I.J. Intelligent Systems and Applications, **2012, 6, 1-13**
Published Online June 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2012.06.01

A General Framework for Multi-Objective Optimization Immune Algorithms

Chen Yunfang
College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China
Email:chenyunfang999@gmail.com

***Abstract*—**Artificial Immune System (AIS) is a hotspot in the area of Computational Intelligence. While the Multi-Objective Optimization (MOP) problem is one of the most widely applied NP-Complete problems. During the past decade more than ten kinds of Multi-Objective optimization algorithms based on AIS were proposed and showed outstanding abilities in solving this kind of problem. The paper presents a general framework of Multi-Objective Immune Algorithms, which summarizes a uniform outline of this kind of algorithms and gives a description of its principles, mainly used operators and processing methods. Then we implement the proposed framework and build four typical immune algorithms on it: CLONALG, MISA, NNIA and CMOIA. The experiment results showed the framework is very suitable to develop the various multi-objective optimization immune algorithms.

***Index Terms*—**Multi-Objective Optimization, Artificial Immune Systems, Algorithms Framework.

I. Introduction

Many problems in the engineering field or scientific research have more than one objective to be optimized simultaneously. This is Multi-Objective Optimization Problems (MOP). Different from optimization problems which only have a single objective, there is usually no unique solution that meets all objectives of the problem. Instead, it aims at finding all the solutions which could build a balance of distance among every objective. This character of MOP Problems greatly increases the searching difficulty at the time we try to find solutions. Early in 1987, Serafini [1] proposed that the computational complexity of these kinds of problems was NP-Complete. Therefore, new high-efficiency heuristic algorithms are needed to solve it.

Several methods have been proposed to solve this kind of problem, such as genetic algorithms, Particle Swarm Optimization, Multi-Agent System algorithm and so on.

Genetic Algorithms (GA) are based on the principle of evolution and natural genetics. They simulate a natural process of creatures` evolution and natural selection, applied into data searching field. To solve MOP Problems, GA treats the solution-set of MOP as a creature population, the objectives as a natural condition, and dominated solutions as elite ones in the population. So solving process of MOP can be linked with the natural selection of creature populations. The main algorithms based on GA are NSGA [2] (NSGAII is its update version), MOEAD [3], PAES [4] and others.

Particle Swarm Optimization (PSO) is one of the classical swarm intelligence algorithms that simulate social behaviors to guide swarms of particles towards the most promising regions of the search space [5]. PSO has a flexible and well-balanced method to increase and adapt global exploration and local search abilities.

Multi-Agent System (MAS) is a system made up of multiple interacting intelligent agents. MAS can be used to solve problems that are difficult or impossible for a single agent or massive system to solve. When applied to MOP, MAS forms a set of individual agents with their own goals that can interact with each other. During the algorithm process, agents changed towards their goals and exchange information with other agents, and finally solve the problems, as with FGAS algorithms in paper [5].

The heuristic algorithms mentioned above are all inspired by some natural phenomena. Furthermore in 1986, J. Doyne Farmer and other four researchers [6] published their work on a new biologically-inspired computing model: Artificial Immune System (AIS). Rooted in animal`s immune system, AIS makes an abstraction of several principles: Negative Selection, Clonal Selection, Immune Networks etc. AIS can be used in computer security, machine learning, pattern recognition and many other areas [7].

Today in the area of MOP, two principles of AIS are widely used: Clonal Selection and Immune Networks.

MOP algorithm based on immune network was first proposed by Jerne [8], which is based on a bio-immune phenomenon where in B-cells are stimulated and suppressed by both antigens and other interacted B-cells. Based on Jerne's study, two sub-classes of immune network theories were proposed: De Castro's discrete immune network model [2] and Hajela's model [9].

Recently, many algorithms based in area have been proposed. The amount of work needed for developing, implementing, testing or doing comparison on Multi-Objective immune algorithms increases greatly. So a common algorithm framework which could cover most of these kinds of algorithms is needed, just like JMetal [10] for GA, which is an object-oriented Java-based framework aimed at the study of meta-heuristics for solving MOP problems. Researchers could conveniently do the development, experimentation and comparison in area of GA-MOP algorithms. But this kind of framework is still lacking in MO-AIS area.

Based on the wide research on typical kinds of MO-AIS algorithms and the high-level abstraction of them, this paper proposes a general framework of Multi-Objective Immune Algorithms (mainly based on the clonal selection principle), which gives a description of its mathematical foundation, principles, algorithmic outline and the common implementation of some commonly used operators and methods. Based on this, we implement four classical MO-AIS algorithms, CLONALG [11], NNIA [12], MISA [13] and CMOIA [14], to show the validity of this framework.

II. Background Knowledge And Definition

2.1 Multi-Objective Optimization Problem

Multi-objective Optimization aims at doing optimization for a serial of functions like:

$$F(x)=(f_1(x), f_2(x),....,f_k(x))^T \quad (1)$$

And $x=(x_1, x_2, ... ,x_m)\in\Omega$, Where x is called the decision variable, and Ω is the feasible region in decision space (or the definition domain of functions).

Without the loss of generality, we take a maximization problem as an example. It aims to achieving maximization for every objective. By definition a decision variable $x_A\in\Omega$ dominates another variable $x_B \in\Omega$ (written as $x_A \succ x_B$) if and only if:

$$\forall i=1,2,...,k,\ f_i(x_A)\geq f_i(x_B)\wedge\exists j=1,2,...,k$$

$$f_i(x_A)>f_i(x_B) \quad (2)$$

That is, one solution is Pareto dominant over another one, only when all the sub-objective values of the first solution are no less than the corresponding sub-objective values of the other solution, and there must exist at least one sub-objective on which the first solution exceeds the second one.

Then we define a decision variable $x^* \in\Omega$ that is a Pareto optimal solution or nondominated solution if:

$$\forall x\in\Omega,\ x^*\succ x$$

Then the Pareto-optimal set is defined as:

$$P^*\triangleq\{x^*\in\Omega\,|\,\neg\exists x\in\Omega, x\succ x^*\} \quad (3)$$

The Pareto-optimal set is the set of all Pareto-optimal solutions. And the corresponding image of the Pareto-optimal set under the objective function space is called the Pareto-optimal front:

$$PF^*\triangleq\{F(x^*)=(f_1(x^*), f_2(x^*),..., f_k(x^*))^T\,|\,x^*\in P^*\} \quad (4)$$

Namely, the solution within the set is considered as optimal only when there are no solutions existing in the set and superior to it, and this set is also named, Pareto front.

Solving a Multi-Objective optimization problem is to find out a suitable pareto- optimal front for it.

2.2 Artificial Immune System

Artificial Immune Systems are developed from biological immune systems. They are heuristic computing systems aimed at solving many kinds of difficult real problems based on the principles, functions, and basic characteristics of bio-immune systems. The main purpose of this research is to go deeply into information processing methods inspired by the biological immune system, and then to work out some new algorithms and engineering models to solve the difficult problems we face reality.

As mentioned above, there are three main principles abstracted from the bio-immune system:

The principle of Negative-Selection is inspired by the phenomena of the thymus that produces a set of mature T-cells which only have the ability to identify non-self-antigens. Negative-Selection principle was first applied by Forrest et al. to detect counterfeit data caused by viruses in computer systems [15]. The main point is to produce a set of self-strings S, which represents the features of the system's normal state. Then generate a set of detectors D, that only recognize the complementary to S. When new data is coming in, these detectors can classify them as being self or non-self. Later, this principle is widely applied in the area of intrusion detection, pattern recognition and etc.

The clonal selection principle got a big breakthrough in the area of computational optimization and pattern recognition by AIS. In particular, abstracted from the bio-immune system, the maturation process of B-cells is decided by its affinity to an antigen, with its associated hyper-mutation process. And it also maintains the idea of using memory cells to retain good solutions to the problem being solved. There are two important features of this process in B-cells that can be exploited for computational algorithms. The first one is that the self-breeding of B-cells is linear to their affinity to the bind antigen, so the higher the affinity, the more clones will be produced. Secondly, when the antibody of a B-cell undergoes the mutation progress, the rate of mutation is in inverse proportion to the affinity of it. Based on these two features, de Castro and Von Zuben proposed one of the most popular clonal selection algorithm, called CLONALG[11], which has been used widely in pattern matching and single object optimization problems.

Jerne [8] proposed the immune network principle based on some of the observed phenomena of the bio-immune system, for example, learning and memory. The precondition of this theory is that any lymphocyte receptor of an organism can be recognized by a sub-set of the total receptor repertoire. The receptors of this sub-set have their own recognizing set and so on, so an interactive immune network is formed. Immune networks are often considered as self-inspired networks. For lack of foreign antigens, Jerne supposed that the immune system must make a behavior or activity inspired from interactions with it-self or from these interactions behavior such as tolerance and memory emerge.

Before 2002, researchers developed two basic immune optimization algorithms: the CLONALG algorithm based on clonal selection principle and the aiNet algorithm based on immune network principle. Subsequently, based on the foundation of CLONALG, Garrett [16] developed Adaptive Clonal Selection (ACS) by improving a number of processes and parameters in it. Research based on aiNet extended aiNet to opt-aiNet. Then Freschi and Repetto developed a multi-objective version of the opt-aiNet called Vector Artificial Immune System (VAIS) [18].

After Luh et al. [19] proposed a Multi-objective Immune Algorithm (MOIA), MO-AIS research grew into maturity. Following that tens of algorithms were proposed. The typical ones are Wong et al. `s HAIS [20], and Gong et al.`s NNIA [12].

III. The Algorithm Framework

Abstracting from the information above, the logical mapping between immune system (AIS) and multi-objective optimization problems (MOP) is listed inTable1:

Table 1: Mapping Between Biological Immune System and Multi-Objective Optimization Problem

AIS	MOP
Antigen	multi-objective problem
antibody, B-Cell, T-Cell	Solution
memory cell	Archive
immune clone	clonal selection algorithm
Affinity	quality of solution
antibody generation	Iteration

Antibodies always try to best recognize antigens in biological immune procedures. This is very similar to the evaluation of MOPs. Consequently antigens can be seen as multi-objective problems, Antibodies are treated as solutions to that problem and antigen-antibody pair affinity can be seen as the quality of the solution to the problem. After relation mappings like this, the biological immune mechanism is brought into the field of multi-objective optimization.

Several main parts were included in a typical MO-AIS algorithm. When we try to solve a problem using an AIS algorithm, the steps in Fig.1 are that we should follow:

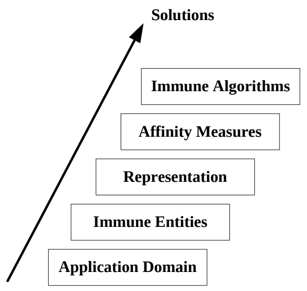


Figure 1. The Steps to Implements an MO-AIS

Then we could define an individual process in MO-AIS algorithms as an operator, which means a sub-process with a set of comparatively independent algorithms to accomplish a part of the function in a whole MO-AIS algorithm. The operators could be classified to some kinds according to their functions and have many different implementations.

Finally, from the definition of AIS and MOP problems, we could draw an abstract graph for the algorithms` general framework. That is to say, the common outline which most Multi-Objective immune algorithms (mainly based on the clonal selection principle) obey.

In a word, this kind of algorithm commonly can be divided into 3 parts: the initial of population, the iteration of the immune progress and the final process or output of the solution-set. The first part generates the initial solutions and puts them into the population achieve, and then initializes all the parameters and containers of the algorithm.

The second part (the immune progress`s iteration) is commonly divided into 4 steps: selection of elite solutions, copying selected solutions, mutation or crossover the solutions, the recombining and processing of the memory or solution-set. And before them it will commonly do an evaluation of objective values and constraints. In this part, the biological immune progress is applied to the solutions, so the solutions in the population could have a trend of better affinity as a whole.

After the stop condition of the second part occurs, the third part happens. This part will perform a final process on the solution-set and output it.

Based on the information mentioned above, the common outline of Multi-Objective immune algorithms is listed below:

Table 2: The Common Outline of MO-AIS

```
1.  Define the Algorithm parameters. And Initialize the population
    P with generated antibodies.
2.      While(Stop Condition of Iteration not reached)
        a) Calculate the objective values and constraints of all
           solutions (called "Evaluation").
        b) Perform selection on population P, the solutions selected
           forms S(t).
        c) Clone S (t) to form a copy C (t).
        d) Mutate or do crossover on C (t) to get A (t).
        e) Recombinant on P, A (t) and so on or put elite solutions
           into memory achieve.
3.      Get Pareto-front from M (t).
```

3.1 Initial of Population

In this part, two task are accomplished: 1) defining the algorithm parameters, such as the stop condition of part 2, the size of the population (usually the same as the output solutions' number), and others. And 2) filling the initial population with generated antibodies.

Mostly, we initialize the population by randomly generating some antibodies, but in some cases we generate the antibodies according to grids of objective space or a problem-specific method.

Initializing the population using the second method usually brings faster convergence or improvement of search capabilities. For example, some algorithms may initialize the population by dividing decision variable space into a certain number of parts relative to the desired population size. So they generate an initial population of a uniform distribution of solutions, making the searching result more diverse.

3.2 Immune Progress Iteration

This part is to describe the immune search process formally. It is usually treated as an iteration which could be divided into four steps, which is summarized by 3 operators.

The first two steps are the selection of elite antibodies and the immune clone process of them. As they are always one after another, we usually combine them into one operator: clonal selection. The population going through this operator will have the antibodies with lower affinity enhanced and the antibodies with higher affinity inspired. This operator is one of the most important operators that makes the population to convergence side in this kind of algorithm.

After that it performs the step of variation of antibodies` information. This step aims at disturbing the solutions, whether elite ones or not, to make them drift off the original position, and then to explore the new solutions unreached before. Many operators can be applied to this step, even the operators genetic algorithms used, such as crossover operator. And based on the genetic algorithms` operator mutation, MO-AIS could use its unique operator to process this step: hyper-mutation.

At the end of iteration, the processing of the population happens. Some algorithms may apply an operator called recombination, which means recombining the solution-sets previously steps produced. Recombination operator may increase the diversity of the population, and after that nearly all algorithms will form a new population with the ones this iteration produced, as the initial population of the next one.

3.3 The Final Process of the Antibodies

Different algorithms have different way to store and output antibodies. Some of them store antibodies into memory, and before the elite antibodies added, they will do a check of the antibody`s fitness or other attributes, to insure that antibodies in memory will be more dominated as a whole.

Memory mechanism is one of the unique mechanisms AIS has, it plays an important role in maintaining the quality of final output solution-set. In some algorithms it is even able to affect the immune operator's performance. But some immune optimization algorithms do not use memory to store antibodies, they usually form an individual population rarely directly undergoing the immune iterations. This population absorbs its members by some condition usually when iteration ends. In the end its contents may be regarded as the final Pareto-Front.

IV. The Immune Iterator Operators

4.1 The Clonal selection Operators

As mentioned above, the selection and clone of elite antibodies is a core process of MO-AIS. As we know, this operator generally can be divided into two parts: first is selection, which is to select dominant antibodies from the initial population, This part is processed mainly by sorting the solutions in the current generation with the standard algorithms defined. And then take the cloning procedure: copying the selected antibodies by a certain time which is defined by algorithms. Usually the copied antibodies will fill the population outputted. Here is the common outline of this operator:

Table 3: The Common Outline of Clonal Selection Operator

Input:	input population **P** , selecting rate **Rs** , selecting sort standard **Ds** , clone rate **Rc** , clone times definition **Dc**
Select:	1) Sorting **P** by **Ds** , forms **P'** 2) If the size of **P' > Rs**, Taking the first **Rs** of the **P'** , put them into a new population **Pc**, otherwise, let **Pc=P'**
Clone:	3) **For each** antibody **A*i* in Pc** : a) Copy **A*i*** by times of **Dc*Rc** b) Put the copied antibodies into **Pc**
Output:	**Pc**

Different algorithms usually have different **Ds** and **Dc**, and **Rs Rc** usually could be defined by user before algorithms executed. Now we give out some different clonal selection operators` definition:

1) Affinity Rank Selection

This method is a widely used selection method; firstly we must define the measure of affinity, a simple way is to define it by the sum of Euclidean-distance between current antibody and best antibody in this iteration and previous iteration (Wang X. L. [21] et al.):

$$Aff_i = |x_i - x_c| + |x_i - x_g| \tag{5}$$

X_C, X_g are the best antibodies in this and the previous generation. So **Ds** is: the greater **Aff*i*** is, the lower the antibodies ranked, as the increasing order of **Aff*i*** .

2) Non-dominated Neighbor-based Selection

This selection method was first brought by Gong`s NNIA in 2010, it measures the selection scale by a value called crowding-distance [2]:

$$\zeta(d,D) \triangleq \sum_{i=1}^{k} \frac{\zeta_i(d,D)}{f_i^{\max} - f_i^{\min}} \tag{6}$$

$f_i^{\max}$ and $f_i^{\min}$ are the maximum value and the minimum value of the i-th objective and $\zeta_i(d,D)$=

$$\begin{cases} \infty, \text{ if } f_i(d)=\min\{f_i(d')|d'\in D\} \text{ or } f_i(d)=\max\{f_i(d')|d'\in D\} \\ \min\{f_i(d')-f_i(d'')|d',d''\in D: f_i(d'')<f_i(d)<f_i(d')\}, \text{otherwise} \end{cases} \tag{7}$$

So **Ds** is: if **ζ(d,D)** of an antibody is greater, it ranks higher, as the descending order by **ζ(d,D)**.

3) Proportional Cloning

This clone method is also based on the crowding-distance definition the copy times of an antibody $\mathbf{a}_i$ as:

$$q_i = \left\lceil n_c \times \frac{\zeta(a_i,\mathrm{A})}{\sum_{j=1}^{|\mathrm{A}|}\zeta(a_j,\mathrm{A})} \right\rceil \tag{8}$$

Where $\mathbf{n_c}$ is **Rc** in this paper. So **Dc** is: for each antibody, copy it $\mathbf{q}_i$ times.

4) Clustering-Based Clone

This clone definition is newly proposed in algorithm CMOIA [22], it uses a k-means clustering method to cluster the population before cloning it. When cloning starts, **Dc** is: if the antibody to be copied belongs to the cluster which has the most antibodies among all clusters, don't copy it; if the antibody belongs to the cluster whose number of antibodies is greater than the average of all, copy it 1/2***Rc** times, otherwise copy it 2***Rc** times. This method is proved that could greatly increase the diversity of the population.

4.2 Hyper-Mutation and Crossover Operators

To explore the formerly unsearched area of the objective space, a procedure perturbing antibodies' information is needed. In MO-AIS algorithms, many types of methods could be used, such as GA`s mutation, GA`s crossover and the AIS`s unique operator, hyper-mutation.

The immune cells` frequently mutation plays an important role in the human`s immune systems, and also it does in AIS: first it ensures the diversity of the populations; secondly, combined with the clonal selection operators, it will greatly increase the affinity of the population as a whole.

Based on the bio-immune mutation procedure and inspired by the GA`s mutation, the hyper-mutation operator was proposed. This kind of operator's effect on the antibodies' content information, makes perturbation on the solutions. So the MO-AIS procedure will have the ability of exploring previously unreached solutions.

Different from GA`s mutation, hyper-mutation uses the strategy that the mutation rate for each antibody is different, usually as the increasing order of the affinity of it. So the optimal solutions won't probablely get lost and the undominanted solutions will change rapidly during the procedure.

The common outline of this kind of operator:

Table 4: The Common Outline of Hyper-Mutation Operator

Input:	input population **P** , base mutation rate **Rb** , mutation probablity **Rm** t∈[0,1] , affinity measure **Ma**
	For each antibody **A*i* in P** 1) Calculate the affinity **aff*i*** of **A*i*** by **Ma** 2) Generate a random float **f**∈[0,1] 3) **If (** f∈[0,**Rm**]) **Then For each** decision variable **X*i* in A*i*** a) Generate a random float **u** ∈ [-**Rb**,**Rb**] b) Calculate **s**=**u*****aff*i*** c) Let **X*i*** += **s** **Else** do nothing 4) Put **A*i*** to **Pm**
Output:	**Pm**

Different from the clonal selection type, the hyper-mutation operators' MO-AIS algorithms used mostly obey the same strategy; the only change between them usually is the measure method of affinity **Ma**. So we consider all these kinds of operators as one.

Besides hyper-mutation operators, many kinds of operators could be applied to this procedure. The classical GA`s mutation performs perturbation better than hyper-mutation and is widely used in MO-AIS algorithms. We also often apply GA`s crossover operators on this, though it does not match AIS`s biological essence. The results prove that those kind of operators sometimes bring a good diversity or congresses to MO-AIS.

4.3 Recombination and Memory Operators

After the previous two steps, some operators, such as clonal selection, GA crossover, hyper-mutation, etc., are applied on populations. Each operator will output a set of antibodies with different characteristics, so have a combination of them maybe good for the result`s quality. In bio-immune systems, genetic recombination is a process by which a molecule of DNA is broken and then joined to a different one. Inspired by this, some algorithms` researchers work out a kind of operator called Recombination and apply it to the antibodies sets derived from previous steps.

To simplify, the recombination operator could be represented by a set of other operators. If we suppose the two set to be recombined is **C** = ($c_1, c_2, c_3, \ldots, c_i$) and **A** = ($a_1, a_2, a_3, \ldots, a_j$), then the recombination $\mathbf{T}^{\mathbf{R}}$ is:

$\mathbf{T}^{\mathbf{R}}$(**C**, **A**)=**Recombine**(c_{i1}, a_{j1})+ **Recombine**(c_{i2}, a_{j2})+ **Recombine**(c_{i3}, a_{j3})+…..+ **Recombine**(c_{im}, a_{jn})

The function **Recombine** is the recombination method of two antibodies. Usually it is a crossover operator. The two parameters of **Recombine** are the two anti-bodies selected from **A** and **C**. The selection principles maybe ordered by choosing from one set and equal-probably random choosing from another and so on.

Some algorithms will run this operator after all other steps of immune iteration are done, but some may apply it before hyper-mutation an etc. Then the methods of final processing of populations gotten from previous steps in one iteration are different. In this step, the important strategy of memory is widely used.

In the bio-immune system, the memory cells bring support the secondary immunity process. They remember the anti-bodies with high affinity to specific antigens, and when those antigens come again, immune system will quickly generate pertinent anti-bodies by the information stored in memory cells. Inspired by this, MO-AIS algorithms use memory operator to store elite solutions. Unlike normal operators, the memory operator is mainly a place to store solutions. The main operating process is when a new solution inserted. Below is the outline for a solution inserting into memory:

Table 5: The Common Outline of Memory Inserting

```
Input:    input solution S
Inside:   memory Population P (with limited capacity), antibody
          checking standard Cs
1)  If( P is not full )
        Insert S into P
    else
        a)  Checking S with Cs in P
        b)  If( S dominates in P)
            i)  Delete the least dominated
                antibody in P
            ii) Insert S into P
Output:   None
```

The checking standard **Cs** decides whether **S** could be inserted into memory, depending on the situation of both **P** and **S**. Different memory operators are mainly distinguished by **Cs**. Here are two examples of memory operator implementation.

1) Affinity Memory

This is a common version of a memory operator. It only measures the anti-bodies by their affinity of. If the antibody to be inserted, S has higher affinity than the worst one in the memory, S should replace it.

2) Adaptive Grid Memory

This is a kind of memory that has an antibody storage with an adaptive grid character [23]. When the coming anti-bodies belong to a crowded grid of objective space, it will not be inserted. This strategy is able to maintain a full diversity of anti-bodies inside memory.

At the last of the iteration, some algorithms will add elite solutions into memory while others will recombine several parts of the solution-set into a new population. Even some of them will re-generate some new solutions into the population.

V. Algorithm's Implement

To fully prove the feasibility and usefulness of the framework mentioned above, we implement four typical MO-AIS algorithms based on the framework: CLONALG NNIA, MISA and CMOIA.

5.1 CLONALG: A SOP Algorithm but the MOP Algorithms` Basement

Early in this paper, we have introduced de Castro`s CLONALG [11], which is the first algorithms that applied the clonal selection principle to pattern matching and optimization areas. Though this algorithm`s procedure is too simple to solve complex Multi-Objective optimization problems, it is usually used in to solve single object ones. It is still the foundation of many Multi-Objective immune algorithms.

Under this framework, CLONALG mainly use the clonal selection operator (in the model of doing clone shortly after the selection ended) and hyper-mutation operator, then the main steps in the immune iteration process is listed below:

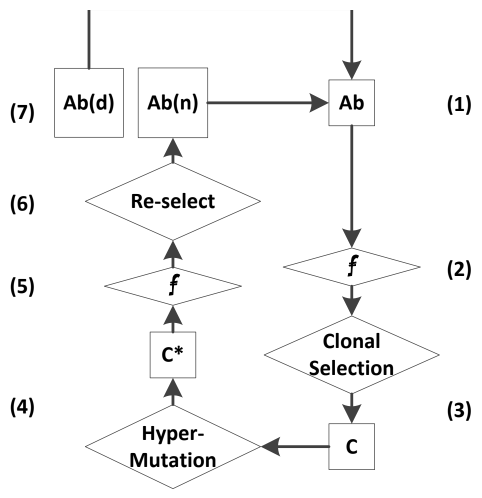


Figure 2. Computational Procedure for CLONALG

In this diagram, (1)**Ab** is the initial population after one iteration, and in (2)(5) we calculate the objective values and constraint values of anti-bodies, and update their fitness value.

Then in (3), we apply the clonal selection operator, and set **Ds** defined by the Hamming-Distance [24] measure of the function defining space. And **Dc** is:

$$D_c = \sum_{i=1}^{N} round(\beta \bullet N) \tag{9}$$

In this case , β is the affinity measure the same as **Ds**. Other parameters could be defined by users. It also uses the standard hyper-mutation operator mentioned above and set **Ma**=**Ds**.

No memory operator is applied to CLONALG, and in steps (6) (7), it re-selects **C*,** usually with a higher **Rs** than the first selection to form **Ab(n)**. Then it re-generates some new random anti-bodies **Ab(d)** to make a complement of re-selection`s washing out.

5.2 MISA: An Adaptive Grid improved Algorithm

The outstanding performance on the diversity of solutions is a big advantage for which MO-AIS is superior to other algorithms. Different methods are used to maintain a good balance between convergence and diversity in solving progress. MISA is a success of this. This algorithm applies an adaptive grid to the memory store of solutions, break the objective space into several partitions, and measure the solution number in each partition. The anti-bodies which belong to the much crowding partitions will be suppressed to some extent. And MISA is an algorithm with much more complex than CLONALG, so it could be applied on MOP problems.

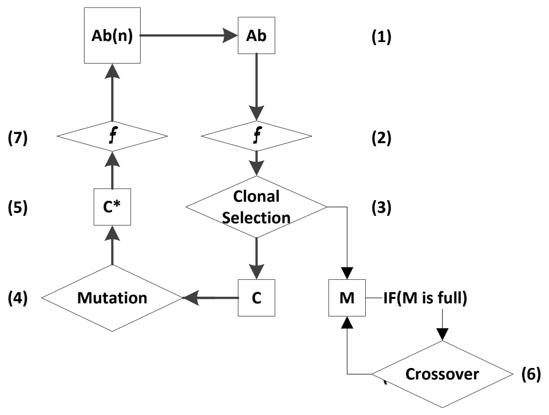


Figure 3. Computational Procedure for MISA

Under this framework, **M** is a memory store called secondary memory; it has an adaptive grid inside to store solutions. In the clonal selection operator, **Ds** is the common method to measure affinity while **Dc** is decided by the grid in **M** to which a solution belongs. If the solution to be copied belongs to the most crowded grid in **M**, it will be copied as zero. If while it belongs to the grid with an number that is above the average of solutions, it will be copied **Rc** times. Otherwise it will be copied 2* **Rc** times.

If **M** is full, a common crossover operator will be applied to it between steps (5) and (7). That is to force the elite anti-bodies in memory to exchange their information.

In (4), MISA will perform both a uniform and non-uniform mutation on populations, and in the non-uniform mutation, the **Rb** will decrease over time from 0.9 to 0.3.

MISA neatly solves all kinds of MOP problems with good convergence. It is a typical example of AIS for MOP problems.

5.3 NNIA: Algorithmic Solutions Weighted by Crowding-Distance

Many approaches could be taken to improve the solution quality of MO-AIS algorithms. The algorithm proposed by M. G Gong-NNIA brought the measure of Crowding-Distance into MO-AIS research. This measure acts is important in NNIA from affinity-weighed to clone-rate calculation.

Crowding-Distance is a measure of how close a solution is to its neighbors in the same Pareto-front. Large average Crowding-Distance represents more diverse population. The immune iteration outline is like this:

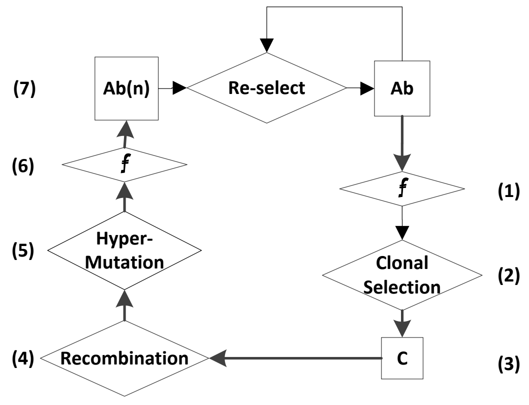


Figure 4. Computational Procedure for NNIA

In step (2), NNIA uses crowding-distance selection and proportional clone operators to form the clonal selection. The two parts of the operator are both mentioned above and both measured by Crowding-Distance.

In step (4), it performs a recombination between selection result and clone result in (2) using the crossover method. As our recombination operator is defined, it picks up an antibody in order from the clone result and randomly takes an antibody from the selection result, then does crossover on each pair to form a new population. Then a normal hyper-mutation is applied to the population.

Finally in (7), the previous iteration`s result will be combined with this iteration`s result, then a crowding-distance selection is applied to them to select the next generation`s initial population.

NNIA is highly efficient at solving MOP problems. Experiments show it could produce diverse solution-sets within many fewer generations than MISA. But some versions of its implementation may take a long time to finish the calculation of one generation.

5.4 CMOIA: The Clustering-Based Clone Algorithm

As we mentioned above, MISA uses adaptive grid memory to force the solution-set to maintain diversity while NNIA tries to use a new affinity measurement to improve the quality of the results. Not long ago a new method of keeping balance between the diversity and convergence of MO-AIS was proposed. CMOIA uses the K-means clustering on the clone operator to maintain the balance of local searching and widely exploring the objective space to achieve a good result for some MOP problems.

Using the clustering clone operator mentioned above, CMOIA divided the objective space into several clusters by the K-Means Clustering algorithm. When cloning happened, solutions that belong to crowded clusters will be copied less and vice versa. So the space which is not fully developed will have its search power appended. Using this strategy on normal MO-AIS algorithms, CMOIA`s outline is like this:

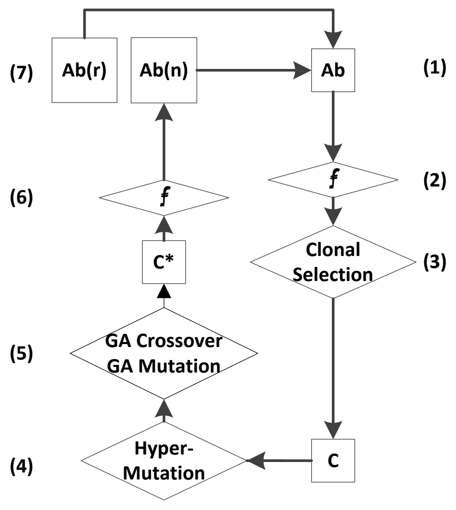


Figure 5. Computational Procedure for CMOIA

In step (3), CMOIA applies the normal affinity selection and its clustering clone operator on the clonal selection process. Step (4) is the normal hyper-mutation operator while after that GA`s mutation and crossover operator are applied on the population to accelerate the local search around less crowded clusters. At last the result from step (3),(5) and some randomly generated solutions are put together to form the next generation's initial population.

Experiments show that in many MOP problems CMOIA has unique advantages in convergence, diversity and distribution uniformity than most other MOP algorithms.

VI. The Experiment and Conclusion

We implement the four algorithms under our proposed algorithmic framework. To prove the feasibility and effectiveness of our framework, several tests are done on those algorithms.

6.1 Test Problems and Quality Indicators

1) Test Problems

Two SOP problems were tested on CLONALG while four MOP problems were tested on the other three algorithms.

1. G01 and G03

The two problems with only one object were tested on CLONALG the maximization objective is listed below:

$$g_{01}(x) = \sin^6(5\pi x)$$
$$g_{03}(x,y) = x\bullet\sin(4\pi x) - y\bullet\sin(4\pi y + \pi) + 1$$

The area defined by G01 is [0,1] and in G03 both x, y are in area [-1,2].

2. SCH

This is a simple MOP problem defined by Schaffer in 1984, the two objectives are:

$$f_1(x) = \begin{cases} -x, & if\ x \le 1 \\ -2+x, & if\ 1 < x < 3 \\ 4-x, & if\ 3 < x \le 4 \\ -4+x, & if\ x > 4 \end{cases}, \quad f_2(x) = (x-5)^2$$

While we usually defined x in [−5, 10].

3. DEB

DEB is a much more complex two-object problem with 2 decision variables:

$$f_1(x) = x_1$$
$$f_2(x) = (1+10x_2)\times\left[1-\left(\frac{x_1}{1+10x_2}\right)^2 - \frac{x_1}{1+10x_2}\sin(8\pi x_1)\right]$$

Each variable is in [0, 1].

4. ZDT1

ZDT is a serial of two objective problems with many decision variables, we take 30 variables in [0, 1] of this problem in our tests:

$$f_1(x) = x_1\,,\quad f_2(x) = g(x)\left[1-\sqrt{x_1/g(x)}\right]$$
$$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$$

5. DTLZ3

The DTLZ problems serial can be scaled to any number of decision variables and objectives. We took 3 objectives and 7 variables in our test.

$$f_1(x)=\frac{1}{2}x_1x_2...x_{k-1}(1+g(x_k))$$
$$f_2(x)=\frac{1}{2}x_1x_2...(1-x_{k-1})(1+g(x_k))$$
$$\vdots$$
$$f_{k-1}(x)=\frac{1}{2}x_1(1-x_2)(1+g(x_k))$$
$$f_k(x)=\frac{1}{2}(1-x_1)(1+g(x_k))$$
$$where\quad g(x_k)=100\left[|x_k|+\sum_{x_i\in x_k}\left((x_i-0.5)^2-\cos(20\pi(x_i-0.5))\right)\right]$$

All the variables are in [0,1].

2) Quality Indicators

Quality indicators are used to weight the result of the MOP problems' quality, just as diversity, coverage and others.Two quality indicators are used in our test.

1. Generation Distance (GD)

GD is a measure of the distance between the true and generated Pareto front. It is defined as:

$$GD \triangleq 1/N_{known}\sqrt{\sum_{i=1}^{N_{known}} d_i^2} \tag{10}$$

Nknown is the solution number of the generated front, and **di** is the shortest Euclidean distance between each solution in generated front and the true front. The smaller GD is , the better the quality of the generated front.

2. Hyper-Volume Rate (HVR)

This indicator is usually used to measure the diversity of generated pareto-fronts. It is the hypervolume ratio of generated pareto-front and true front.The hypervolume is defined as:

$$\mathrm{VOL}\left(\bigcup_{(x,y)\in P}[R_x,x]\times[R_y,y]\right) \tag{11}$$

With **VOL()** being the usual Lebesgue measure, and [Rx,x] [Ry,y] are the points of each front. The closer HVR is to 1.0 , the more diversity the generated front has.

6.2 Algorithm Parameters Setting

1. CLONALG

Using GLONALG, we only tested two simple single objective problems, G01 and G03, so the parameter set is not complex: we set the whole population size to 10 in one iteration and the maximum iteration limit is 1000.

For clonal selection operators, **Rs** is 4 and **Rc** is 5. For the hyper-mutation, **Rm** is 1.0 in G01 and 0.5 in G03, **Rb** is 0.7 in both problems. In final re-select process, we select out first 4/5 anti-bodies and second newly generated 1/5 antibodies.

2. MISA

In our version of MISA, we set the population size as 200 and the maximum iteration limit to 10000. For clonal selection operators, **Rs** is 20 and **Rc** is 5. For the hyper-mutation, **Rm** equals the inverse of problem`s dimension , **Rb** is as big as the defination area of the problems. For GA Crossover operator, in our version we use SBXCrossover and set the probability to 0.8.

For the original implementation, we take the one coded in C++ by the MISA`s author. The population size and iteration limit are the same, while other parameters obey the settings in paper [13].

3. NNIA

On our version of NNIA, we set the population size to 100 and the maximum iteration limit to 500. For clonal selection operators, **Rs** is 40 and **Rc** is 5. For the hyper-mutation, **Rm** equals the inverse of problem`s dimension. **Rb** is as big as the definition area of the problems. Other parameters obey the settings in paper [12].

For the original implementation, we take the one coded using Matlab by the NNIA`s author. All parameters obey the settings in paper [12].

4. CMOIA

For CMOIA`s original implementation based on our framework, we only execute the single test on that version as a complement of paper [4]. The population size is set at 200 and the maximum iteration limit is 5000. For the clonal selection operator **Rs** is 100 and **Rc** is 5. For the hyper-mutation, **Rm** is 0.25 and **Rb** is as big as the definition area of the problems. For crossover we set it the same as MISA.

6.3 Test Results and Analyses

1. CLONALG

We listed the test result of CLONALG independently for it is tested using two SOP problems. Also we do not apply any quality indictors to it.

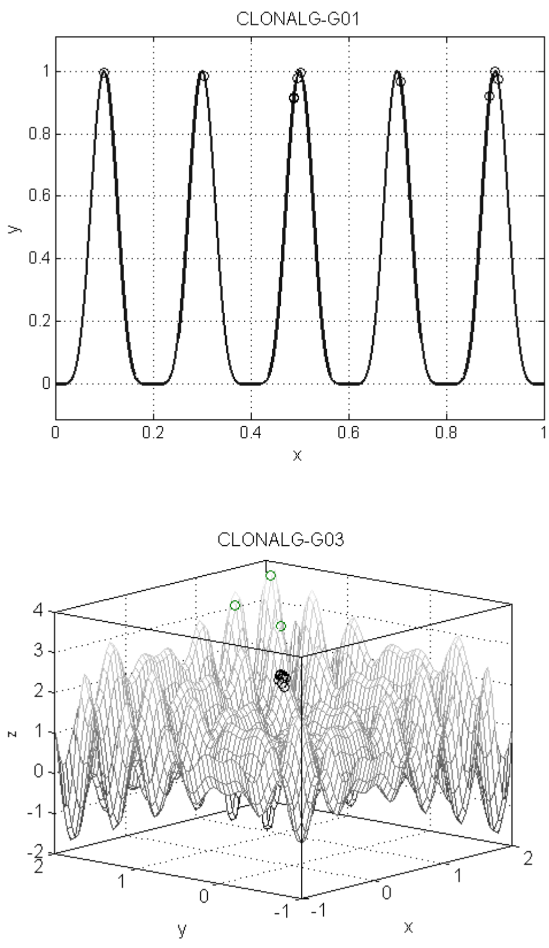
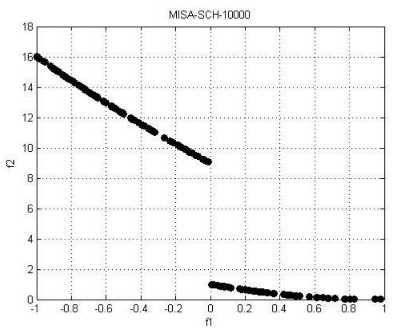


Figure 6. CLONALG on Problem G01, G03

From Fig. 6 we can see our CLONALG implementation could easily solve the G01 problem as well as the original version, but in the complex G03 problem, our implementation performs a bit weaker.

2. MISA

On MISA and the other two MO-AIS algorithms, we applied a more comprehensive test then CLONALG. We firstly test four MOP problems each 30 times, then apply two quality indicators to them to get an average result.



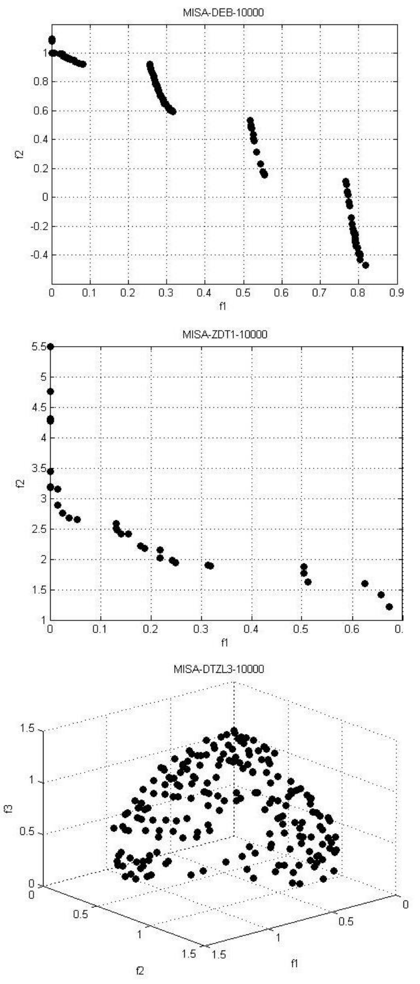


Figure 7. MISA on Problem SCH, DEB, ZDT1, DTLZ3

Then two quality indicators are applied to our version and the original implementation, 30 times average data is listed in Table 6.

Table 6: Quality Compare on MISA

	GD (*10^{-5})		HVR	
	Ours	Origin	Ours	Origin
SCH	1.951	1.875	0.9939	0.9956
DEB	1775	436	0. 9121	0.9369
ZDT1	33999	28490	----	----
DTLZ3	174	158	0.7910	0.8128

From the table we know that on SCH\DEB\DTLZ3, MISA could work out an acceptable result, but in ZDT1, both of them could not work well. And in all problems our version trailed past the original version, but the gap is not big.

3. NNIA

Using NNIA we execute the same test methods as with MISA. The result shows as follow:

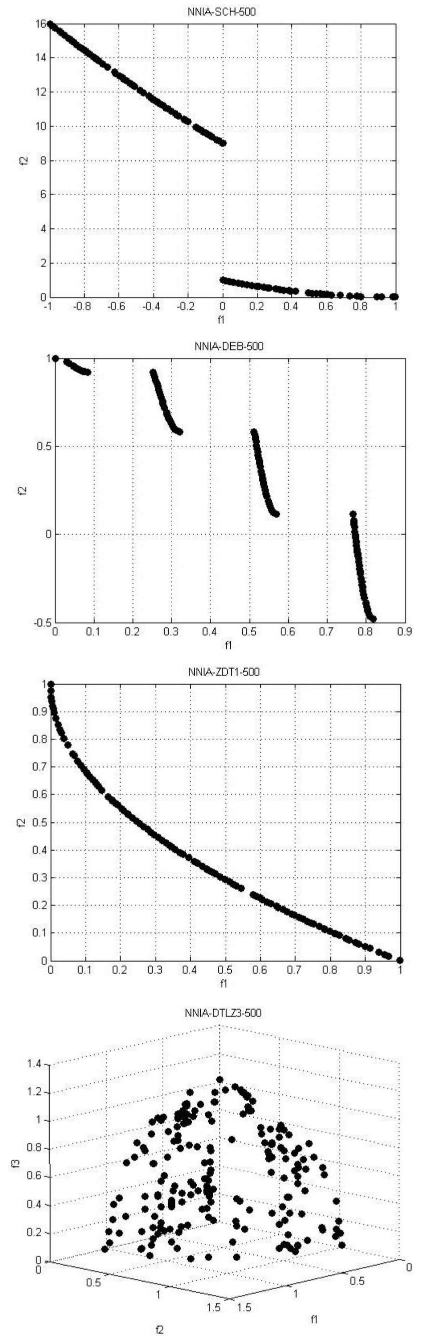


Figure 8. NNIA on Problem SCH, DEB, ZDT1, DTLZ3

Table 7: Quality Compare on NNIA

	GD (*10^{-5})		HVR	
	Ours	Origin	Ours	Origin
SCH	2.747	2.559	0.9939	0.9956
DEB	613	589	0. 9121	0.9369
ZDT1	7.611	6.909	0.9722	0.9810
DTLZ3	163	76.48	0.7910	0.8128

From the data we know that all versions of NNIA perform well on the four test problems. The quality gap between our version and original version of NNIA is much smaller than with MISA.

4. CMOIA

For the reason mentioned above, we tested CMOIA using only one version. The results shows as follow:

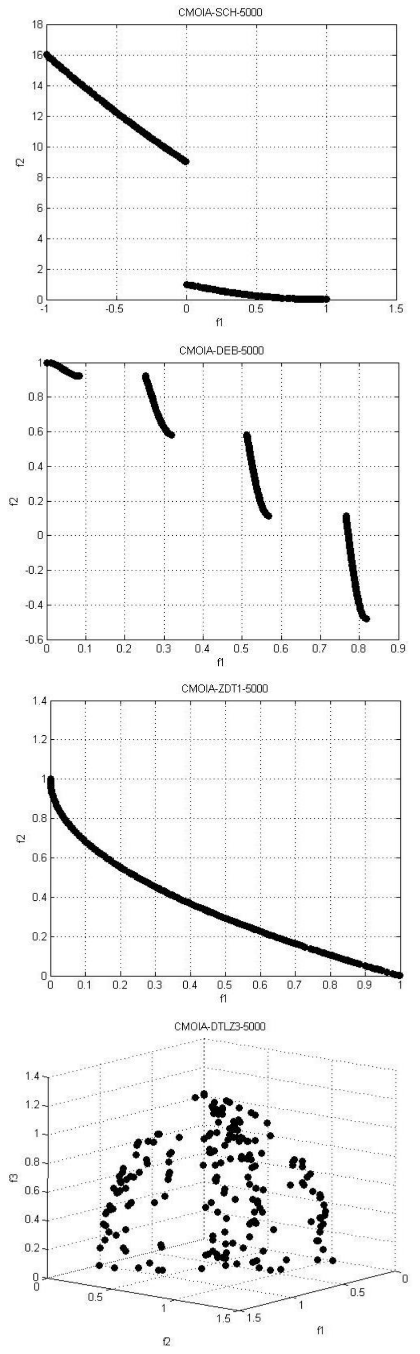


Figure 9. CMOIA on Problem SCH, DEB, ZDT1, DTLZ3

Table 8: Quality Indicators on CMOIA

	GD (*10^{-5})	HVR
SCH	1.932	0.9959
DEB	822	0. 8955
ZDT1	3.279	0.9803
DTLZ3	245	0.7456

The original version of CMOIA is implemented by our team using this framework. We listed the previous test result here only to show that our framework could serve well for new MO-AIS algorithm research.

VII. Conclusion And Future Work

In this paper, we suggest that all MO-AIS algorithms (mainly based on the clonal selection principle) could be summed up by a series of common principles, strategies and processes to form a specific framework. Then though the analysis of the MO-AIS algorithms` characteristic's we describe the framework`s conception, architecture and implementation methods. We explained the algorithm implementation method using our framework by implementing four typical MO-AIS algorithms. Finally several tests are applied on our implementation to show that our framework is fully useful and feasible.

In the future we will try to develop a highly-organized framework for this area including the implementation of algorithms based on immune network principles, and publish the framework`s code in a highly readable account.

The main difference between MO-AIS algorithms nowadays is the improvement of operators` implementations and the weighting methods. Our framework could serve very well for researching, comparing and proposing MO-AIS algorithms.

Acknowledgment

The authors would like to thank the anonymous reviewers for their careful reading of this paper and for their helpful comments. This work was supported by the National Natural Science Foundation of China under grant no. 61170065.

References

[1] Serafini, P. Some considerations about computational complexity for multi-objective combinatorial problems LNEMS, Vol. 294, Springer , p222-232, 1987.

[2] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.

[3] Q. Zhang and H. Li, MOEA/D: A multi-objective evolutionary algorithm based on decomposition. IEEE Transaction Evolutionary Computation, vol. 11, no. 6, pp. 712–731, 2007.

[4] Knowles, J. D. & Corne, D. W.: The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization. In Proc. of 1999 Congress on Evolutionary Computation, vol. 1,pp. 98–105. Piscataway, NJ: IEEE Press, 1999.

[5] HC. Lau, W. Wang. A Multi-Agent Approach for Solving Optimization Problems involving Expensive Resources, In Proc of 2005 ACM Symposium on Applied Computing: 79-83, 2005.

[6] Farmer, J. D., N. H. Packard and A. Perelson. The Immune System, Adaptation, and Machine Learning. Physica D 22(1-3): 187-204, 1986.

[7] J. Zheng, Y. Chen, W. Zhang. A Survey of Artificial Immune Applications. Artificial Intelligence Review 43:19-34, 2010.

[8] Jerne, N. K.: Towards a Network Theory of the Immune System. Ann. Immunology, Vol. 125C, 373-389, 1974.

[9] P. Hajela and J. Lee. Constrained genetic search via schema adaption: An immune network. Structural and Multidisciplinary Optimization, Vol. 12, No. 1: 11-159, 1996.

[10] J.J. Durillo and A.J. Nebro and E. Alba.The jMetal Framework for Multi-Objective Optimization: Design and Architecture. In Proc of 2010 Conference on Evolutionary Computation: 4138-4325, 2010.

[11] L.N. de Castro and F.J. Von Zuben. Learning and Optimization Using the Clonal Selection Principle, IEEE Transaction Evolutionary Computation, Vol. 6, No. 3:239-251, 2002

[12] M. Gong, L. Jiao et al.. Multiobjective Immune Algorithm with Nondominated Neighbor-based Selection, Evolutionary Computation Vol.16, No.2: 225–255, 2008.

[13] C. Coello and N.C. Cortes. Solving Multiobjective Optimization Problems using an Artificial Immune System, Genetic Programming and Evolvable Machines, 6:163–190, 2002.

[14] F. Sun, Y. Chen, W. Wu. Multi-objective Optimization Immune Algorithm Using Clustering. In Proc of 2010 International Conference of Bio-Inspired System and Signal Processing, IEEE: 9-13, 2010.

[15] S. Forrest, A.S. Perelson et al.. Self-Nonself Discrimination in a Computer. In Proc of 1994 IEEE Symposium on Research in Security and Privacy: 202-212, 1994.

[16] Garrett, S.M., 2004, Parameter-free, adaptive clonal selection, In Proc of Congress on Evolutionary Computing, Portland Oregon, USA, Volume 1, pp. 1052-1058, 2004.

[17] L.N de Castro and Timmis. An artificial immune network for multimodal function optimization. In proceedings of the IEEE Congress on Evolutionary Computation Honolulu :699-704, 2002.

[18] Freschi, F., Repetto, M. Multiobjective optimization by a modified artificial immune system algorithm. 4th International Conference on Artificial Immune Systems, Lecture Notes in Computer Science 3627:248-261, 2005.

[19] Luh, G.C., Chueh, C. H., and Liu W.W.. MOIA: Multi-objective Immune Algorithm. Engineering Optimization, 35 (2): pp. 143-164, 2003.

[20] Wong, E.Y.C., Yeung, H.S.C., and Lau, H.Y.K. Immunity-based hybrid evolutionary algorithm for multi-objective optimization in global container repositioning, Journal of Engineering Applications of Artificial Intelligence, Vol. 22 Issue 6: 842-854, 2009.

[21] Wang, X.L., Mahfouf, M.. ACSAMO: an adaptive multi-objective optimization algorithm using the clonal selection principle. In Proc of 2nd European Symposium on Nature inspired Smart Information Systems, 2006.

[22] Knowles, J. D. & Corne, D. W.: The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization. In Proc. of 1999 Congress on Evolutionary Computation (ed. P. J. Angeline), vol. 1,pp. 98–105. Piscataway, NJ: IEEE Press. 1999.

[23] Knowles, Joshua D. and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation, 8(2):149–172, 2000.

[24] Hamming, Richard W.. Error detecting and error correcting codes, Bell System Technical Journal 29 (2):147-160, 1950.

CHEN Yunfang (1976-) , male, Nanjing, China, Associate Professor, Ph.D., his research directions include intrusion detection system, intelligence computation, immune algorithm and social computing.

How to cite this paper: Chen Yunfang,"A General Framework for Multi-Objective Optimization Immune Algorithms", International Journal of Intelligent Systems and Applications(IJISA), vol.4, no.6, pp.1-13, 2012. DOI: 10.5815/ijisa.2012.06.01