speed of the DMIA is faster than GA when $Pm = 0.03$. Both DMIA and GA exhibit fast convergence when $Pm = 0.3$. DMIA can obtain the optimum value in 20 iterations, whereas GA can reach the optimum value in 30 iterations (Fig. 9b).

Fig. 9. Optimization processes using different parameters.

Fig. 10. Optimization of different danger areas after the 100-times optimization with $Pm = 0.03$. 

The parameters of the algorithm are similar to those in Example 1. Fig. 9 shows the optimization result curves with distinct danger areas. The algorithm degrades into a GA if the radius of the danger area equals 5 when $x, y \in [-5.12, 5.12]$. Fig. 9 also shows that DMIA exhibits a better local searching capability than GA; the convergence speed of the DMIA is faster than GA when $Pm = 0.03$. Both DMIA and GA exhibit fast convergence when $Pm = 0.3$. DMIA can obtain the optimum value in 20 iterations, whereas GA can reach the optimum value in 30 iterations (Fig. 9b).
Fig. 10 shows the final optimal result curve after the 100-times optimization with different $P_m$ values. The dotted line denotes $R = 3$ and $P_m = 0.03$. The solid line denotes $R = 5$. $R = 3$ produces a better optimization result than $R = 5$.

Fig. 11 shows the optimization process curve with the same danger area radius $R$ and different $P_m$. Figs. 11a and b show the optimization process in GA and DMIA, respectively. A higher mutation probability and suitable danger area are helpful in searching for the optimal antibody.

Fig. 12. Optimization result comparison curve of DMIA and GA.
The crossover operator is the basic operator in GA, whereas the mutation operator is the key operator in DMIA. Thus, we add a crossover operator into the algorithm to simulate GA completely when $R = 5$. In the GA, $R = 5$, $P_m = 0.03$, and $P_c = 0.7$; in DMIA, $R = 3$ and $P_m = 0.3$. The optimization stability of GA is worse than that of DMIA although GA uses a crossover operator in Fig. 12. The optimization result of GA is better than that shown in Fig. 10. However, GA remains insufficient compared with DMIA.

Example 3.

$$f_3(x, y) = -\left( x^2 + y^2 \right)^{0.25} \left( \sin^2 50 \times (x^2 + y^2)^{0.1} + 0.1 \right), \quad x, y \in [-5.12, 5.12]$$

Example 3 describes Schaffer’s F7 function with the maximum global value at 0. The F7 is a complicated function with a vibrating circuit ridge outside the maximum global value [39]. This function can verify the global and local optimization capability of the algorithm. Fig. 13 shows $f_3$.

The algorithm parameters include the following: The binary string length is 40, the population size is 100, the mutation probabilities are 0.04 and 0.4, and the maximum number of iterations is 100. We adopt different danger areas to test the algorithm. Fig. 14 shows the three-dimensional graphics of the optimization process with different danger areas. In Figs. 14a–c, the danger area is 3, and $P_m$ is 0.4. In Figs. 14 (d) to (h), the danger area is 5, and $P_m$ is 0.4. The sphere is the danger area and the black “*” is the antibody. At the beginning of the program, the antibodies spread throughout the entire variable domain. The danger area center refers to the semi-optimum antibody, and a danger area is then established according to this semi-optimum antibody. The antibodies are closer to the optimum antibody and climb to the peak of the “mountain” gradually for several iterations. Finally, we obtain the optimum result. In Figs. 14a–c, the algorithm is convergent in five iterations, whereas in Figs. 14d–h, the algorithm is convergent after 50 iterations. The suitable danger area has important effects on the convergence speed and precision of the result. Fig. 15 shows the optimization process curves, wherein no crossover operator is used. We add a crossover operator into the algorithm to simulate GA completely. Fig. 16 shows the optimization process curve of standard GA. The optimization result is worse than that shown in Fig. 15b, although the optimization result and convergence speed are better than that in Fig. 15a. The algorithm with higher $P_m$ has the best optimization performance.

Fig. 17 shows the optimization curves with different parameters, particularly danger area $= 3$ and $P_m = 0.4$, with a population size of 150. The optimization curve has a better capacity than the previous one. The population size is bigger; thus, the possibility of finding the optimal antibody in finite iterations is increased.

Example 4.

$$f_4(x, y) = \sum_{i=1}^{n} \left( 10 \cos(2\pi x_i) - x_i^2 \right), \quad x_i \in [-5.12, 5.12]$$

$f_4$ is a multimodal high-dimensional function. Numerous local maximum values in the searching space are present, which increase with the function dimension. Therefore, optimization is difficult to achieve [14, 27, 37]. The experiment is used to evaluate the performance of DMIA in high-dimensional function optimization. Fig. 18 shows $f_4$ in three dimensions ($n = 2$) to illustrate its complexity.

The algorithm parameters include the following: the string length is 100, with each binary variable string length measuring 10; the population size is 100; the mutation probabilities are 0.04, 0.4, 0.6, and 0.8; and the maximum number of iterations is 300.
The variable domain is $[-5.12, 5.12]$. The algorithm degrades into a simplified GA if the radius of the danger area is 5. Therefore, we can compare the performance of DMIA with the simplified GA.

![Three-dimensional graphics of the optimization process with different danger areas.](image)

**Fig. 14.** Three-dimensional graphics of the optimization process with different danger areas.
The size of the searching space and number of local minima increase with the dimension of the function; hence, a higher dimension makes the problem more difficult. Therefore, this experiment studies the performance of DMIA in 10-dimensional function optimization.

\[
f_{10}^{4}(x, y) = 350 + \sum_{i=1}^{10} \left( 10 \cos(2\pi x_i) - x_i^2 \right), \quad x_i \in [-5.12, 5.12]
\]

Fig. 19 shows the convergence result comparisons of \( f_{10}^{4} \) using DMIA with different \( P_m \), indicating that DMIA has better performance with higher \( P_m \).

Fig. 20 shows the crossover operator added to the simplified GA, the “*” line, “-” line, and the “+” line are simplified GA convergence results with different \( P_m \), and the “c” line is the standard GA (SGA). The figure shows that SGA has better performance than simplified GA.
Fig. 21 shows the convergence result comparison of DMIA and GA. The performance of DMIA differs from that of GA, wherein a boundary is observed in the figure. The DMIA has better performance, whereas GA has poor performance. The convergent speed of the DMIA is faster than that of GA; the DMIA optimization result is also better. In DMIA, the algorithm with higher $Pm$ has better performance.
Example 5. Comparison with other algorithms.

DMIA exhibits better performance than GA according to the descriptions above. We provide a comparison with other algorithms in high-dimensional function optimization. The same high-dimensional function is adopted as Example 4.

\[
f_{10}^{4}(x, y) = 350 + \sum_{i=1}^{10} (10 \cos(2\pi x_i) - x_i^2)
\]

where \(x_i \in [-5.12, 5.12], i = 1, 2, \ldots, 10\). A maximal value 450 exists at the point (0,0,0,0,0,0,0,0,0,0). Some parameters of DMIA are set according to other algorithms to compare with these other algorithms under the same conditions. The population size \(N\) is 180. String length is 240. \(P_m\) is 0.4, and \(R\) is 3.

The convergent speed of DMIA is faster than SGA (Fig. 22).

The performance of the results is comparable with King Crossover Based Elitist Genetic Algorithm (KEGA) [43], Adaptive Immune Algorithm (AIA) [45], artificial immune algorithm based on euclidean distance and king-crossover (DKBAIA) [46], and Modified Distance and King-crossover-based Artificial Immune Algorithm (MDKBAIA) [44]. Furthermore, Table 1 lists the optimization results using DMIA and other algorithms. Other algorithms can provide better solutions after 626 genera-

### Table 1
Comparison of optimization results using DMIA and other algorithms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max iteration</td>
<td>4000</td>
<td>4000</td>
<td>4000</td>
<td>4000</td>
<td>500</td>
</tr>
<tr>
<td>Optimal result</td>
<td>449.709632</td>
<td>441.915806</td>
<td>447.740688</td>
<td>449.276990</td>
<td>449.998486</td>
</tr>
<tr>
<td>Average iteration</td>
<td>626</td>
<td>31332</td>
<td>3441</td>
<td>840</td>
<td>424</td>
</tr>
</tbody>
</table>
sions. DMIA can yield the best solution within 424 generations compared with other algorithms. Therefore, DMIA can offer users better performance in dealing with complex optimization problems.

Thus, DMIA exhibits superior performance compared with other algorithms (GA, SGA, KEGA, AIA, DKBAIA, and MDKBAIA). DMIA is an effective method that can be easily used in solving real-world applications regardless of the complexity of the problem.

5. Conclusion

In this paper, we proposed a basic framework of DMIA, which differs from the conventional AIS algorithm based on the danger model theory. Danger signal and danger area were introduced into the algorithm. The algorithm is initialized by the danger signal, after which the antibodies are divided into two categories according to the danger area in the optimization. DMIA can improve the efficiency of the conventional AIS algorithm. In DMIA, we adopt the proportional selection, mutation, and danger operators to update the population, thereby resulting in good global searching capability and quick convergence. The algorithm establishes and focuses on the antibodies in the danger area during the optimization. We then used the operators to update the population. DMIA with higher mutation probability and suitable danger area has better optimization performance. Thus, we proposed a basic framework of DMIA, although further studies are still recommended; for instance, areas of interest include advanced mutation and clone operators that can be added to the algorithm to improve its performance.

Acknowledgements

The author thanks Prof. Shijie Zhou, Prof. Ning Wang, and Ms. Yuan Yuan for their kind help. This paper is supported by the Shandong Province Science and Technology Development Plan of China (Grant No. 2011GGE27053).

References


