Evol. Intel. (2011) 4:203–218
DOI 10.1007/s12065-011-0064-1

RESEARCH PAPER

Artificial immune optimization system solving constrained omni-optimization

Zhuhong Zhang

Received: 9 October 2010 / Revised: 10 September 2011 / Accepted: 22 September 2011 / Published online: 13 October 2011

Abstract This work investigates an artificial immune optimization system suitable for single and multi-objective constrained optimization. In this optimizer, an evaluation index, which can decide the importance of individual in the current population, is developed to accelerate population division; the niching-like proliferation scheme is introduced to strengthen the diversity of population. Thereafter, those diverse antibodies, with the help of immune evolution operations, evolve their structures along different directions. Theoretical results show that such optimization system is convergent with low computational complexity. Experimentally, one such optimizer is sufficiently examined by a suite of single and multi-objective test problems. Comparative experiments illustrate that the optimizer with some striking characteristics is a potentially alternative optimization tool for constrained omni-optimization.

Keywords Constrained omni-optimization · Artificial immune systems · Clonal selection · Convergence · Computational complexity

1 Introduction

With the development of evolutionary computation, some representative evolutionary algorithms only suitable for either single or multi-objective optimization problems have been proposed in the literature [1–6]. Nevertheless, in practice, because of different requirements and goals, many optimization problems, e.g., university timetable, engineering design, flow-shop scheduling, and so on, frequently change their type of optimization; in other words, they sometimes include only one objective within some time span but multiple conflicting objectives within another time span. This requires that a single search method could deal concurrently with single and multi-objective optimization so-called omni-optimization. Thus, some advanced omni-optimization techniques are desired when concurrently finding the optimal solution(s) and Pareto-optimal solutions for such kind of problem, but fewer achievements are reported in the literature.

Although lots of researchers still focus on exploring evolutionary algorithms for single or multi-objective optimization problems, the research on omni-optimization is in progress. Deb et al. [7] made an encouraging attempt to develop the concept of omni-optimization. In their work, an omni-optimizer, based on the well-known NSGA-II [3], was reported. In such optimizer, the classical crowding distance method included in NSGA-II was modified to calculate distances between candidates both in the design space and in the objective space. The purpose of such modification is to guarantee that the optimizer is effective for multi-global optimization problems. Especially, when being degenerated to solve single-objective problems, it is similar to the standard evolutionary algorithm $(\mu + \lambda)$-ES. The experimental results in the present work indicate that such optimizer is competitive for some multi-objective problems. Additionally, in order to deal with practical engineering design and simulation-based optimization problems, several researchers also investigated omni-optimization with discrete variables by developing improved genetic algorithms [8–10]. For instance, in Klanac and

Z. Zhang (✉)
Institute of System Science and Information Technology, College of Science, Guizhou University, Guiyang 550025, Guizhou, People's Republic of China
e-mail: sci.zhzhang@gzu.edu.cn

Jelovica [8], constraints were converted into additional objectives, and the weighting strategy assumed designing the scheme of fitness. The optimizer is based on vectorization and one conventional genetic algorithm. However, it remains unanswered how to decide the weights of objectives.

In the field of intelligent optimization, another outstanding computational paradigm, Artificial Immune Systems, has been exhaustively studied, especially, immune optimization. Many researchers made great contribution to sufficiently excavating the inherent metaphors of the immune system in order to solve multi-modal optimization problems, and accordingly, lots of immune genetic or immune optimization algorithms were developed [11–15]. These algorithms are designed specially to deal only with either single or multi-objective problems. As associated to the relation between bio-inspiration and existing immune-based optimization algorithms, one can easily know that the immune metaphors of density and vaccine are usually cited to improve evolutionary algorithms [16, 17], and also that three immune principles of negative selection, clonal selection and immune network are borrowed to explore immune optimization algorithms.

The negative selection principle and the concept of vaccine have been preliminarily taken into account for function optimization [18–20], especially constrained optimization. For instance, in Aragón [18], a T-cell model, based on the process of T-cell response, was presented in the context of constrained optimization. In this model, a dynamic tolerance factor was designed specially to identify the constraint-violation degree of an individual, while several mutation operators were developed to evolve different types of cells so that those better cells were found as fast as possible. In [20], the concept of vaccine was utilized skillfully to design an artificial immune system for multi-modal function optimization, in which, after the design space was divided into equal subspaces, the vaccine was randomly picked up in each subspace. The authors claimed that it was able to achieve the promising performance.

The clonal selection principle is an extremely useful bio-inspiration in artificial immune systems. A large number of immune characteristics and mechanisms such as learning, memory, diversity and affinity maturation have been widely cited to investigate immune optimization. The first optimization algorithm for multi-modal function optimization, clonal selection algorithm originally proposed by de Castro and Von Zuben [21], was originated from the principle. After such pioneering work, a series of immune-based optimization techniques for single or multi-objective problems were established [22, 23]. Especially, some original artificial immune models were reported [24–32], among which a few constraint-handling techniques had been successfully applied to constrained multi-objective problems [31, 32]. More details can be found in [26, 27]. In addition, more recent studies indicate that multi-objective immune optimization is a competitive and active research topic [33–37].

As associated to the clonal selection principle and immune network theory, de Castro and Von Zuben [38] suggested an artificial immune network (aiNet) solving data analysis and clustering. Thereafter, in order to handle multi-modal function optimization with static or dynamic environments, their group, together with Timmis, extended aiNet into two valuable versions (opt-aiNet and dopt-aiNet) with slight differences [39, 40]. Nevertheless, although immune optimization was investigated intensively, few achievements useful for omni-optimization have been found in the literature. Coelho and Von Zuben [41] suggested an improved artificial immune network (omni-aiNet) for such kind of optimization which extended the version of opt-aiNet [39]. omni-aiNet and opt-aiNet share some main mechanisms, but there are some essential differences. In omni-aiNet, one reported mechanism, Gene Duplication, is introduced to enhance the ability of evolution. The fundamental experiments in the present study hint that this algorithm is potential for single-objective problems with fewer constraints, but it also exposes some drawbacks when solving multi-objective problems.

In the present work, an artificial immune optimization system for omni-optimization with nonlinear constraints, simply written as omni-AIOS, is developed. It is not an extension version of the previous work, and especially, it differs from omni-optimizer and omni-aiNet [7, 41]. More details can be found in Sect. 4.3. In omni-AIOS, in order that the current population can be divided rapidly into subclasses with different levels, an evaluation index is first designed specially to decide the importance of individual in the population, which helps omni-AIOS enhance the speed of sorting. Second, after each subclass is required to eliminate redundant individuals by means of a suppression radius index, those survival individuals in each subclass proliferate their clones, where the size of clones for each individual is the number of individuals suppressed by it. Simulation illustrates that omni-AIOS is a competitive optimizer for omni-optimization with nonlinear constraints.

2 Preliminaries and basic theoretical results

2.1 Problem formulation and basic concepts

Consider the following constrained omni-optimization (**P**):

$$\begin{aligned}&\underset{x\in\Omega}{\mathrm{Min}}\,\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))\\ &\text{s.t.}, g_i(\mathbf{x}) \le 0,\ h_j(\mathbf{x}) = 0,\ 1 \le i \le I, 1 \le j \le J,\end{aligned}$$

with $M \geq 1$, where $f_i(\mathbf{x})$ is the i-th sub-objective function with $1 \leq i \leq M$, and $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ are the nonlinear constraint functions with $1 \leq j \leq I$ and $1 \leq k \leq J$; Ω is a bounded and closed domain in R^p. $\mathbf{x} \in \Omega$ is said to be a feasible solution if it satisfies the above constraints; otherwise, it is called an infeasible solution. For $\mathbf{x}, \mathbf{y} \in \Omega$, it is said that $\mathbf{x}$ dominates $\mathbf{y}$ in the objective space, simply written as $\mathbf{x} \prec_f \mathbf{y}$, if $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$ with $1 \leq i \leq M$, and there exists l such that $f_l(\mathbf{x}) < f_l(\mathbf{y})$ with $1 \leq l \leq M$; especially, when $M = 1$, $\mathbf{x} \prec_f \mathbf{y}$ is equivalent to $f(\mathbf{x}) < f(\mathbf{y})$. Notice that it is easy to prove that the relation of dominance between candidates is of the transitive property. Next, assume that $\Gamma(\mathbf{x})$ is a penalty-based violation function which measures the magnitude of violations of the constraints for candidate $\mathbf{x}$. For simplicity, define it as follows in this paper.

$$\Gamma(\mathbf{x}) = \sum_{i=1}^{I} \max\{g_i(\mathbf{x}), 0\} + \sum_{j=1}^{J} h_j^2(\mathbf{x}). \tag{1}$$

Here, if $\Gamma(\mathbf{x}) < \Gamma(\mathbf{y})$, candidate $\mathbf{x}$ is said to be better than candidate $\mathbf{y}$. Thereafter, the concept of constraint-dominance [42] is cited to compare two candidates for problem (**P**).

Definition 2.1 Let $\mathbf{x}, \mathbf{y} \in \Omega$, it is said that $\mathbf{x}$ constrained-dominates $\mathbf{y}$ (simply written as $\mathbf{x} \prec_c \mathbf{y}$), if one of the following conditions holds:

(a) $\mathbf{x}$ and $\mathbf{y}$ are feasible, and $\mathbf{x} \prec_f \mathbf{y}$;
(b) $\mathbf{x}$ is feasible but $\mathbf{y}$ is not;
(c) $\mathbf{x}$ and $\mathbf{y}$ are infeasible, and $\Gamma(\mathbf{x}) < \Gamma(\mathbf{y})$.

Based on the above definition, one can see that for arbitrary candidates $\mathbf{x}$ and $\mathbf{y}$ in Ω, only one is true among $\mathbf{x} \prec_c \mathbf{y}, \mathbf{y} \prec_c \mathbf{x}$ and $\mathbf{x}|_c\mathbf{y}$, where $\mathbf{x}|_c\mathbf{y}$ denotes $\mathbf{x} \not\prec_c \mathbf{y}$ and $\mathbf{y} \not\prec_c \mathbf{x}$.

Definition 2.2 $\mathbf{x} \in \Omega$ is called an optimal solution ($M = 1$) or a Pareto-optimal one ($M > 1$) if and only if there does not exist $\mathbf{y} \in \Omega$, s.t. $\mathbf{y} \prec_c \mathbf{x}$. Let X be a finite subset in Ω; $\mathbf{x} \in X$ is said to be a best or nondominated individual with respect to X only if there is not $\mathbf{y} \in X$, s.t. $\mathbf{y} \prec_c \mathbf{x}$.

2.2 Evaluation index and basic properties

One can easily see from Definition 2.1 that the relation of constraint-dominance $\prec_c$ is of the transitive property, i.e., if $\mathbf{x} \prec_c \mathbf{y}$ and $\mathbf{y} \prec_c \mathbf{z}$, then $\mathbf{x} \prec_c \mathbf{z}$. Further, let $m(\mathbf{x})$ stand for the set of remembers dominated by $\mathbf{x}$ in the finite subset X in the sense of constraint-dominance, and accordingly one can obtain $m(\mathbf{y}) \subset m(\mathbf{x})$ if $\mathbf{x} \prec_c \mathbf{y}$. Also, let Ω' represent the set of feasible candidates in X. Based on such preliminaries, define a bivariate function $B_{etter}(\mathbf{x}, \mathbf{y})$,

$$B_{etter}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \mathbf{x} \in \Omega', \mathbf{x} \prec_c \mathbf{y}, \\ 0 & \mathbf{x} \in \Omega', \mathbf{x} \not\prec_c \mathbf{y} \vee \mathbf{x} \notin \Omega', \mathbf{x} \prec_c \mathbf{y}, \\ -1 & \mathbf{x} \notin \Omega', \mathbf{x} \not\prec_c \mathbf{y}. \end{cases} \tag{2}$$

This way, an evaluation index function $L_{evel}(.) : X \to R$, which decides the importance of individual in X, is given by

$$L_{evel}(\mathbf{x}) = \sum_{\mathbf{z} \in X} b_{etter}(\mathbf{x}, \mathbf{z}). \tag{3}$$

Especially, by Definition 2.1, one knows that Eq. 3 is equivalent to the following formula,

$$L_{evel}(\mathbf{x}) = \begin{cases} |m(\mathbf{x})|, & \mathbf{x}\,\text{is feasible}, \\ |m(\mathbf{x})| - |X|, & \mathbf{x}\,\text{is infeasible}, \end{cases} \tag{4}$$

where $|Z|$ represents the number of elements in set Z. Following the design of the evaluation index, the following properties are acquired, among which only the proof of Theorem 2.1 is given in "Appendix 1".

Theorem 2.1 *Given* $\mathbf{x}, \mathbf{y} \in X$, *if* $L_{evel}(\mathbf{x}) > L_{evel}(\mathbf{y})$, *then* $\mathbf{y} \not\prec_c \mathbf{x}$.

This theorem hints that $\mathbf{x}$ is not worse than $\mathbf{y}$ if $L_{evel}(\mathbf{x}) > L_{evel}(\mathbf{y})$. Further, one can conveniently decide whether a candidate is superior to another one in X through the following corollary.

Corollary 2.1 *Given* $\mathbf{x}, \mathbf{y} \in X$, *there are the following properties according to whether* $\mathbf{x}$ *and* $\mathbf{y}$ *are feasible:*

(a) *if* $\mathbf{x}$ *is feasible, then* $L_{evel}(\mathbf{x}) \geq 0$; *otherwise, there must be* $L_{evel}(\mathbf{x}) < 0$;
(b) *if* $\mathbf{x}$ *and* $\mathbf{y}$ *are feasible,* $\mathbf{x} \prec_c \mathbf{y} \Leftrightarrow L_{evel}(\mathbf{x}) > L_{evel}(\mathbf{y})$, *and* $\mathbf{x}|_c\mathbf{y} \Leftrightarrow L_{evel}(\mathbf{x}) = L_{evel}(\mathbf{y})$;
(c) *if* $\mathbf{x}$ *and* $\mathbf{y}$ *are infeasible,* $L_{evel}(\mathbf{x}) \leq L_{evel}(\mathbf{y}) \Rightarrow \mathbf{x} \not\prec_c \mathbf{y}$.

3 Clonal selection principle

The natural immune system is one of the most important organisms for living bodies. When an antigen intrudes the immune organism, the receptors of the specific lymphocytes, e.g., B, T and memory cells, commence responding to the antigen, owing to the intrinsic functional of immune protection. The repetitive process of recognition, proliferation, learning, memory and combat against one such antigen guarantees that a large number of plasma and memory cells can be created. During this process of immune response, some matureted B cells can be found in that the active ones suffer a repetitive process of affinity maturation after cloning. Thereafter, such maturated B cells secrete antibodies capable of neutralising the intruding antigen, among which some antibodies become memory cells which circulate between blood, lymph, or tissues. If the living bodies encounter the same or similar antigens once again, memory cells can rapidly destruct them. Such response can be explained by the two well-known principles of clonal selection and immune network. The clonal selection principle formulates a process of immune evolution, which B cells react to the invading antigen. To this point, some B cells that best match to the antigen are first selected to proliferate by cloning. Further, the clones differentiate into plasma and memory cells. The memory cells can urge the immune system to make a faster response. Second, the plasma cells suffer a process of affinity maturation. Subsequently, a suppression mechanism is taken to eliminate those redundant cells.

The above immune response theory is a useful bio-inspiration capable of being utilized to investigate omni-AIOS for omni-optimization. To achieve such goal, the self-maintenance of diversity based on suppression is devoted to find diverse candidates, and the metaphors of antibody evolution above are contributed to search simultaneously for multiple high-quality candidates.

4 Formulation of omni-AIOS and designs of mechanisms

4.1 omni-AIOS' formulation

Let antigen Ag stand for problem (**P**) as in Sect. 2.1, and antibody Ab be viewed as a real-coded feasible or infeasible candidate. Memory cells represent those best antibodies without any constraint violation found until the current population. omni-AIOS can be described in detail below.

omni-AIOS: Artificial Immune Optimization System

```
1. Input population size $N$, memory size $M_e$ and terminational iteration number $T$,
   suppression magnitude $\beta$, distribution index for mutation $\mu$, mutation rate $p_m$.
2. Set empty memory archive, $M_{emory} := \phi$, and $n := 1$.
3. Generate initial antibody population $A_n$ with size $N$.
4. While $n \le T$ do
5.   Execute evaluation on $A_n$ through equation (3);
6.   Split $A_n$ into subclasses, $(F_1, F_2, ...) :=$Fast_Sorting $(A_n)$;
7.   Update memory set, $M_{emory} :=$Update$(M_{emory}, F_1)$;
8.   Set $j := 1$ and empty clonal archive, $C^* := \phi$;
9.   While $|C^*| \le N$ do
10.      Carry out dynamic proliferation, $F_j' :=$Proliferate$(F_j)$;
11.      Find clones mutated, $C_j^* :=$Mutate$(F_j')$;
12.      Collect clones, $C^* := C^* \cup C_j^*$;
13.      $j := j + 1$.
14.  End while
15.  Create the next population, $A_{n+1} :=$Select$(F_1 \bigcup C^*)$.
16. End while
17. Output: $M_{emory}$.
```

In the above formulation, steps 5 and 6 aim at rapidly sorting the current population into subclasses, in which the elements of each subclass have the same importance. Step 10 eliminates the same or similar elements through the information on antibody distribution, while proliferating some clones by means of the survival antibodies. Steps 10 to 11 create new antibodies for each subclass. Steps 4 to 16 constitute a loop of immune evolution which finds the desired solutions.

4.2 Module illustration

In the case of multi-objective optimization as in problem (**P**), i.e., $M > 1$, Corollary 2.1 as in Sect. 2.2 provides a guideline for rapidly dividing a finite population Z into subclasses; however, when $M = 1$, another segment method is needed to divide population Z. All these can be achieved by the following module.

```
Pseudo-code for population sorting
$(F_1, F_2, ...)$ :=Fast_Sorting($A_n$)                         ▷ Best subclass $F_1$ and so on
$Z \leftarrow A$, $Z := (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$;        ▷Decreasingly rank all elements by equation (4)
$l \leftarrow 1$, $k \leftarrow 1$;
                                  ▷ Multi-objective case ($M > 1$)
while $|Z| > 0$ do
    $F_l \leftarrow \{\mathbf{x}_k\}$, $Z \leftarrow Z \backslash \{\mathbf{x}_k\}$;              ▷ Insert $\mathbf{x}_k$ into $F_l$ and eliminate it from $Z$
    while $Level(\mathbf{x}_k) = Level(\mathbf{x}_{k+1})$ do              ▷ Check the same important elements
        $k \leftarrow k+1$,$F_l \leftarrow F_l \cup \{\mathbf{x}_k\}$, $Z \leftarrow Z \backslash \{\mathbf{x}_k\}$;
    end while
    $l \leftarrow l+1$, $k \leftarrow k+1$;                          ▷ Create the next subclass
end while
                                  ▷ Single-objective case ($M = 1$)
$F_l \leftarrow \{\mathbf{x}_k\}$, $Z \leftarrow Z \backslash \{\mathbf{x}_k\}$;
$k := 2$;
while $|Z| >= N - 2$ do                                     ▷ Decide subclasses
    while $\beta \times d(\mathbf{x}_k, \mathbf{x}_{k+1}) < d(\mathbf{x}_{k-1}, \mathbf{x}_k) + d(\mathbf{x}_{k+1}, \mathbf{x}_{k+2})$ do ▷ Check similarity
        $F_l \leftarrow F_l \cup \{\mathbf{x}_k\}$, $Z \leftarrow Z \backslash \{\mathbf{x}_k\}$,$k \leftarrow k+1$;
    end while
    $l \leftarrow l+1$;
end while
$F_l \leftarrow F_l \cup \{x_{N-1}, x_N\}$;                        ▷ Collect the last two antibodies
return $F_1, F_2, ...$
```

In the above module, after ranking the antibodies by means of Eq. 3, A_n is split into subclasses. Note that in the case of $M > 1$, if there is at least a feasible antibody in F_1, all elements in such subclass are feasible. The notation of $d(\mathbf{x}, \mathbf{y})$ represents the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. In addition, from the viewpoint of computational complexity, the maximal computational complexity is $O(N\, log_2^N)$, because of the sorting method.

In Update (M_{emory}, F_1), those elements in F_1 without any constraint violation are admitted to enter into M_{emory}. Further, eliminate those identical or dominated elements in terms of the above concept of constraint-dominance. Thereafter, if the size of the memory set is beyond M_e with $M_e \leq N$, the well-known crowding distance method is used to delete those redundant elements. Notice that the distance measure in such method is executed in the objective space but in the design space if a multi-global optimization problem is solved.

Proliferate (F_j) only keeps those diverse antibodies in F_j and decide their clonal sizes. To this end, when $M > 1$, a suppression radius $\rho_j(\mathbf{x})$ of $\mathbf{x} \in F_j$ is designed as follows,

$$\rho_j(\mathbf{x}) = \frac{\sigma_j}{1 + \sigma_j + d_{\min}(\mathbf{x})}, \tag{5}$$

where $d_{\min}(\mathbf{x})$ is the Euclidean distance between $\mathbf{x}$ and the $\mathbf{x}$'s nearest element in F_j; σ_j is the variance of distances $d_{\min}(\mathbf{x})$ with $\mathbf{x} \in F_j$, which reflects the diversity information on F_j. Subsequently, in relation to Eq. 5, the set of those antibodies in the neighborhood of $\mathbf{x}$ is defined as

$$c_j(\mathbf{x}) = \{\mathbf{y} \in F_j : d(\mathbf{x}, \mathbf{y}) < \rho_j(\mathbf{x})\}; \tag{6}$$

so, $|c_j(\mathbf{x})|$ is viewed as the clonal size of $\mathbf{x}$. Notice that once some antibody, e.g. $\mathbf{y}$, belongs to the neighborhood of $\mathbf{x}$, $|c_j(\mathbf{y})|$ is set as 0; in such case, $\mathbf{y}$ is said to be suppressed by $\mathbf{x}$. After so, those antibodies in F_j, which are not suppressed by others, proliferate their clones through

$$F_j' = \{(\mathbf{x}, |c_j(\mathbf{x})|) : \mathbf{x} \in F_j\}. \tag{7}$$

Besides, when $M = 1$, pick up the best element in F_j' e.g. $\mathbf{x}_0$, and accordingly, $F_j' = \{(\mathbf{x}_0, |F_j|)\}$.

In the above scheme, when $M > 1$, different antibodies are allocated different suppression radiuses, which avoids effectively the difficulty of deciding the adjustable population suppression radius. Equations 5 and 6 hint an important idea: the larger the subclass variance, the more the suppressed antibodies; conversely, the fewer antibodies are suppressed.

Mutate (F_j') only keeps those mutated clones. In other words, all clones in F_j' change their genes with the same mutation rate p_m through the polynomial mutation [3], namely, $\mathbf{x}' = \mathbf{x} + \gamma_j \Delta_{\max}$, where γ_j is defined as follows:

$$\gamma_j = \begin{cases} (2u)^{\frac{1}{\eta_j+1}} - 1, & \text{if } u < 0.5, \\ 1 - [2(1-u)]^{\frac{1}{\eta_j+1}}, & \text{if } u \geq 0.5, \end{cases} \tag{8}$$

with uniformly distributed random variable $u \in (0, 1)$. Notice that $\eta_j = \mu(1 + \frac{j-1}{L_p})$ with $\mu > 1$, and L_p is the number of online obtained subclasses in step 6 as in omni-AIOS.

Select ($F_1 \cup C^*$) consists of some better and diverse antibodies from a combinational population $F_1 \cup C^*$. Let $F_f(X)$ and $F_{inf}(X)$ be composed of feasible and infeasible antibodies in X respectively. Hence, the desired set is decided through the following pseudo-code.

```
Pseudo-code for selection
$P =: Select(F_1 \bigcup C^*)$
$X_1 \leftarrow F_f(F_1 \bigcup C^*)$, $X_2 \leftarrow F_{inf}(F_1 \bigcup C^*)$;        ▷ Population division
if $|X_1| \geq N$
  $P \leftarrow F_{non}(X_1)$;        ▷ Insert different and better antibodies into set $P$
  if $|P| > N$
    $P \leftarrow Crowding(P, N)$;        ▷ Keep N diverse antibodies
  end if
end if
if $|X_1| < N$
  $P \leftarrow X_1 \cup Crowding(X_2, N - |X_1|)$;        ▷ Pick up some diverse antibodies in $X_2$
end if
return $P$
```

In the above module, $F_{non}(X_1)$ comprises of nondominated antibodies in X_1 if $M > 1$ and different ones otherwise. Also, *Crowding* (X, m) only keeps m antibodies in X by means of the crowding distance method in which the distance measure is the same as that in Update (M_{emory}, F_1).

4.3 Comparative analysis

This section presents the main differences of omni-optimizer [7], omni-aiNet [41] and omni-AIOS. Besides the schemes of constraint-handling and mutation, these three kinds of approaches have apparently different characteristics and performances, due to their bio-inspiration and design ideas. The major differences are listed below.

- Bio-inspiration. omni-optimizer bases on the theory of evolution and genetics. omni-aiNet is related to the clonal selection principle, immune network theory and DNA-based gene duplication. omni-AIOS is an immune optimization algorithm only based on the clonal selection principle. Although omni-aiNet and omni-AIOS share the biological inspirations of cloning, hypermutation and selection, their schemes of operations, except the similar mutation, are greatly different, which can be known below.
- Parameter settings. omni-AIOS demands 6 parameters, i.e., population size, memory size, maximal iteration number, suppression magnitude, distribution index for mutation and mutation rate, whereas either of omni-optimizer and omni-aiNet has 7 parameters [41]. Especially, omni-AIOS and omni-optimizer have the fixed population size, but omni-aiNet performs with a dynamic population size. Additionally, omni-optimizer consists of 6 operations: Tournament, Crossover, Mutation, Ranking, Crowding distance and Sorting; omni-aiNet is composed of 7 operations: Cloning, Hypermutation, Gene duplication, Ranking, Grid procedure, Binary tournament and Random insertion; omni-AIDOS only includes 5 operations: Memory update, Fast sorting, Dynamic proliferation, Mutation and Selection.
- Comparison between individuals. These approaches adopt the same idea of penalty to handle the constraints. omni-AIOS performs comparison between individuals in terms of the concept of constraint-dominance, but omni-optimizer and omni-aiNet do so through constrained ε-dominance.
- Population sorting. omni-AIOS utilizes one dominance-based evaluation index proposed in this work to decide the levels of individuals in the current population with computational cost $O(Nlog_2^N)$, and then those individuals with the same level are put into the same subclass; on the other hand, omni-optimizer and omni-aiNet uses the well-known nondominated sorting method based on ε-dominance to divide the population, where the computational cost is $O(MN^2)$.
- Evolution. In the process of evolution, although omni-optimizer, omni-aiNet and omni-AIOS share the same scheme of polynomial mutation, their distribution indices for mutation, η, are determined with different fashions. In omni-optimizer, η is a fixed parameter defined by the user; omni-aiNet determines it dynamically at the interval [5, 20]; omni-AIOS decides it dynamically at the interval [μ, 2μ). It should be pointed out that omni-aiNet and omni-AIOS involve in completely different proliferation schemes. Namely, omni-aiNet demands that each individual in the current population with size N proliferate a clonal subset of fixed size N_c so that the sum of sizes of clonal subsets is NN_c, whereas omni-AIOS utilizes one niching-like suppression radius function proposed in this work to delete those redundant individuals in the current population; in omni-AIOS, those survival individuals receive their respective clonal sets with different sizes, in which the number of clones for a given survival

individual is the number of individuals suppressed by it, and hence the number of all clones is N.

- Selection. omni-optimizer picks up N better and diverse individuals in a combinational population of size $3N$ to constitute the next population, relying upon the crowding distance method of which Euclidean distances are calculated both in the design space and in the objective space. In omni-aiNet, the process of selection is carried out in a combinational population of size $N(1 + N_c)$, by means of the conventional nondominated sorting method and the grid procedure. However, omni-AIDOS selects N better individuals in a combinational population of size $2N$ through the evaluation index above and the crowding distance method in which Euclidean distances are taken into account in the object space or only in the design space (when a multi-global optimization problem is solved).

4.4 Computational complexity and convergence

In this subsection, two theoretical results are given, whose proofs are displayed in "Appendix 1". Following the description of omni-AIOS above, one can see that steps 5, 6, 7, 10 and 15, which decide omni-AIOS' computational complexity, need to compare those antibodies or calculate their objectives and constraints.

Theorem 4.1 *The maximal complexity of omni-AIOS is* $O(N(N(M + I + J) + p))$, *where* M, I, J, *and* p *are mentioned in Sect.* 2.1.

In order to analyze omni-AIOS' convergence, let S be the antibody space (i.e., design space), and suppose that S is a finite set. Let $S^{\leq N}$ represent a state space composed of antibody populations whose sizes are not beyond N. Each element of the state space is called a state. omni-AIOS can be formulated by the evolution chain,

$$A_n \to (F_1, F_2, \ldots) \to (F'_1, F'_2, \ldots) \to C^* \to A_{n+1}.$$

Since its operations are independent of n, the above chain is a Markov chain. Further, let $M_f(S, \prec_c)$ be the set of the optimal or Pareto-optimal solution(s) of problem (**P**) as in Sect. 2.1. This way, the following conclusion can be acquired.

Theorem 4.2 *omni-AIOS is convergent for any initial population distribution, i.e.,*

$$\lim_{n\to\infty} Pr\left\{A_n \cap M_f(S, \prec_c) \neq \emptyset\right\} = 1. \tag{9}$$

5 Performance criteria

Three criteria [1, 31], suitable for multi-objective optimization, are cited below. Assume that two algorithms A and B are executed respectively K times on a given multi-objective problem. Accordingly, two serials of solution sets are acquired, $\{A_k\}_{k=1}^K$ and $\{B_k\}_{k=1}^K$.

- Average coverage ratio (ACR). ACR is a criterion which measures the whole difference of the optimized qualities for algorithms A and B, defined as

$$\mathrm{ACR}(A, B) = \frac{1}{K^2}\sum_{i,j=1}^{K} C(A_i, B_j), \tag{10}$$

where

$$C(A_i, B_j) = \frac{|\{y \in B_j : \exists x \in A_i, s.t.x \prec_c y\}|}{|B_j|}. \tag{11}$$

Equation 10 means that algorithm A has the globally better solution quality than algorithm B if ACR(A, B) > ACR(B, A).

- Average density (AD) and average coverage scope (ACS). AD can be used to evaluate the average distribution performance of the solution sets found, given by

$$\mathrm{AD} = \frac{1}{K}\sum_{i=1}^{K}\sqrt{\frac{1}{|A_i| - 1}\sum_{j=1}^{|A_i|}(\bar{d}_i - d_{ij})^2}, \tag{12}$$

where

$$d_{ij} = \min_{j\neq l, 1\leq l\leq |A_i|}\{d(x_j, x_l)|x_j, x_l \in A_i\}, \bar{d}_i = \frac{1}{|A_i|}\sum_{j=1}^{|A_i|} d_{ij}.$$

Equation 12 hints that the solution sets gotten by algorithm A have good distribution if AD is small. Average coverage scope ACS is the average coverage width of all the solution sets obtained by such algorithm.

6 Numerical experiments

This section examines omni-AIOS' performances including its searching effect, efficiency and computational complexity, relying upon sixteen single and multi-objective, uni-global benchmark problems and two single and multi-objective, multi-global test problems listed in detail in "Appendix 2". All experiments are executed on a personal computer with CPU/3.0 GHz and RBM/2.0 GB. Two representative omni-optimization techniques, i.e., omni-optimizer and omni-aiNet proposed by Deb and Tiwari [7] and by Coelho and Von Zuben [41], respectively, are chosen to compare against omni-AIOS. In order to list the statistical results, the memory set of M_{emory} as in Sect. 4.2 is also inserted into omni-optimizer and omni-aiNet. It is used to store and update the solutions found. In addition, omni-AIOS and omni-optimizer have the same population size

60; in omni-aiNet, the initial population size is also given 60; the number of generations between suppressions is designated as 1, for which the purpose is to require that the three approaches have the same terminational criterion. The same terminational iteration of such three methods is specified according to different kinds of optimization problems below. The remaining parameter settings of omni-optimizer and omni-aiNet are taken from the references [7, 41]. For omni-AIOS, by experimental tuning the rational value ranges of β, μ and p_m are obtained for all the test problems, i.e., $\beta \in (0.1, 0.6)$, $\mu \in [5, 10]$ and $p_m \in [0.8, 1]$. The values of such parameters are different according to different kinds of test problems, which can be found in the following subsections.

6.1 Single-objective, uni-global constrained optimization

In this subsection, the three algorithms are tested on eight benchmark problems g01 to g08 given in "Appendix 2"; especially, the dimensions of g02 and g03 are taken 20. Since the problems g01 to g04 are solved relatively easily, all the algorithms take their terminational iterations 2000 for these four problems and 5000 otherwise; the size of M_{emory} is 1. In omni-AIOS, take $\beta = 0.5$, $\mu = 10$ and $p_m = 0.8$. Following these preliminaries, the three algorithms are respectively executed 100 runs on each test problem. The acquired statistical results are listed in Tables 1 and 2 below.

Through the Tables 1 and 2, the approaches can all deal effectively with g01, g03 and g08. For g02 and g04, omni-aiNet and omni-AIOS can acquire the better effects than omni-optimizer. Notice that although g05 has not any feasible solution, omni-optimizer and omni-AIOS can all find the suboptimal solutions, and omni-AIOS presents the relatively stable searching effect because of the values on Mean and SD; however, omni-aiNet can only acquire some solutions with somewhat large constraint violations. When solving g06, omni-aiNet can get the best effect, and omni-AIOS is secondary; when dealing with g07, either omni-optimizer or omni-aiNet can find better solutions than omni-AIOS, but the values on Best and Mean illustrate that omni-AIOS can acquire the stable effect for 100 runs. On the other hand, through the runtime of each algorithm as in Tables 1 and 2 for each test problem, one can see that the spent runtime of omni-AIOS is at most 63.29% of that required by omni-aiNet for any of g01, g02, g03 and g04, and 33.78% for each of the other test problems. It is also easy to see that omni-AIOS is globally faster than omni-optimizer when solving the test problems.

Table 1 Statistical results obtained by omni-AIOS with 100 runs

Algori.	Prob.	Optimum	Best	Mean	SD	AT
omni-AIOS	g01	−15	−15	−14.9996	3.1101×10^{-3}	2.81
	g02	−0.803619	−0.799339	−0.771373	0.0221	6.41
	g03	−1.0	−1	−0.999959	1.13×10^{-4}	6.16
	g04	−30665.539	−30664.2	−30656.8	10.8149	3.77
	g05	5126.498	5126.49	5217.5	219.295	13.80
	g06	−6961.814	−6924.03	−6921.76	10.7334	5.87
	g07	24.306	24.7496	24.7511	5.74×10^{-3}	12.46
	g08	−0.095825	−0.095825	−0.0958249	9.385×10^{-8}	6.18

AT average runtime (second)

Table 2 Statistical results gotten by omni-optimizer and omni-aiNet with respectively 100 runs

Prob.	omni-optimizer [7]				omni-aiNet [41]			
	Best	Mean	SD	AT	Best	Mean	SD	AT
g01	−14.9994	−14.944	0.2113	6.24	−15	−14.9995	5.65×10^{-4}	4.77
g02	−0.792516	−0.746811	0.0387	10.40	−0.803526	−0.69232	0.05878	10.13
g03	−1	−0.999981	1.52×10^{-5}	8.90	−1	−0.9999996	1.47×10^{-6}	13.12
g04	−30658	−30577.7	86.0179	3.717	−30665.2	−30592.6	82.7524	6.93
g05	5126.18	5280.47	267.107	10.4	5067.37	5400.59	266.053	63.07
g06	−6810.37	−6704.28	45.397	6.25	−6959.15	−6945.31	8.76529	17.52
g07	24.4788	26.2533	1.41209	17.0	24.329	27.2028	2.03486	17.8
g08	−0.095825	−0.0958249	2.89×10^{-7}	7.41	−0.095825	−0.0958249	1.87×10^{-7}	18.34

Summarily, omni-AIOS can obtain the relatively satisfactory effects for all the test problems except g06, while taking less time to solve each problem. For the two algorithms compared, omni-aiNet spends more time to deal with these problems except g05, and their performance effects have slight differences.

6.2 Multi-objective, uni-global constrained optimization

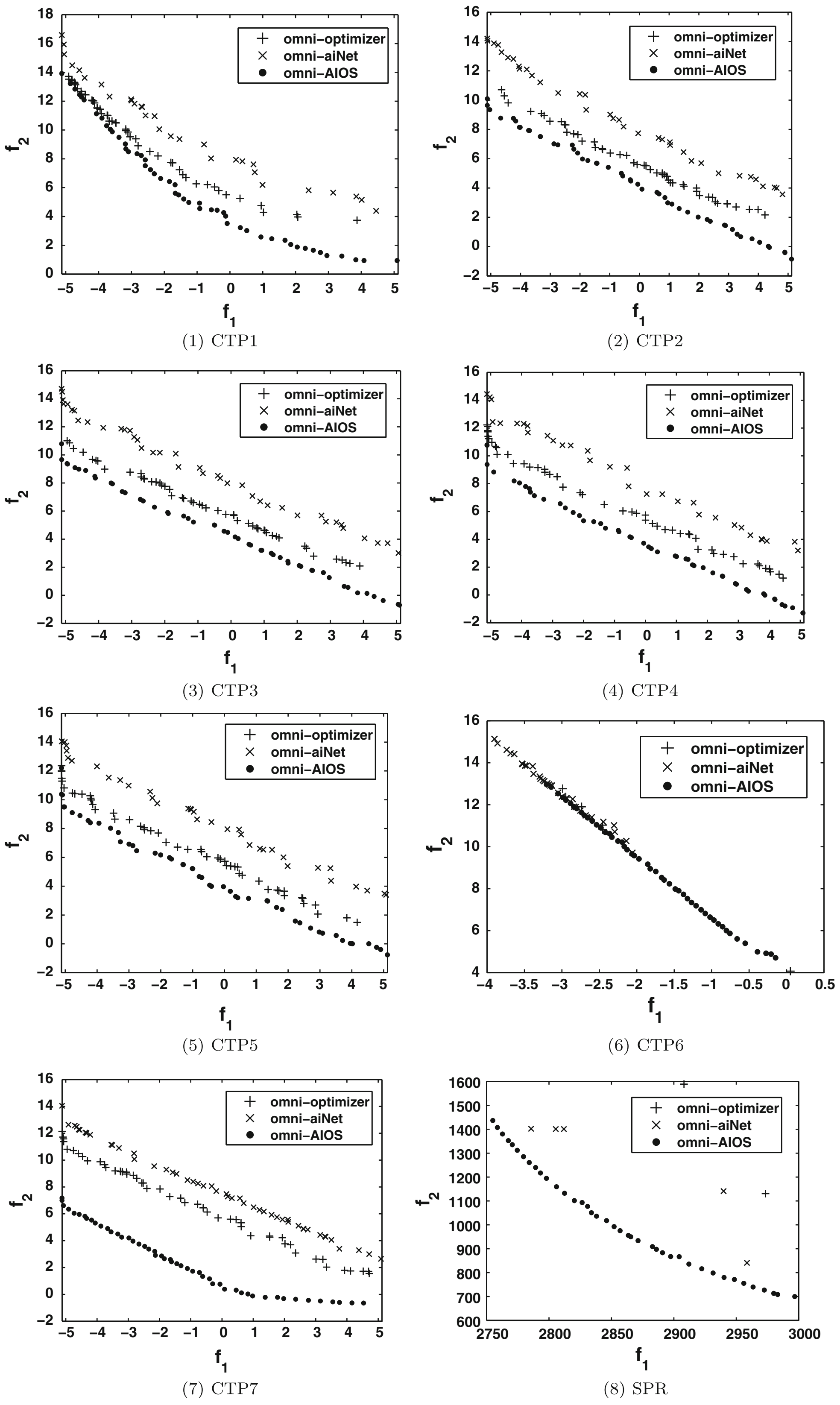
Depending on the three performance criteria as in Sect. 5, the three algorithms are examined their abilities of solving constrained multi-objective problems by means of a suite of difficult problems CTP1 to CTP7 and one difficult engineering optimization problem SPR as in "Appendix 2". The size of M_{emory} is set as 50. Further, each of the above algorithms is continually executed 30 runs on each of the problems. In omni-AIOS, take $\beta = 0.5$, $\mu = 10$ and $p_m = 1$. On one hand, the dimensions of those theoretical test problems CTP1 to CTP7 are all designated as 100; the purpose doing so is to observe whether the above three algorithms can efficiently solve high-dimensional optimization problems with constraints. Especially, take $J = 30$ in CTP1; namely, CTP1 includes 30 constraints. Thereafter, the statistical values are listed in Table 3 below, while the nondominated points found by the three algorithms with respectively only one execution for each test problem are drawn in Fig. 1 below. Table 3 and Fig. 1 can draw the following conclusions.

The experimental results illustrate that omni-optimizer, omni-aiNet and omni-AIOS can all find some feasible solutions for each of the theoretical test problems CTP1 to CTP7, but their optimized qualities have significant differences. The values on ACR in Table 3 illustrate that each nondominated set gotten by omni-AIOS can almost cover 98% of that obtained by either of the other two algorithms for each of the theoretical test problems except CTP1 and CTP6, and on the other hand, for each of such test problems except CPT6, each of nondominated sets found by omni-optimizer can almost cover 92% of that obtained by omni-aiNet. All these results indicate that omni-AIOS can obtain the best effects when solving the above high-dimensional multi-objective optimization problems. Besides, the statistical values on AD and ACS in Table 3 show that the nondominated points in the objective space found by omni-AIOS are with satisfactory distribution and wide coverage scope; especially, one can see that those nondominated points gained by omni-AIOS and omni-aiNet have similar distribution for some test problems. It should be pointed out

Table 3 Comparison of statistical results of nondominated sets found with 30 executions. ACR, AD and ACS are mentioned in Sect. 5

Prob.	ACR(.,.) (%)	omni-optimizer [7]	omni-aiNet [41]	omni-AIOS	AD	ACS	AT
CPT1	omni-optimizer	0	92.75	17.33	0.21	13.10	4′47″
	omni-aiNet	0.07	0	0	0.24	15.59	1′46″
	omni-AIOS	51.47	99.76	0	0.14	20.07	54″
CPT2	omni-optimizer	0	96.22	0.20	0.16	13.41	3′28″
	omni-aiNet	0	0	0.07	0.24	15.18	14″
	omni-AIOS	99.53	99.05	0	0.27	15.95	17″
CPT3	omni-optimizer	0	96.62	0	0.17	13.69	3′20″
	omni-aiNet	0	0	0	0.26	15.03	17″
	omni-AIOS	99.87	99.25	0	0.23	15.90	14″
CPT4	omni-optimizer	0	94.63	0	0.15	13.32	3′8″
	omni-aiNet	0	0	0	0.24	15.31	16″
	omni-AIOS	99.67	99.17	0	0.20	15.79	14″
CPT5	omni-optimizer	0	94.63	0	0.15	13.31	3′41″
	omni-aiNet	0	0	0	0.25	15.26	14″
	omni-AIOS	99.93	98.79	0	0.22	15.69	16″
CPT6	omni-optimizer	0	3.20	1.62	0.16	11.02	3′8″
	omni-aiNet	55.25	0	2.47	0.16	6.5	1′15″
	omni-AIOS	65.97	39.91	0	0.05	9.32	15″
CPT7	omni-optimizer	0	97.70	0	0.15	14.12	2′45″
	omni-aiNet	0	0	0	0.24	15.21	1′21″
	omni-AIOS	99.93	100	0	0.23	14.79	26″
SPR	omni-optimizer	0	0	0	51.95	279.99	37′30″
	omni-aiNet	75.00	0	0	94.21	400.68	7′30″
	omni-AIOS	100	100	0	6.97	1030.39	4′

Fig. 1 Comparison of nondominated fronts for CPT 1, CPT2, …, CPT7 and SPR



that omni-optimizer seems to be able to find better distributed nondominated points than either of omni-AIOS and omni-aiNet for some problems; in fact, many nondominated points found by it have great similarity, which can be known in terms of the figures except Fig. 1(8).

On the other hand, in the case of solving the engineering optimization problem, although the three algorithms can all find some feasible solutions, through Table 3 one can know that omni-AIOS can acquire the best optimized quality in that each nondominated set found by it covers 100% of that gotten by either of omni-optimizer and omni-aiNet, which can also be illustrated by Fig. 1(8). Furthermore, those nondominated sets gotten by omni-AIOS are with the widest coverage scope and the satisfactory distribution, in comparison with those obtained by the other two algorithms. Further, for one such problem, omni-aiNet can obtain the better solution quality than omni-optimizer, as each nondominated set gotten by the former covers 75% of that acquired by the latter. Thus, for this problem, omni-AIOS can acquire the best effect, and omni-aiNet is secondary.

In addition, the three algorithms all adopt the constraint-dominance concept to handle the constraints, whereas they have different efficiencies. Table 3 shows that omni-AIOS is faster than either of the other two algorithms when searching for the Pareto-optimal solutions for each test problem; especially, omni-AIOS' average runtime is at most 1/2 of that required by omni-aiNet for CTP1, CTP6, CTP7 and SPR. When solving each of other problems, the average runtime demanded by omni-aiNet is almost the same as that required by omni-AIOS. Further, one can also see that omni-AIOS spends at most 1/12 of the runtime required by omni-optimizer to solve each test problem. The main reason is that the evaluation index proposed in this work reduces greatly omni-AIOS' computational complexity. Experiments illustrate that omni-optimizer has high complexity, because it adopts the crowding distance method involving both the design space and the object space.

Summarily, for each of the above multi-objective problems, the three algorithms display their respective behaviors. omni-AIOS can obtain the more satisfactory effect and higher efficiency than any of omni-optimizer and omni-aiNet. On the other hand, omni-optimizer can acquire the better effect than omni-aiNet, but it results in high-computational complexity.

6.3 Single-objective, multi-global optimization

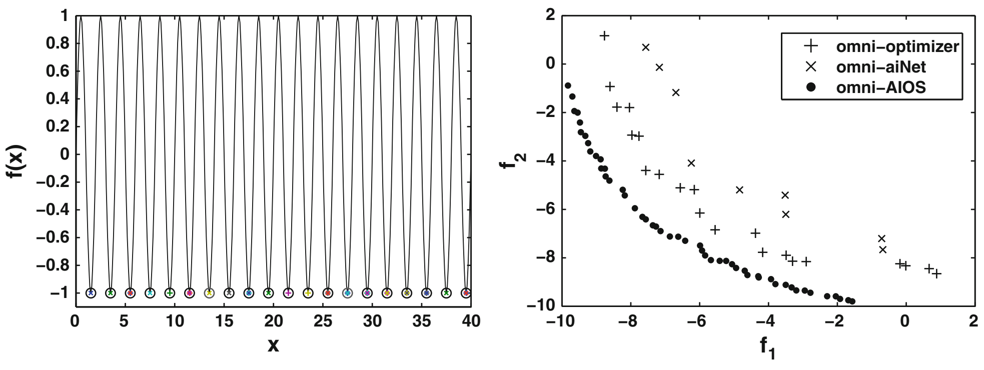
Here, an optimization problem G01, proposed by Deb and Tiwari [7], is listed in "Appendix 2". It includes twenty optimal solutions $2k - \frac{1}{2}$ with $k = 1, 2, \ldots, 20$. The above algorithms execute 20 times with the same terminational iteration number 500. The size of M_{emory} is set as 20. In omni-AIOS, take $\beta = 0.3$, $\mu = 6$ and $p_m = 0.3$.

Although the three algorithms can all find the minimum for a single run, they have subtle differences with respect to sizes of solution sets found and computational time. omni-optimizer, omni-aiNet and omni-AIOS acquire in turn 15, 17 and 20 optimal solutions, which can be known from Fig. 2(1) below. Relatively, omni-aiNet can get more optimal solutions for some runs. On the other hand, they spend averagely 1.31, 1.28 and 1.29 s to find optimal solutions within a run period. Summarily, omni-AIOS is also competitive for such problem.

6.4 Multi-objective, multi-global optimization

The bi-objective optimization problem G02 with dimension 10 given in "Appendix 2", proposed by Deb and Tiwari [7], is cited to examine the above algorithms with the same terminational iteration 500. The size of M_{emory} is 50. In omni-AIOS, take $\beta = 0.5$, $\mu = 10$ and $p_m = 1$. Similarly solving the test problems as in Sect. 6.2, the methods can obtain their statistical results listed in Table 4 below. Their Pareto-fronts acquired with respectively only one execution are drawn in Fig. 2(2) below.

Table 4 indicates that omni-optimizer and omni-aiNet are difficult in finding the Pareto-optimal solutions in comparison to omni-AIOS, as their nondominated points



(1) G01 (2) G02

Fig. 2 **(1)** comparison of sizes of solution sets found for G01; the notations of +, x and o denote the minimal points gotten by omni-optimizer, omni-aiNet and omni-AIOS, respectively. **(2)** nondominated fronts found for G02

Table 4 Comparison of statistical results for G02

ACR(.,.) (%)	omni-optimizer [7]	omni-aiNet [41]	omni-AIOS	AD	AS	AT
omni-optimizer	0	96.70	0	0.51.21	12.22	1.08″
omni-aiNet	1.31	0	0	68.48	10.94	0.35″
omni-AIOS	96.86	97.78	0	0.10	12.98	0.30″

found are far from the theoretical Pareto-front, which can be known through Fig. 2(2). omni-AIOS can acquire the best effect for such problem, and omni-optimizer is secondary. In addition, omni-optimizer spends more time to solve such problem than either of omni-aiNet and omni-AIOS. All these present that omni-AIOS is useful for multi-objective, multi-global optimization problems.

7 Conclusions

With the increasing requirement of solving real-world optimization problems, omni-optimization will become increasingly popular. Thus, this work suggests an artificial immune optimization system (omni-AIOS) capable of handling omni-optimization, inspired by the dynamic characteristics and mechanisms of the immune system and also the existing concept of constraint-dominance. In omni-AIOS, an efficient population division scheme is developed to enhance the speed of optimization by means of an evaluation index proposed in this work, and meanwhile the niching-like proliferation scheme strengthens the diversity of population as well as the ability of exploration and exploitation. Further, the theoretical results demonstrate that omni-AIOS is convergent with low-computational complexity. Experimentally, it is compared against the two existing optimization algorithms for omni-optimization. The experimental results demonstrate that: (1) omni-aiNet is more suitable for single-objective problems but somewhat high-computational complexity; omni-optimizer is more useful for multi-objective problems but high-computational complexity, and (2) in comparison with omni-optimizer and omni-aiNet, omni-AIOS has low-computational complexity, while being capable of effectively dealing with omni-optimization with nonlinear constraints and low or high dimensions.

Acknowledgements This work is supported in part by National Natural Science Foundation NSFC (60565002, 61065010), China.

Appendix 1: Proof of conclusions

Proof of Theorem 2.1 Assume $\mathbf{y} \prec_c \mathbf{x}$. First, if both $\mathbf{x}$ and $\mathbf{y}$ are feasible, one can see $L_{evel}(\mathbf{x}) < L_{evel}(\mathbf{y})$, due to $|m(\mathbf{x})| > |m(\mathbf{x})|$. Second, if $\mathbf{y}$ is feasible but $\mathbf{x}$ is not, it follows similarly that $L_{evel}(\mathbf{x}) < L_{evel}(\mathbf{y})$. Further, if $\mathbf{y}$ is not feasible, $\mathbf{x}$ is also so because $\mathbf{y} \prec_c \mathbf{x}$, and accordingly one gets $L_{evel}(\mathbf{x}) < L_{evel}(\mathbf{y})$, because of $|m(\mathbf{y})| > |m(\mathbf{x})|$. These cases illustrate that the conclusion is true. □

Proof of Theorem 4.1 Only give the proof of computational complexity for the multi-objective optimization case ($M > 1$). In step 5, each antibody is required to compute and compare $[(N-1)(M+I+J)+1]$ times; step 6 needs to rank those antibodies with $N\,log_2^N$ times. These antibodies are required to enter into corresponding subclasses. This needs $(N-1)$ times to check which antibodies have the same importance. Therefore, steps 5 and 6 execute to compare and compute a_n times,

$$a_n = N[(M+I+J)(N-1)+1] + N\,log_2^N + N - 1$$
$$\leq N(N-1)(M+I+J) + N\,log_2^N + 2N. \tag{13}$$

Hence, the maximal computational complexity for these two steps is $O((M+I+J)N^2)$. In step 7, since M_{emory} only collects feasible antibodies, this step requires b_n comparisons to find at most M_e survival memory cells,

$$b_n \leq (M_e + |F_1|)^2 + (M_e + |F_1|)log_2^{M_e+|F_1|}$$
$$\leq (M_e+N)^2 + (M_e+N)log_2^{M_e+N}. \tag{14}$$

On the right side of Eq. 14, the first term represents the computational times deleting those identical and dominated members in $M_{memory} \cup F_1$; the second term stands for the number of computations which is used to decide M_e memory cells by the crowding distance method. Therefore, the maximal computational complexity of step 7 is $O(N^2)$, due to $M_e \leq N$.

In steps 10 to 12, for the j-th subclass F_j, since each antibody is required to find its nearest one, the amount of computations and comparisons for step 10 is $(M+1)|F_j|(|F_j|-1) + |F_j|^2 + |F_j|$. On the other hand, step 11 needs to execute $p|F_j|p_j$ times. In addition, step 12 evaluates those mutated clones with $(M+I+J)|F_j|$ times. Therefore, the amount of computations and comparisons for these steps is given by

$$c_n = \sum_{j=1}^{l}[(M+1)|F_j|(|F_j|-1) + |F_j|^2 + |F_j|$$
$$+ p|F_j|p_j + (M+I+J)|F_j|]$$
$$\leq N[(M+1)(N-1) + N + p + M + I + J + 1]. \tag{15}$$

Hence, the maximal complexity of such steps is $O(N(N(M + I + J) + p))$.

In step 15, first divide $F_1 \cup C^*$ into two subclasses: feasible solution set X_1 and infeasible solution set X_2, which needs to carry out $|F_1 \cup C^*|$ times; second, if $|X_1| > N$, there needs at most $(M + I + J)|X_1|^2$ times to find nondominated antibodies and conversely, the crowding distance method is executed on X_2, for which the total of computations and comparisons is $M|X_2|log_2^{|X_2|}$. Thus, since $|X_1|$ and $|X_2|$ are not larger than N, this step needs at most d_n times to compute or compare those antibodies,

$$\begin{aligned} d_n &= \max\{(M+I+J)|X_1|^2, M|X_2|log_2^{|X_2|}\} \\ &\le (M+I+J)N^2. \end{aligned} \tag{16}$$

Summarily, because of $M_e \le N$, the overall worst case complexity of Omni-AIOS is

$$\begin{aligned} O(B) &= O((M+I+J)N^2) + O(MN^2) \\ &\quad + O(N(N(M+I+J)+p)) + O((M+I+J)N^2) \\ &\approx O(N(N(M+I+J)+p)). \end{aligned} \tag{17}$$

□

Proof of Theorem 4.2 Given $X \in S^{\le N}$, through steps 6 and 10 there exists $(X_j, X_j') \in S^{\le N} \times S^{\le N}$ satisfying

$$P\{F_j = X_j|A_n = X\} = 1, P\{F_j' = X_j'|F_j = X_j\} = 1. \tag{18}$$

Further, through the mutation operator and for $\forall \mathbf{x} \in X$ and $\mathbf{y} \in S$, there exists some j with $\mathbf{x} \in F_j$. In such case, $\mathbf{x}$ can attain $\mathbf{y}$, provided that μ is decided reasonably as in (8). Thus, $P\{mutate(\mathbf{x}) = \mathbf{y}\} > 0$, where $mutate(\mathbf{x})$ is obtained by $\mathbf{x}$ through the polynomial mutation. Hence, for $Z_j \in S^{\le N}$ with $|Z_j| = |X_j'|$, one can imply that $P\{C_j^* = Z_j |F_j' = X_j'\} > 0$. So, it is necessary to hold the formula for $\forall X, Z \in S^{\le N}$,

$$\begin{aligned} P\{C^* = Z|A_n = X\} &= \prod_{j\ge1} P\{C_j^* = Z_j|F_j = X_j\} \\ &= \prod_{j\ge1} P\{C_j^* = Z_j|F_j' = X_j'\} > 0. \end{aligned} \tag{19}$$

Further, since the series of $\{A_n\}_{n\ge1}$ is monotone, the above conclusion is true, relying upon Theorem 4 [43].

□

Appendix 2: Test problems

This section presents eight popular single-objective, uni-global constrained benchmark problems g01 to g08 [44], one single-objective, multi-global problem G01 [7], eight popular multi-objective, uni-global benchmark problems CPT1 to CPT7 and SPR [45, 46], and one multi-objective, multi-global problem G02 [7].

$$g01.\,\mathrm{Min} f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

s.t.,

$$\begin{aligned} &g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0, \\ &g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0, \\ &g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0, \\ &g_4(x) = -8x_1 + x_{10} \le 0, \\ &g_5(x) = -8x_2 + x_{11} \le 0, \\ &g_6(x) = -8x_3 + x_{12} \le 0, \\ &g_7(x) = -2x_4 - x_5 + x_{10} \le 0, \\ &g_8(x) = -2x_6 - x_7 + x_{11} \le 0, \\ &g_9(x) = -2x_8 - x_9 + x_{12} \le 0, \\ &0 \le x_i \le 1,\ 1 \le i \le 9,\ 0 \le x_i \le 00,\ 10 \le i \le 12,\ 0 \le x_{13} \le 1. \end{aligned}$$

$$g02.\,\mathrm{Min} f(x) = -\left|\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}}\right|$$

s.t.,

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \le 0, g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0,$$

$$n = 20, 0 < x_i \le 10, 1 \le i \le n.$$

$$g03.\,\mathrm{Min} f(x) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i \text{ s.t.,}$$

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0,$$

$$n = 20,\ 0 \le x_i \le 1,\ 1 \le i \le n.$$

$$\begin{aligned} g04.\,\mathrm{Min} f(x) = &5.3578547x_3^2 + 0.8356891x_1x_5 \\ &+ 37.293239x_1 - 40792.141 \end{aligned}$$

s.t.,

$$\begin{aligned} g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 \\ &\quad - 0.0022053x_3x_5 - 92 \le 0, \\ g_2(x) &= -5.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 \\ &\quad + 0.0022053x_3x_5 \le 0, \\ g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 \\ &\quad + 0.0021813x_3^2 - 110 \le 0, \\ g_4(x) &= -0.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 \\ &\quad - 0.0021813x_3^2 + 90 \le 0, \\ g_5(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 \\ &\quad + 0.0019085x_3x_4 - 25 \le 0, \\ g_6(x) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 \\ &\quad - 0.0019085x_3x_4 + 20 \le 0, \end{aligned}$$

$$78 \le x_1 \le 102,\ 33 \le x_2 \le 45,\ 27 \le x_i \le 45 (3 \le i \le 5).$$

$$g05.\,\mathrm{Min} f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

s.t.,

$$g_1(x) = -x_4 + x_3 - 0.55 \le 0, g_2(x) = -x_3 + x_4 - 0.55 \le 0,$$
$$h_1(x) = 1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x1 = 0,$$
$$h_2(x) = 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0,$$
$$h_3(x) = 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0,$$
$$0 \le x_1 \le 1200, 0 \le x_2 \le 1200,\ -0.55 \le x_3 \le 0.55, -0.55 \le x_4 \le 0.55.$$

$$g06.\,\mathrm{Min} f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

s.t.,

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0,$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0,$$
$$13 \le x_1 \le 100,\ 0 \le x_2 \le 100.$$

$$g07.\,\mathrm{Min} f(x) = x_1 1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

s.t.,

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0,$$
$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0,$$
$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0,$$
$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0,$$
$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0,$$
$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0,$$
$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0,$$
$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0,$$
$$-10 \le x_i \le 10,\ 1 \le i \le 10.$$

$$g08.\,\mathrm{Min} f(x) = -\frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

s.t.,

$$g_1(x) = x_1^2 - x_2 + 1 \le 0,$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0,$$
$$0 \le x_1,\ x_2 \le 10.$$

$$G01.\,\mathrm{Min} f(x) = sin(\pi x_1)cos(\pi x_2) 0 \le x_1, x_2 \le 40.$$

$$CPT1.\,\mathrm{Min} f(x) = (f_1(x), f(x))$$

s.t.,

$$f_1(x) = x_1, f_2(x) = g(x)exp(-\frac{f_1(x)}{g(x)}),$$
$$c_j(x) = f_2(x) - a_j exp(-b_j f_1(x)) \ge 0, 1 \le j \le J.$$
$$g(\mathbf{x}) = 1 + \frac{1}{100}\sum_{i=1}^{100}[x_i^2 - cos(2\pi x_i)],$$
$$x_i \in [-5.12, 5.12], 1 \le i \le 100.$$

$$CPT2 - CPT7.\,\mathrm{Min} f(x) = (f_1(x), f(x))$$

s.t.,

$$f_1(x) = x_1, f_2(x) = g(x)(1 - \frac{f_1(x)}{g(x)}),$$
$$c(x) = cos(\theta)(f_2(x) - e) - sin(\theta)f_1(x) \ge a|sin(b\pi(sin(\theta)(f_2(x) - e) + cos(\theta)f_1(x))^c)|^d,$$
$$g(x) = An + \sum_{i=1}^{n} x_i^2 - Acos(\omega x_i), A = 10,$$
$$\omega = 2\pi, x_i \in [-5.12, 5.12], 1 \le i \le n,$$

where $g(x)$ is the same as the function as in CPT1.

$$Speed\,Reducer(SPR).\,\mathrm{Min} f(x) = (f_1(x), f(x))$$

s.t.,

$$f_1(x) = 0.7854x_1x_2^2(\frac{10x_3^2}{3} + 14.933x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2),$$
$$f_2(x) = \frac{\sqrt{(\frac{745x_4}{x_2x_3})^2 + 1.69 \times 10^7}}{0.1x_6^3},$$
$$g_1 : \frac{1}{x_1x_2^2x_3} - \frac{1}{27} \le 0,\ g2 : \frac{1}{x_1x_2^2x_3^2} - \frac{1}{397.5} \le 0,$$
$$g3 : \frac{x_4^3}{x_2x_3x_6^4} - \frac{1}{1.93} \le 0,\ g4 : \frac{x_5^3}{x_2x_3x_7^4} - \frac{1}{1.93} \le 0, g5 : x_2x_3 - 40 \le 0, g6 : \frac{x_1}{x_2} - 12 \le 0,$$
$$g7 : 5 - \frac{x_1}{x_2} \le 0, g8 : 1.9 - x_4 + 1.5x_6 \le 0, g9 : 1.9 - x_5 + 1.1x_7 \le 0, g10 : f_1(x) \le 1300,$$
$$g11 : \frac{\sqrt{(\frac{745x_5}{x_2x_3})^2 + 1.575 \times 10^8}}{0.1x_7^3} \le 1100, 2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8,$$
$$17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5 \le x_7 \le 5.5.$$

$$G02.\,\mathrm{Min}\,\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})),$$
s.t.,
$$f_1(\mathbf{x}) = \sum_{i=1}^{n} sin(\pi x_i), f_2(\mathbf{x}) = \sum_{i=1}^{n} cos(\pi x_i),$$
$$n = 10, 0 \le x_i \le 6, 1 \le i \le n.$$

References

1. Zitzler E, Thiele L (1999) Evolution algorithms for multiobjective optimization: methods and application. Doctoral thesis ETH No. 133398. Swiss Federal Institute of Technology, Zurich
2. Leung YW, Wang YP (2000) Multiobjective programming using uniform design and genetic algorithm. IEEE Trans Syst Man Cybern C Appl Rev 30(3):293–304
3. Deb K, Agrawal S, Pratap A et al (2002) A fast elitist nondominated sorting genetic algorithm for multi-objective optimization: NSGA-II. IEEE Trans Evol Comput 6:182–197
4. Aguirre AH, Rionda SB, Coello CA Coello et al (2004) Handling constraints using multiobjective optimization concepts. Int J Numer Methods Eng 59(15):1989–2017
5. Arumugama MS, Rao MVC, Palaniappan R (2005) New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems. Appl Soft Comput 6(1):38–52
6. Kaya M (2009) MOGAMOD: multi-objective genetic algorithm for motif discovery. Expert Syst Appl 36(2):1039–1047
7. Deb K, Tiwari S (2005) Omni-optimizer: A procedure for single and multi-objective optimization. In: Coello Coello CA , Aguirre AH, Zitzler E (eds) Proceedings of 3rd. EMO, LNCS , Mexico, vol 3410, pp 47–61
8. Klanac A, Jelovica J (2007) A concept of omni-optimization for ship structural design. Advancements in Marine Structures.In: Guedes Soares and Das (eds) Proceedings of MARSTRUCT 2007, the 1st international conference on marine structures, glasgow, pp 473–481
9. Klanac A, Jelovica J, Niemelänen M et al. Structural omni-optimization of a tanker. COMPIT'2008, Liege
10. Palonen M, Hasan A, Siren K (2009) A genetic algorithm for optimization of building envelope and HVAC system parameters. Eleventh international IBPSA conference
11. Timmis J, Knight T, De Castro LN et al An overview of artificial immune systems. In: Paton R, Bolouri H, Holcombe M (eds) Computation in cells and tissues: perspectives and tools for thought, natural computation series.. pp 51–86 Springer, Berlin, (2004)
12. Huang XY, Zhang ZH, He CJ et al (2005) Mordern intelligent algorithms: theory and applications. Science Press, Beijing
13. Campelo F, Guimarães FG, Igarashi H (2007) Overview of artificial immune systems for multi-objective optimization. Proceedings of the 4th international conference on evolutionary multi-criterion optimization, Matsushima, pp 937–951
14. Hart E, Timmis J (2008) Application areas of AIS: the past, present and the puture. J Appl Soft Comput 8(1):191–201
15. Timmis J, Honec A, Stibord T et al (2008) Theoretical advances in artificial immune systems. Theor Comput Sci 403(1):11–32
16. Jiao LC, Du HF, Gong MG (2006) Immune computing on optimization: learning and recognition. Science press, Beijing
17. Su ZG, Wang PH, Yu XJ (2010) Immune genetic algorithm-based adaptive evidential model for estimating unmeasured parameter: estimating levels of coal powder filling in ball mill. Exp Syst Appl 37(7):5246–5258
18. Aragón VS, Esquivel SC (2008) Optimizing constrained problems through a T-cell artificial immune system. J Comput Sci Technol 8(3):158–165
19. Cao XB, Qiao H, Xua YW (2007) Negative selection based immune optimization. Adv Eng Softw 38(10):649–656
20. Woldemariam KM, Yen GG (2010) Vaccine enhanced artificial immune system for multimodal function optimization. IEEE Trans Syst Man Cybern B Cybern 40(1):218–228
21. de Castro LN, Von Zuben FJ (2000) The clonal selection algorithm with engineering applications. In: workshop proceedings of GECCO, workshop on artificial immune systems and their applications, Las Vegas, pp 36–37
22. de Castro LN, Timmis J (2002) Artificial immune systems: a new computational intelligence approach. Springer, Berlin
23. Cutello V, Narzisi,G Nicosia G et al. (2006) Real coded clonal selection algorithm for global numerical optimization using a new inversely proportional hypermutation operator. The 21st annual ACM symposium on applied computing, SAC 2006. ACM Press 2, Dijon, pp 950–954
24. Coello CA ~ Coello (2005) Solving multiobjective optimization problems using an aritificial immune system. Genetic programming and evolvable machine, pp. 163–190
25. Brownlee J (2006) IIDLE: an immunological inspired distributed learning environment for multiple objective and hybrid optimisation. 2006 IEEE congress on evolutionary computation, sheraton vancouver wall centre hotel, Vancouver
26. Freschi F, Coello Coello CA, Repetto M Multiobjective optimization and artificial immune system: A review. http://www.igi-global.com/downloads/excerpts/33155.pdf
27. Campelo F, Guimaraes FG, Igarashi H (2007) Overview of artificial immune systems for multi-objective optimization. In: Obayashi S et al (eds) EMO 2007, LNCS 4403, pp 937–951
28. Gong MG, Jiao LC, Du HF et al (2008) Multiobjective immune algorithm with nondominated neighbor-based selection. Evol Comput 16(2):225–255
29. Omkar SN, Khandelwal R, Yathindra S et al (2008) Artificial immune system for multi-objective design optimization of composite structures. Eng Appl Artif Intell 21(8):1416–1429
30. Tan KC, Goh CK, Mamun AA et al (2008) An evolutionary artificial immune system for multi-objective optimization. Eur J Oper Res 187(2):371–392
31. Zhang ZH (2007) Immune optimization algorithm for constrained nonlinear multiobjective optimization problems. Appl Soft Comput 7(3):840–857
32. Xiao HS, Zu JA (2007) A new constrained multiobjective optimization algorithm based on artificial immune systems. 2007 international conference on mechatronics and automation, Harbin, pp 3122–3127
33. Hong L (2009) An adaptive multi-objective immune optimization algorithm. 2009 IITA international conference on control, automation and systems engineering, pp 140–143
34. Aydin I, Karakose M, Akin E A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Appl Soft Comput (Available online)
35. Chen JY, Lin QZ, Ji Z (2010) A hybrid immune multiobjective optimization algorithm. Eur J Oper Res 204(2):294–302
36. Hu ZH (2010) A multiobjective immune algorithm based on a multiple-affinity model. Eur J Oper Res 202(1):60–72
37. Gao JQ, Wang J (2010) WBMOAIS: a novel artificial immune system for multiobjective optimization. Comput Oper Res 37(1):50–61
38. de Castro LN, Von Zuben FJ aiNet: an artificial immune network for data analysis. In: Abbass HA, Sarker RA, Newton CS (eds) Data mining: a heuristic approach, Chapter XII.. pp 231–259 Idea Group Publishing, USA, (2001)

39. de Castro LN, Timmis J (2002) An artificial immune network for multimodal function optimization. Proceedings of IEEE world congress on evolutionary computation. pp 669–674
40. Gomes LCT, de Sousa JS, Bezerra GB et al (2003) dopt-aiNet and the gene ordering problem. In: Information technology magazine 3(2):27–33
41. Coelho GP, Von Zuben FJ (2006) omni-aiNet: an immune-inspired approach for omni-optimization. In: Bersini H, Carneiro J (eds) ICARIS 2006, LNCS 4163, pp 294–308
42. Oyama A, Shimoyama K, Fujii K et al (2005) New constraint-handling method for multiobjective multiconstraint evo-lutionary optimization and its application to space plane design. In: Schilling R, Haase W, Periaux J (eds) Evolutionary and deterministic methods for design, optimization and control with applications to industrial and societal problems (Eurogen 2005), pp 1–13 Germany
43. Van Veldhuizen DA (1999) Multi-objective evolutionary algorithms: classifications, analysis, and new innovation. Elec Comput Eng Air Force Institute of Technology, Wright-Patterson AFB, Dayton
44. Mallipeddi R, Suganthan PN (2006) Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-rarameter optimization. Kanpur genetic algorithms laboratory (KanGAL), Indian Institute of Technology, Kanpur, PIN 208 016, India, Technical report
45. Deb K, Pratap A, Meyarivan T (2001) Constrained test problems for multi-objective evolutionary optimization. First international conference on evolutionary multi-criterion optimization. In: Zitzler Eckart E, Deb K, Lothar TL et al (eds) Lecture notes in computer science no. 1993, Springer pp 284–298
46. Kurpati A, Azarm S, Wu J (2002) Constraint handling improvements for multiobjective genetic algorithms. Struct Multidisc Optim 23:204–213