Multi-Objective Artificial Immune System Algorithm based on Hypervolume

Thomas Pierrard

Université de Nantes, UFR Sciences
2 rue de la Houssinière BP92208,
F44322 Nantes cédex 03 – France

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Departamiento de computación,
Av. Instituto Politécnico Nacional No. 2508,
Col. San Pedro Zacatenco,
México, D.F. 07300

Abstract

This document presents a new artificial immune system algorithm based on the clonal selection principle and hypervolume contribution. The main aim is to investigate the performance of this class of algorithm with respect to other algorithms which are representative of the state of the art in solving Multi Objective Problems (MOP). The main features of an immune system algorithm are implemented and some results are provided comparing them to the well-known algorithm NSGA-II. The results show that artificial immune system algorithms based on hypervolume are competitive, even when using only a few of the features, among the many available, that a true immune system can offer.

Keywords: Multi Objective Optimization, Artificial Immune System, Hypervolume

1. Introduction

For the last decade, evolutionnary algorithms (EAs) have been widely used to solve MOPs. An EA uses some mechanisms inspired by biological evolution, which have been shown to be efficient on a large set of difficult problems, including NP-Hard or even NP-complete class problems. EAs are said to be population-based algorithms because they use a set (population) of solutions that is updated at each iteration (generation). The main advantage of EAs, and metaheuristics in general, is that at each generation, the algorithm is able to provide solutions (exact or approximate) in a reasonable low amount of time (polynomial complexity). On the contrary, exact algorithms always lead to solutions (the best ones) after a costly search, even if using intelligent methods (e.g. Branch & Bound). The only drawback of EAs is that, in general, their convergence cannot be guaranteed. Nevertheless, in practice, EAs generate approximations that are generally sufficiently good to justify their use.

Email address: tom.pierrard@gmail.com (Thomas Pierrard)
URL: http://tom.pierrard.free.fr (Thomas Pierrard)

Many metaheuristics have been designed aiming to solve as many problems as possible while changing a minimum number of parameters from the algorithm. However, most of the existing algorithms which solve MOPs have to be tuned to be efficient on a large set of different and complex problems. The purpose of the algorithm proposed here is not to overcome this difficulty but rather to create a fast and robust algorithm with basic features based on artificial immune system algorithms and hypervolume contribution. The future addition of more elaborate self-adaptation mechanisms to the proposed approach should improve its performance and robustness. Next, we provide some basic concepts related to multi-objective optimization and artifical immune systems.

1.1. Multi Objective Optimization

In multi-objective optimization, the aim is to optimize two or more objective functions (which are normally in conflict with each other) at the same time. Objectives could be maximized or minimized, but here, we assume that all of them are to be minimized. The problem can be unconstrained or constrained and is generally modeled as:

$$\left[\begin{array}{lll} opt & f_i(\vec{x}) & \forall i \in \{1,...,m\} \\ s.t & & \\ & g_j(\vec{x}) \leq 0 & \forall j \in \{1,...,p\} \\ & h_k(\vec{x}) = 0 & \forall k \in \{1,...,q\} \\ & x_l \in [lb, ub] & \forall l \in \{1,...,n\} \end{array}\right]$$

where:
$opt \in \{min, max\}$,
m is the number of objective functions,
p is the number of inequality constraints,
q is the number of equality constraints,
n is the number of decision variables of the problem,
lb, ub are the lower and upper bound of each variable x_l, respectively.

Finding the optimal vector $\vec{x}^*$ of a single objective problem can be easily defined as

$$\nexists \ \vec{x} \ \texttt{such that} \ \ f_1(\vec{x}) < f_1(\vec{x}^*)$$

In the multiobjective case, each vector solution has to be optimized through more than one objective function, then, for m objectives, the problem can be described as

$$\{min \ \ f_i : \vec{x} \in \mathbb{R}^n \rightarrow \mathbb{R} \ \ \forall i \ \ | \ \ g_j(\vec{x}) \leq 0, \ \ h_k(\vec{x}) = 0 \ \ \forall j,k\}$$

In this case, there is no unique solution. Indeed, $\forall i, j \in \{1,2\}$, and for two solutions $\vec{x}_1$ and $\vec{x}_2$, we can suppose that $f_1(\vec{x}_1) < f_1(\vec{x}_2)$ and $f_2(\vec{x}_2) < f_2(\vec{x}_1)$. Then we can't deduce anything by comparing these solutions and they are considered as two feasible solutions of the problem. In order to define formally this relation, we need to introduce the notion of Pareto dominance.

1.1.1. Pareto Dominance

One vector $\vec{x}^*$ dominates (in the Pareto sense) a vector $\vec{x}$ if and only if $f_i(\vec{x}^*) \leq f_i(\vec{x}) \ \forall i \in \{1,...,m\}$ and there exists at least one i such that $f_i(\vec{x}^*) < f_i(\vec{x})$ (assuming minimization). We also say that $\vec{x}$ is dominated by $\vec{x}^*$. A binary operator "$\prec$" is defined as:

$$\vec{x} \ \texttt{dominates} \ \vec{y} \Longleftrightarrow \vec{x} \prec \vec{y}$$

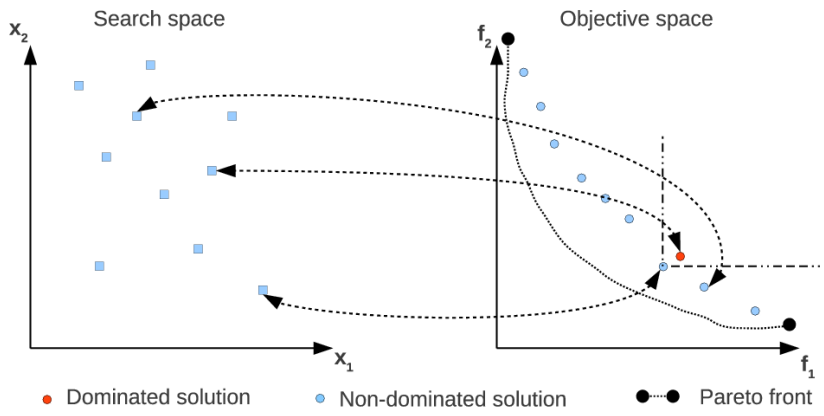


Figure 1: A simple bi-objective problem with two variables

1.1.2. Pareto Optimal Set & Pareto Front

All the solutions whose vectors are not dominated by any other vector in $\mathbb{R}^m$ are said to be **non-dominated**. Pareto optimal set is composed of all vectors in the search space $\mathbb{R}^n$ that are non-dominated. The image of the Pareto optimal set (i.e., their objective function value) form the Pareto front. The main goal in multi-objective optimization is to generate as many elements of the Pareto optimal set as possible. A simple graphical representation is given in Figure 1.

1.1.3. Spread & Convergence

When working with multi-objective problems, two important notions have to be well-understood : convergence and spread. The convergence allows to measure how far the solutions found are, from the true Pareto front; a classical performance measure to calculate it is the Generational Distance (GD[1]). The algorithm's convergence can also be seen as the time (or number of generations) needed to reach the Pareto front. For example, an algorithm would have a good convergence if

$$\exists\, \epsilon \in \mathbb{N},\ \epsilon \ll T,\ \texttt{such that}\ \lim_{t \to \epsilon} GD_t(S) = 0$$

where:
S is the set of solutions found,
T is the maximum number of generations.

The spread indicates how well-distributed are the solutions on the true Pareto front (or its approximation). This is an important indicator as it gives more choices to the decision-maker when choosing one or more non-dominated solutions. The spread proposed in [1] is a widely used indicator that adopts, of course, the distance between solutions, but also the Pareto front's extreme solutions, in order to take into account the maximum interval where the solutions are found for each objective.

1.2. Multi Objective Artificial Immune System (MOAIS)

1.2.1. The Immune System

The immune system's role is to defend the body against infections. It has several defenses against outside attacks: the barrier of the skin, mucous membranes, and the passive system defense of cells, but the

[1] Mean distance betweeen solutions found so far and the nearest solution belonging to the true Pareto front

functioning of antibodies is its main element. Usually, when a foreign element is detected by the immune system, an immediate elimination reaction sets in. This reaction involves phagocytic cells and lymphocytes that circulate continuously throughout the body. This reaction is fast and called non-specific, meaning that the immune system attacks the antigen without knowing its nature.

Depending on the severity of the infection, this rapid and non-specific immune response may not be sufficient to eliminate the intruder. A second reaction, slower and more specific will be set up: it puts into play the recognition of the foreign element by immune cells. Following the recognition, immune cells specifically adapted for the destruction of the foreign agent (lymphocytes) will multiply rapidly. Some of these clones may be corrupt, and a risk of generating autoimmune cells occurs. The immune system is able to suppress self-generated cells (suppression of similar individuals in MOO). Subsequently, the organism keeps track of this encounter with the foreign element (thanks to the B cells). There is some form of memory in the immune system. This will optimize the specific immune response, which will be faster at a forthcoming encounter with the same foreign element.

Immunity is a very complex system that is difficult to simulate retaining all of its characteristics. Nevertheless, the key ideas that can be used to build an AIS are:

- A set of immune agents (antibodies) that try to find the best binding to fit to pathogen agents (antigens).
- A set of cells that record characteristics of antigens previously encountered.
- Communication between these entities.
- The capability of some cells to clone (asexual reproduction) and mutate.

Some more features of AIS are given in [2], nevertheless, most of them are not relevant for MOO and, therefore, are not discussed here.

1.2.2. AIS applied to MOO

In MOAIS, we usually consider two sets of solutions, antibodies and antigens. Differences between these are defined by the designer of the algorithm. The most common idea is to split good and bad solutions. Interactions between the solutions (Ag-Ab, Ag-Ag,...) are usually defined by a function called “affinity” with classical methods such as distance measures, Pareto dominance ranking, etc. Depending on the affinity value a selection and a cloning process occurs, then the clones are mutated. Finally, a strategy is used to generate the new population and to store the best solutions found so far (archiving is nowadays a common feature in MOEAs). In [2], a canonical algorithm gives the main procedure of an MOAIS, which is reported here in Algorithm 1 with the notation of this document.

The canonical MOAIS algorithm first defines the problem, like all population-based algorithms (line 1). An archive is defined (line 2) in order to store the non-dominated solutions found so far. The online population is initialized (line 3) containing the solutions from the current generation. The main loop starts and performs the following steps until a stop criterion is met. The algorithm evaluates the online population (line 5) using objective functions and constraints. Depending on the choices made, the solutions of the set B are analysed and given an affinity value (line 6), the archive A can be used, for example to define the new affinities between best solutions found so far. The cloning selection is triggered following stochastic or deterministic rules (line 7), based on affinities values or not. The cloning process is usually done based on the affinity values (proportional cloning), while the mutation of each individual can have several variants (line 8). The two previous steps are commonly adapted in all AIS, this is called the clonal selection principle.

Algorithm 1: Outline of the canonical MOAIS

1 Define the search space $\mathbb{S}$, objectives functions f_i, constraints g_j, h_k ;
2 $A(t = 0) \longleftarrow$ Initialize offline population;
3 $B(t = 0) \longleftarrow$ Initialize online population with random individuals;
4 **while** $\neg$ *stop criterion* **do**
5 Evaluate population $B(t)$ using f_i, g_j, h_k;
6 $B_1(t) \longleftarrow$ Define affinities($B(t)$, $[A(t)]$);
7 $B_2(t) \longleftarrow$ Selection for cloning($B_1(t)$, $[A(t)]$);
8 $B_3(t) \longleftarrow$ Proliferation and mutation($B_2(t)$);
9 $B_4(t) \longleftarrow$ Diversification & Suppression;
10 $B(t + 1) \longleftarrow B_3(t) \cup B_4(t)$;
11 $A(t + 1) \longleftarrow$ Update($A(t)$, $B(t + 1)$);
12 $t \longleftarrow t + 1$;
13 **end**

The diversification procedure (line 9) is not mandatory, its goal is to add methods to bring some diversity to the population usually by creating new random individuals. Suppression is not mandatory either and can be applied to delete some individuals (responsible for autoimmune disorder), particularly to individuals that are not relevant for further optimization. The new population is generated taking into account the best clones (line 10), applying some predefined rules. Eventually, the archive is updated (line 11).

1.3. State of the art

An overview of Artificial Immune System for MOO is given in [3]. It shows that MISA[2] is considered as the first MOAIS proposed in the litterature [4]Cruz Cortés, 2002). The algorithm is designed to fit the immune system metaphor and it follows the canonical algorithm previously presented. MISA uses the classical non-dominated Pareto ranking algorithm to classify solutions and to determine which of them will be cloned. The number of clones depends on antibody-antibody affinities. The clones are uniformly mutated according to their antigen-antibody affinities whereas other solutions use non-uniform mutation. An adaptive grid is used to ensure diversity in the fixed size archive. Selection to access the archive is determined by some defined rules based on the non-dominated Pareto ranking. This algorithm is the first AIS explicitly designed to solve MOP, and its results show that in spite of being, based on simple rules, MISA can be efficient. MISA was successively improved until 2005 but, after that, metaphors from the immune system were not followed as strictly as before. The following algorithms presented are chosen according to five critera:

1. Respect of the canonical AIS algorithm
2. No use of a recombination operator
3. Implemented for real-coded variables
4. Detailed results
5. Most recently published

[2]Multiobjective Immune System Algorithm

1.3.1. VAIS

In [5], Freschi & Repetto (2005, 2006) present VAIS[3], an algorithm using selection, cloning, mutation, suppression and an archiving process. For non-dominated individuals, fitness is determined by the strength defined in SPEA2 [6]. For dominated solutions, fitness corresponds to the number of individuals which dominate them. A suppression procedure is used for the archive as well as a diversification procedure by allowing a fixed number of random individuals to enter the archive. Results are compared with NSGA-II and show that VAIS can outperform NSGA-II on unconstrained and constrained problems such as Tanaka, Viennet and Zitzler. Nevertheless, no results are provided on DTLZ problems, which are considered to be harder to solve.

1.3.2. IDCMA/NNIA

IDCMA[4] was presented in [7] (Jiao, Gong, Shang, Du & Lu, 2005). As the algorithm had difficulties solving DTLZ problems, NNIA[5], was later presented as an improved version of IDCMA in [8] (Gong, Jiao, Du & Bo, 2008). The selection mechanism which chooses the set of candidates to be cloned is based on non-dominated solutions. If the non-dominated solutions are beyond a certain threshold, then the crowding distance is used. The archive process uses the same methods to select candidates to enter the archive. In NNIA, recombination is used, nevertheless, some results are presented with and others without this feature. It shows that recombination is a powerful method that gives better results regarding the ”two sets coverage” between NNIA-X[6] and NNIA (ZDT[1-4,6], DTLZ[1-4,6]).

1.3.3. IFMOA

In [9], IFMOA[7] (Lu, Jiao, Du & Gong, 2005) is presented. The affinity assignment is based on Pareto strength[6] and antibody-antibody affinity is inversely proportional to the sum of two smallest Euclidean distances between an antibody and the rest of the population. The “immune forget unit” is a set of solutions that are not participating in clonal proliferation. Results are given by comparing the algorithm to MOGA and SPEA2 on six unconstrained MOP. On the results shown, the algorithm performs well but no results are given for more difficult problems.

1.3.4. omni-aiNet

Coelho & Von Zuben (2006) presented omni-aiNet as a single and multi objective optimizer in [10]. First, all the individuals will be cloned N_c times. N_c is a parameter. A random variation with rates inversely proportional to its affinity to the antigen is applied to each generated clone. Polynomial mutation is used to apply variations to the clones. Solutions are arranged in classes, the better the class, the smaller the variation. The algorithm is using suppression and diversification. Unfortunately, results are provided only by comparing the algorithm with another algorithm called ”DT omni-optimizer”. Moreover, results are only graphical and on only focused three problems.

1.3.5. SMS-EMOA

It is worth presenting the well-known SMS-EMOA [11] algorithm, which is based on Hypervolume. At each iteration, a new solution is generated by means of randomised variation operators. Then, the

[3] Vector Artificial Immune System
[4] Immune Dominance Clonal Multiobjective Algorithm
[5] Nondominated Neighbor Immune Algorithm
[6] NNIA without recombination
[7] Immune Forgetting Multiobjective Optimization Algorithm

algorithm sorts the solutions by ranks and discards an individual from the worst rank that contributes the least in maximizing the Hypervolume. All results show that SMS-EMOA is a very powerful solver even for difficult problems. Its main drawback however,is its high computational complexity, which is related to the calculation of the hypervolume contribution which can be very time consuming when the number of objectives increases.

2. The MOAIS-HV algorithm

In this section, MOAIS-HV is detailed. First, some basic about hypervolume are discussed, then the algorithms, data structures and other choices that have been taken to implement MOISA-HV are presented. The main goal of the algorithm is to investigate the quality of the results while combining AIS and Hypervolume. The algorithm is designed to be tested on a large number of problems, to have a low complexity, to respect the number of function evaluations and to follow the characteristics of a pure AIS algorithm (no recombination operator is adopted).

2.1. Hypervolume vs. Hypervolume contribution

Hypervolume is a very common indicator used to measure and compare the quality of final solutions in population-based algorithms. The hypervolume measure was originally proposed by Zitzler and Thiele in [12]. This indicator represents the surface (or the volume for more than 2 objectives) of the region dominated by solutions found so far. Let Λ denote the Lebesgue measure, then the $\mathcal{S}$ metric is defined as

$$\mathcal{S}(A, y_{ref}) = \Lambda\left(\bigcup_{y \in A}\left\{y' \,|\, y \prec y' \prec y_{ref}\right\}\right), A \subseteq \mathbb{R}^m$$

where:
A is a subset of the objective space,
y_{ref} denotes a reference point that is dominated by all Pareto-optimal solutions,
" $\prec$ " denotes the dominance relation.

The main drawback of the hypervolume indicator is that no general polynomial algorithm exists in order to calculate it, therefore it has been unpopular for use as an online feature in MOO. Nevertheless, it is now well-known that using the hypervolume indicator provides good results in both convergence and spread. Indeed, it has been shown in [13] that given a finite search space and a reference point, maximizing hypervolume enables the finding of all the non-dominated solutions of the Pareto front.

In this document, hypervolume is used to select μ individuals among $n = \lambda + \mu$ individuals. λ represents the number of individuals discarded one by one. The μ individuals are the candidates to be cloned. The aim is to find the set of μ individuals that maximizes the hypervolume. Two methods are commomly used, each of them having advantages and drawbacks:

- Hypervolume indicator

  This algorithm computes the hypervolume for the whole set of solutions. Nowadays, there is no general polynomial algorithm to calculate it for any number of objectives. Thus, computing $\binom{n}{\mu}$ hypervolumes is considered to be too time consuming.

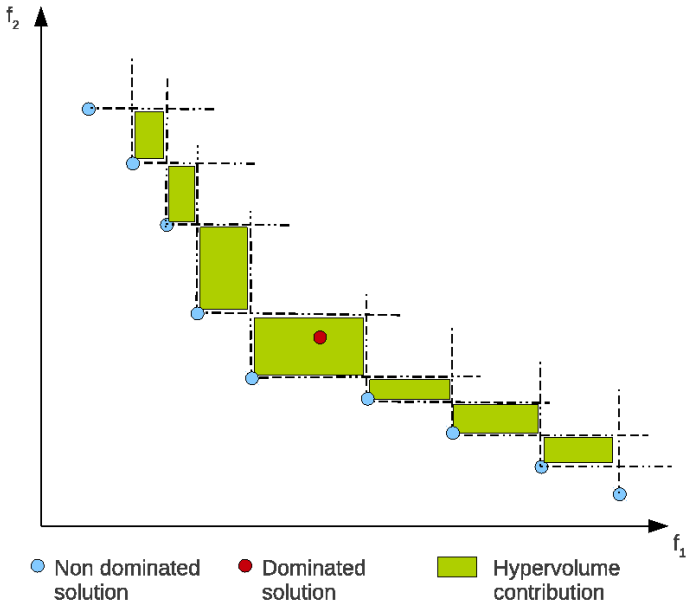


Figure 2: Hypervolume contribution

- Hypervolume contribution

  Hypervolume contribution computes the contribution of each solution in maximizing the hypervolume taking into account its neighbours on each objective, see Figure 2. Selecting the optimal set of μ solutions implies calculating $\binom{n}{\mu}$ conventional hypervolume contributions, which is considered to be computationally too expensive. In [14], an algorithm is presented to compute the set of λ solutions that contributes less in maximizing the hypervolume in $O(n^{\frac{m}{2}} log\ n + n^{\lambda})$, where n is the number of solutions, m the number of objectives, and λ the number of solutions to discard.

  Another method just discards the lowest contributor of a population iteratively until reaching a population of size μ. It has been shown in [15] that this method can lead to a set that is not the optimal according to the hypervolume maximization. Nevertheless, the error ratio is not higher than 35% and the small complexity of such an algorithm makes it more competitive compared to state of the art algorithms.

Considering these facts, in the algorithm presented in the following, the selection of individuals will be processed by discarding iteratively solutions that contribute the least in maximizing the hypervolume. The reason is to avoid using complicated data structures and to have an acceptable complexity. Future work should improve the efficiency of the hypervolume algorithm. The method used for calculating the hypervolume contribution is to sort the population of each objective via a quicksort; each solution is assigned the distance to its neighboor for the first objective. Then, this value is multiplied by distances found by the same process on other objectives. This can be done in $O(mn^3)$, but requires $O(mn^2 log\ n)$ on average.

2.2. Description of the algorithm

The following describes the new immune system algorithm based on hypervolume contribution. The main idea is to maintain an online population of antigens and antibodies. The antigens are considered to be the good solutions, the antibodies the bad ones. These two sets form two new subpopulations. The antigens

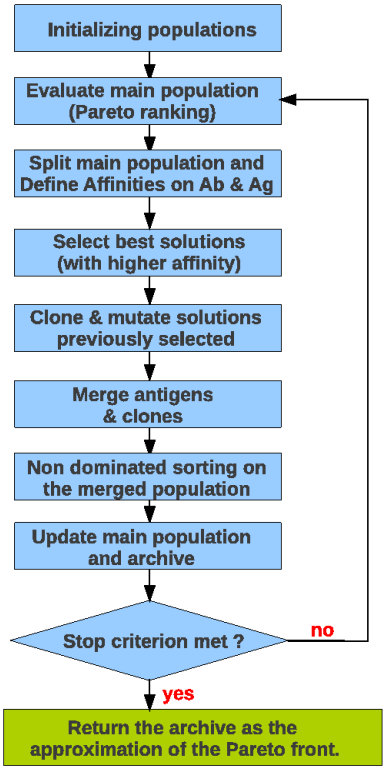


Figure 3: Main algorithm loop

are cloned (best antibodies are cloned too if the number of antigens is insufficient) and a mutation operator is applied. If only one rank exists, candidates to be cloned are selected from individuals that contribute the most to maximize the hypervolume, otherwise successive ranks are selected and hypervolume selection is only applied to the last one. The clones and the best antigens found are merged and the size of the main population is maintained by discarding individuals that contribute the least in maximizing the hypervolume. The main loop of the algorithm is represented in Figure 3. In the algorithm, the following notations will be used:

Q, P: the main population and the pool,
Ab, Ag: the sets of antibodies and antigens (subsets of Q),
n: size of the main population,
$m, ngen$: number of objectives and number of generations.

1. Initializing populations:
   Initialize population Q (Main population) by generating random individuals.
   → fixed size n.
   Initialize Antibodies population Ab to empty.
   → fixed size n.
   Initialize Antigens (or Archive) population Ag to empty.
   → fixed size n. Store the best individuals found so far.
   Initialize a pool P to empty (to store the clones).

→ fixed size $2 * n$.

2. Evaluate all individuals of the population Q.
   → Feasibility and objective values
   For constrained problems, the constraints are handled as in NSGA-II.
   → Fast non-dominated ranking
   In order to compute the ranks of all individuals, the well-known algorithm presented in [16] is used and has a complexity of $O(mn^2)$. Nevertheless, more investigation would have to be done regarding the paper of Bentley (1993) [17] who presents an algorithm to find the convex hull of a set in $mn + O(n^{1-1/m} log^{1/m} n)$ scalar comparisons.

3. Split the population Q into two sets:
   Constrained problems:
   Antigens:
   → Feasible and non-dominated
   Antibodies:
   → Unfeasible and non-dominated
   → Feasible and dominated
   → Unfeasible and dominated
   Unconstrained problems:
   Antigens:
   → Non-dominated
   Antibodies:
   → Dominated

4. Define Affinity for antibodies and antigens.
   All the distance measures are normalized values on all objectives in order to avoid the relative importance between objectives.
   Defining affinity on antibodies:
   For each antibody, select randomly one antigen in Ag. The affinity value of an antibody Ab is defined by its euclidean distance to the selected antigen Ag.

$$Aff(Ab_i) = \frac{1.0}{Eucl\_Dist(Ab_i, Ag)}) \; \forall i$$

   If there's no antigen, each antibody is assigned an affinity based on its rank:

$$Aff(Ab_i) = \frac{1.0}{(Rank(Ab_i) + 1)}) \; \forall i$$

   For each antibody belonging to bounded solutions, the affinity is equal to the maximum affinity found so far. By doing this, antibodies belonging to the extreme solutions on each objective will be more cloned and thus increase the probability to extend the Pareto front.

$$Aff(Ab_{ext}) = max_i(Aff(Ab_i)) \; \forall ext$$

   Defining affinity on antigens:
   For each antigen, the affinity is based on Hypervolume contribution:

$$Aff(Ag_i) = Hyperv\_Cont(Ag_i) + max_j(Aff(Ab_j)) \; \forall i$$

Algorithm 2: Computing Hypervolume contributions

```
Input: Population Ag ;
Initialize Affinities(Ag) to 0.0 ;
for i from 1 to m do
    Sort Ag by obj i;
    for j from 1 to Ag.size() do
        Ag.ind[j].affinity += (Ag.ind[j].obj[i] − Ag.ind[j + 1].obj[i]);
    end
end
for j from 1 to Ag.size() do
    Ag.ind[j].affinity += max_j(Aff(Ab_j));
end
```

The algorithm that computes Hypervolume contribution is presented in algorithm 2
For each antigen belonging to bounded solutions, the affinity is equal to the maximum affinity found so far (on antigens). By doing this, antigens belonging to the extreme solutions on each objective will be more cloned and thus increase the probability to extend the Pareto front.

$$Aff(Ag_{ext}) = max_i(Aff(Ag_i)) \; \forall ext$$

For both antibodies and antigens, the greater the affinity, the better.

5. Clonal selection principle
   Most of the population-based algorithms don't discard dominated individuals when selecting solutions to be cloned or mutated. The main aim in doing this is to keep some diversity in the population. After some experiments, the choice for this algorithm was to select dominated solutions only if necessary (if non-dominated solutions are fewer than the number of clones). Some previous results have shown that cloning the antibodies gives worse results (convergence metric) than considering the best individuals found so far, the antigens. In order to fit to the immune system metaphor, one can consider here that if the main population already contains a certain number of antigens, it means that the immune system has already recognised some pathogen agents and it will use them to perform the cloning process (the antibody which reached the pathogen agent and is now considered as an antigen). The number of clones is usually defined as about 20% of the population. Nevertheless, as the behaviour of the algorithm is not known and some statistics on results still have to be made, a parameter is introduced to control the number of candidates: $nb\_cl \in [0, 1]$. The number of candidates (NC) is defined once for all iterations by the formula:

$$NC = n * 10\% + nb\_cl * n * 30\%$$

In the main population, the antigens and antibodies are classed acording to their affinity. The first NC best solutions are chosen and for each of them a different number of clones will be calculated depending on their affinity. Moreover, these candidates are split into two sets $j = 1, 2$: the extreme solutions and the others. For each set, we define their total number of clones, the extreme solutions will be cloned more at the beginning and less at the end.

Generation 1, total number of clones:
For extreme solutions $P_1 = n * 50\%$, for other solutions $P_2 = n * 50\%$
Generation *ngen*, total number of clones:
For extreme solutions $P_1 = n * 10\%$, for other solutions $P_2 = n * 90\%$

Then, for each *NC* solution, the number of clones of each candidate (*NCC*) is given by:

$$NCC(A_{i,j}) = P_j * \frac{Aff(A_{i,j})}{\sum_{i=0}^{n_j} Aff(A_{i,j})} \quad \forall i, j$$

where:
$A_{i,j}$ is the i^{th} antigen or the i^{th} antibody of the set j,
P_j is the total number of clones for the set j,
n_j is the number of candidates in the set j.

6. Mutation
   Mutation is an important part of any metaheuristic since it guides the search through the generations. It is well-known that some basic mutations can't provide good results as different problems don't need the same proportions of global search and local search to find the Pareto Front.
   Most metaheuristics require parameters in order to tune the algorithm depending on the problem they are solving. As the number of parameters increases, experiments are needed and comparisons with other algorithms can then be difficult. Therefore, a small number of parameters is preferable.
   Regarding mutation, two important choices have to be made:
   - the mutation probability
     It controls the probability to mutate one variable of a vector.
   - the mutation step-size
     It controls the degree of perturbation given to the variable selected to be mutated.

   Concerning the mutation probability, the aim of this study is to simplify the algorithm in order to emphasize the combination MOAIS/Hypervolume and investigate this new idea. Moreover, comparison is easier with NSGA-II while using the same probability for mutation. Thus, the mutation probability *mp* in this document is fixed.

$$mp = \frac{1}{nreal}$$

   Nevertheless, as an improvement, it will be interesting to change the mutation probability throughout the search. For example, SANUM[8] has been shown to be efficient in [18].

7. Hybrid mutation
   Up to now, hybrid mutation [19] has been adopted to find the optimal mutation step-size. The main idea is to use two types of mutation to perform the search. When a variable is selected to be mutated, we perform either a mutation that will provide a long step size, or a small step size. The Gaussian mutation is a well-known mutation that can be very easily tuned to perform long step size or small step size mutation. Let's say we have two types of mutation, namely *GL* (Gaussian local) and *GG* (Gaussian global). At the begining of the search, depending on the problem, we have to explore a

[8] Statistics-Based Adaptive Non-uniform Mutation

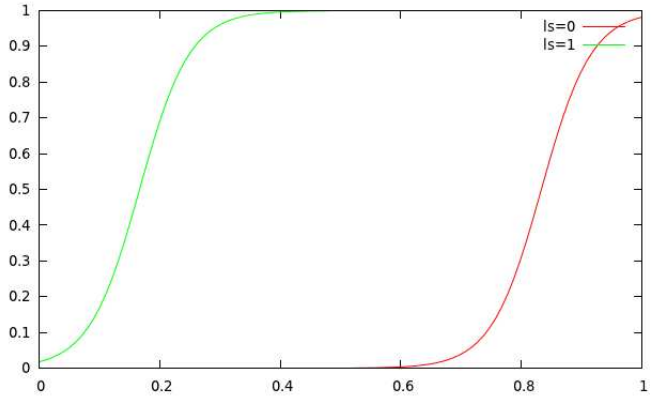


Figure 4: Gaussian mutation type probability. When ls=0, probability to do local search is very low until generation 60 (on 100 generations). If ls=1, probability to do local search is growing fast as soon as the algorithm starts.

large zone in order to improve significantly the fitness of solutions. When the algorithm performs searches near the final generations, we expect a small mutation in order to find solutions closer to the Pareto front. This idea can be seen as a kind of non-uniform mutation, but there, two types of mutation can be processed in one generation. Non-uniform mutation should be explored in a future work on the Local Gausian mutation.
Each time a variable has to be mutated, we compute the following value:

$$p\_mut\_type = \frac{1.0}{(1.0 + exp(-2.0 * (x + p)))}$$

where:
$x = -6.0 + (t/ngen) * 12.0$, increasing with the number of generations.
$p = -4.0 + ls * 8.0$, with ls a parameter which determines the tradeoff between GL and GG.

While the algorithm is running, the probability to choose GL will increase and the mutation will perform more local searches. Figure 4 shows both curves for extreme parameters $ls = 0$ and $ls = 1$.
Local Gaussian mutation is performed following this formula:

$$x_i^* = x_i + (max_i - min_i) * 0.1 * \mathcal{N}(0, st_1)$$

Global Gaussian mutation is performed following this formula:

$$x_i^* = x_i + (max_i - min_i) * 0.1 * \mathcal{N}(0, st_2)$$

where:
max_i, min_i are the bounds of the decision variable,
x_i is the variable to be mutated,
$\mathcal{N}(0, x)$ is the Normal distribution with mean 0 and standard deviation x,
$st_1 \in [0.1, 0.5]$, parameter which controls the local Gaussian mutation step,
$st_2 \in [0.5, 1.5]$, parameter which controls the global Gaussian mutation step.

The curves of the different density of probability for Gaussian mutation are shown in Figure 5.

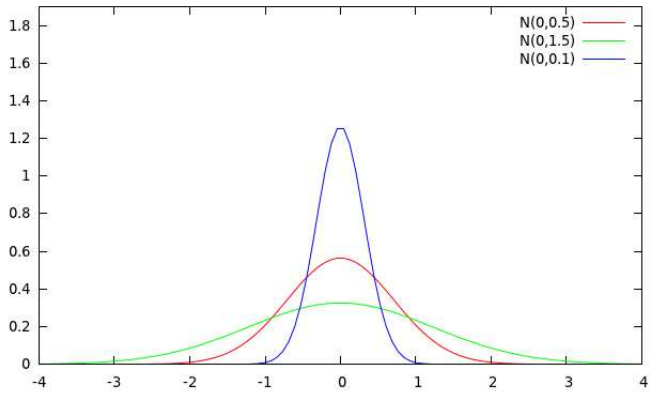


Figure 5: Gaussian mutation. These curves show three different Normal distributions with mean value 0 and three different standard deviations

8. Evaluate the pool: Objectives, feasibility.
9. Add the Antigens into the pool P
10. Non-dominated sorting in the pool P
11. Update the main population Q
    The archiving process is quite simple but some choices have to be made when the archive is full and candidates to enter it are non-dominated individuals. The main aim of the archive is to increase the value of the whole hypervolume at each generation. In order to achieve this situation, new individuals are added to the archive only if they dominate a previous individual. An archiving method is presented in [20] but the complexity is, once again, exponential with the number of objectives. In this document, the hypervolume maximization of the archive will be ensured by only accepting individuals that dominate previous individuals. One drawback of this method is that we assume that the previous generation has a good spread to ensure that all optimal solutions of the Pareto front are reachable (relatively because of the bounded size of the archive) - this is not so in most cases (see Figure 6). The following method is adopted in order to find a good spread of solutions before accepting only individuals that will maximize the Hypervolume.
    (a) Fill the main population with successive ranks from the pool P.
    (b) If the addition of the individuals of the current rank is greater than n, two cases occur:
    - if ($curgen < \frac{2}{3}ngen$), perform Hypervolume discard process (the individual which has the lowest hypervolume contribution is discarded. The procedure is repeated until reaching a rank of size $n - n_{prev}$ which will be added to the main population). The aim here is to find a good spread.
    - Otherwise, only accept individuals that dominate solutions already in the archive by replacing them. The aim here is to maximise the hypervolume.

    The value $\frac{2}{3}$ is obviously a parameter that has to be optimized upon the problem, it corresponds to an approximation of the generation number for which the Pareto front found so far is well-distributed. Knowing when the global optimum front is reached is a hard task... Some ideas to stop the hypervolume discard process would be to detect relative stability on hypervolume of the whole set or on the number of new non-dominated solutions.
12. Split the population (Antigens & Antibodies)
13. Go to step 4 if the stop criterion is not met.

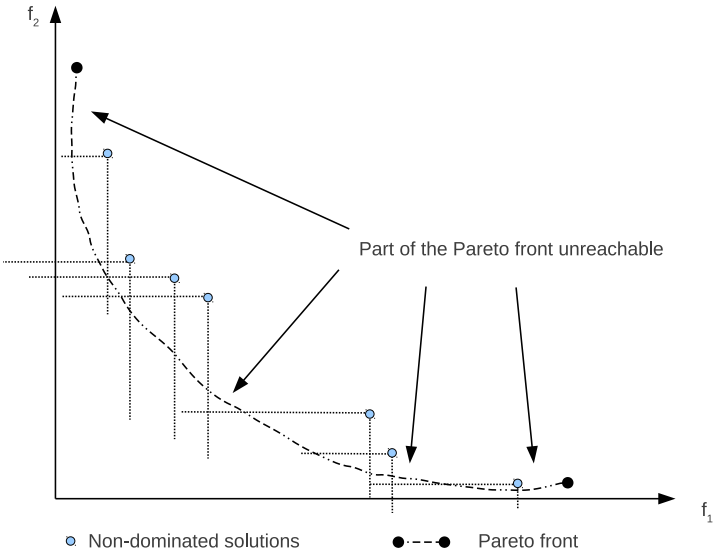


Figure 6: Pareto front gaps. By accepting only solutions that dominate previous solutions, some parts of the Pareto front cannot be reached.

14. Return the antigen population as the approximation of the Pareto front.

2.3. Data structures, Algorithms, Complexity

A MOAIS algorithm can be considered as a genetic algorithm whose features are changed for selection, proliferation, mutation and archiving. In order to keep the comparison with NSGA-II as fair as possible, the implementation of the new algorithm is based on the data structures used in [1] for NSGA-II. Other data structures used are simply arrays and the code is easily understandable and maintainable for further work.

All algorithms used are classical ones (Non dominated sorting, Dominance checker...) or are easily understandable in the code[9], thus not relevant to incorporate in this document.

A competitive complexity is always hard to achieve when dealing with hypervolume. Nevertheless, the complexity of the algorithm presented is $O(mn^3)$ and $O(mn^2 logn)$ on average. This low complexity makes the algorithm suitable for real-world problems.

3. Comparison

An algorithm has been implemented in C to compare NSGA-II and MOISA-HV. Each algorithm is run 100 times on each problem and numerical results are mean values on these runs. The algorithm provides some indicators:

- Pareto Front Hypervolume:

  Pareto front data files are quite hard to find online. Some data bases exist but files are often faulty. The Pareto front provided for the comparator has been done on a population of 1000 individuals for bi-objective problems and 2000 individuals on three-objective problems. Each problem has been run for

[9] http://tom.pierrard.free.fr/thesis/

1000 generations on each algorithm, then final populations merged to keep only the non-dominated solutions.

- Hypervolume (to be maximized):

  This indicator provides good numerical values to analyze the behavior of the algorithms for both convergence and spread. The reference point is computed according to both the algorithms and the true Pareto front. The code used is the one presented by Fonseca in [21].

- Coverage two sets (to be maximized):

  The coverage of two sets is an indicator that counts how many solutions of a population dominate solutions from another population. Because of the non-dominated solutions, this indicator is not symmetrical and should be computed for both sets. The following formula defines the two sets coverage :

$$I_c(A, B) = \frac{|\{\, \vec{x} \in A \mid \exists\, \vec{y} \in B \;:\; \vec{x} \prec \vec{y} \,\}|}{|A|}$$

  where A and B are two sets of final solutions from two different algorithms.

  The value $I_c(B, A)$ is also computed, as well as $I_c(PF, [A, B])$ where PF represents the Pareto front.

- Inverted General distance (to be minimized):

  This indicator was introduced for measuring how far the elements in the Pareto optimal set are from those in the set of non-dominated vectors found. It corresponds to the mean distance between each value of the Pareto Front and the nearest individual from the non-dominated vectors found. It is defined as:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

  where:

  n is the number of vectors in the Pareto optimal set.

  d_i is the Euclidean distance between each of these solutions and the nearest member of the set of the non-dominated vectors found

- Spread (to be minimized):

  The Spread indicator is a diversity metric that measures the extent of spread achieved among the obtained solutions. This metric is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} |d_i - \bar{d}|}{d_f + d_l + \bar{d}\,(n-1)}$$

  where:

  d_i is the Euclidean distance between consecutive solutions.

  $\bar{d}$ is the mean of these distances.

  d_f and d_l are the Euclidean distances to the extreme solutions of the Pareto front.

4. Setting up parameters

The algorithm presented can be tuned with 4 parameters, each of them belongs to $[0, 1]$:

- a: Parameter which defines the trade-off between global search and local search, (Gausian global, Gaussian local).

  0.0 allows a more global search and 1.0 allows a more local search.

- b & c: Parameters which define mutation stepsize for Gaussian mutation.

  0.0 allows smaller step-size mutation whereas 1.0 sets the mutation to a larger step-size.

- d: Parameter which defines the number of candidates to be cloned.

  0.0 represents 10% of the size of the population. 1.0 represents 40% of the population's size.

NSGA-II is used with its usual parameters [1]. For problems that were not originally implemented in NSGA-II, parameters were chosen depending on the number of objectives, variables and constraints that are efficient in other problems. Compilation and execution of MOISA-HV, NSGA-II and the comparator are detailed in their "ReadMe" files.

5. Results

The aim of this document is to compare NSGA-II and MOISA-HV, while being as fair as possible. In order to achieve this, the same number of function evaluations is used for both algorithms. The same population size and same number of generations are used. The new algorithm has been written with a view to having low complexity. A large set of problems are tested to compare the algorithms' capability to be efficient on most of them. Considering these facts, a comparison on the quality of final populations on both algorithms is processed.

A search on parameters has been done, each parameter belonging to $[0, 1]$ with a step of 0.2 (6 values for each parameter). The parameters for MOISA-HV were chosen by considering (maximizing) the ratio Hypervolume found by MOISA-HV / Pareto front. Obviously, the set of parameters chosen doesn't represent the best results provided by the algorithm and more statistics and analysis on the parameters still have to be made. In the following, a list of classical problems is given and some comments are made according to the results shown in appendices. For each problem, a score is given for each algorithm. The score is the number of indicators in which one algorithm is better than the other.

5.1. Bi-objective unconstrained problems

According to the Hypervolume, MOISA-HV performs better on all the problems tested, excepts for ZDT4 in which it gives worse results. Nevertheless, ZDT4 is multi-frontal and it is considered hard to solve. [8] shows that recombination (used in NSGA-II) can play an important role in solving such problems. Concerning the other indicators, some difficulties are encountered on ZDT3. The discontinuous Pareto front of ZDT3 seems to be a difficulty for MOISA-HV. It is worth noting that, when the Pareto front is continuous, the best parameter to control the number of clones is low. (See Table 1)

5.2. Bi-objectives constrained problems

Results show that MOISA-HV performs again well on problems which have a continuous Pareto front. Problem OSY seems to be difficult to solve for MOISA-HV. The reason could be the selection of solutions based on the Hypervolume contribution. Indeed, depending on the shape of the Pareto front, the algorithm may discard individuals belonging to a part of the curve with a very steep slope. (See Table 2)

Name	# Variables	# Objectives	# Constraints	Score(NSGAII)	Score(MOISA-HV)
ZDT1	30	2	0	0	5
ZDT2	30	2	0	0	5
ZDT3	30	2	0	2	3
ZDT4	10	2	0	5	0
ZDT6	10	2	0	0	5
KUR	3	2	0	0	5
SCH1	1	2	0	1	4
SCH2	1	2	0	0	5

Table 1: Bi-objective unconstrained problems

Name	# Variables	# Objectives	# Constraints	Score(NSGAII)	Score(MOISA-HV)
BNH	2	2	2	0	5
OSY	6	2	6	5	0
SRN	2	2	2	0	5
TNK	2	2	2	2	3

Table 2: Bi-objectives constrained problems

5.3. Three objectives problems

On three objectives problems, once again the algorithm performs well and the graphical results show that convergence through the Pareto front is achieved. The DTLZ class problems are known to be hard to solve. Nevertheless, in 4 DTLZ problems, some indicators show that MOISA-HV can perform better, among them two problems are significantly better. The comparison with other MOAIS algorithms is difficult since only a few results on DTLZ problems have been provided in the past. (See Table 3)

Those results show that MOISA-HV is competitive even if using very basic features. Hard problems obviously give worse results but adding new methods taken from the immune system metaphor should significantly improve these results. Implementing an immune system memory seems to be relevant in order to avoid cloning candidates in bad regions of the objective space. Another idea would be to use a

Name	# Variables	# Objectives	# Constraints	Score(NSGAII)	Score(MOISA-HV)
BNH4	2	3	2	0	5
VNT1	2	3	0	0	5
VNT2	2	3	0	1	4
VNT3	2	3	0	0	5
DTLZ1	12	3	0	5	0
DTLZ2	12	3	0	1	4
DTLZ3	12	3	0	4	1
DTLZ4	12	3	0	3	2
DTLZ7	22	3	0	1	4

Table 3: Three objectives problems

diversity function to add some random elements ”between” two good solutions. These solutions would be non-dominated solutions that were suppressed previously by the hypervolume discard process.

6. Conclusion

In this document, a new algorithm has been presented. The aim was to investigate the competitivity of the proposed approach with respect to state-of-the-art MOEA. MOAIS are recent algorithms that have already shown to be efficient. Here, the new idea was to combine basics of MOAIS with Hypervolume, which is known to have good characteristics for achieving convergence and spread. Comparisons on population based-algorithms is still nowadays a hard task and many papers don't follow a standard methodology to present results, which is why some frameworks have recently appeared, such as ParadisEO-MOEO[22] or Jmetal[23]. Nevertheless, these frameworks are either quite complicated to handle or written in a language that is considered ill-adapted for optimization. In this paper, the choice was made to implement a robust and maintainable algorithm in C, as well as a comparator that can be applied to further investigation on other algorithms.

The results show that the new algorithm performs well on a large set of problems even while using very simple features from MOAIS. Moreover, the hypervolume contribution discard process was simplified with the purpose of limiting the complexity of the algorithm. This obviously decreases the final population quality. Many improvements are expected on population-based algorithms working with an online hypervolume indicator, especially working with 3 or more objectives. This algorithm seems to be the first MOAIS using Hypervolume contribution, and many drawbacks still remain to be studied:

- Among the 4 parameters, at least 2 have to be tuned to fit specific problems (GG stepsize and GL stepsize). Nevertheless, some experiments showed that categories of problems share the same parameters.
- Difficulties while solving some problems (ZDT4, OSY, DTLZ)
- Local Gaussian search can lead to a loss of diversity, especially on three objectives problems.

In the litterature, only 2 papers have been found providing results on DTLZ problems solved via Immune System Algorithms. These algorithms are not true Immune System Algorithms because they use a crossover operator, which is quite a powerful feature. Therefore, this algorithm can be considered as the first one that provides results (that are reasonably good) on difficult problems as a true MOAIS. An analysis of the results with an exhaustive search on parameters still has to be made. It is almost certain that bounds of the parameters have to be redefined. Furthermore, many improvements to this algorithm can be envisaged as Artificial Immune System algorithms can offer many more methods taken as a metaphor from the true immune system. The next paragraph gives the main ideas about the work that was still being planned when starting to write this document, and some other investigations that could be done in a further work.

6.1. Future work

- Investigate self-adaptive parameters.

  The algorithms being run with basic parameters always reaches the Pareto front after a few generations, but can be much more efficient if the set of parameters is well-chosen. After some experiments, it appears that the parameter defining the number of clones is the one that has the least influence regarding the final quality of the population. The mutation steps size is a parameter that should be self-adapting. The parameter that controls the mutation type may be self-adaptive as well starting

with a high probability to do global search and trying to detect when the algorithm reaches the Pareto front to increase local Gaussian probability.

- Changing the mutation probability

  In population-based algorithms, $\frac{1}{nreal}$ is commonly used as the mutation probability. It has been shown that this value can be increased at the begining of the search to explore faster and be more efficient. Statistics on previous good mutation values can also be used to guide the search with variables that participate the most in improving the set of solutions.

- Improving Diversity

  The diversity feature presented in the canonical algorithm is not used here. For some problems, not using recombination leads to a significant loss of diversity. The reason can be easily understood by comparing the selection, proliferation and mutation of candidates as simple local searches. To avoid such behavior in the algorithm, random individuals (or from a previous generation) should be created and used as candidates to enter the archive.

- Non-uniform mutation

  Non-uniform mutation has given good results in genetic algorithms. It would be very easy to apply a coefficient to the Gaussian mutation, local, global or both and to try some experiments in order to see if it worth doing. If a method detecting the convergence to the true Pareto front is found, then one could envisage starting a non-uniform mutation by decreasing the step-size mutation after the detection.

- Comparator

  Up to now, the comparator has given good numerical values when comparing two algorithms with a large set of indicators. Nevertheless, the algorithm was implemented quickly and more work could be done to improve its design. Later, the algorithm will be able to generate more results, faster, and will be capable of handling a larger number of runs for experiments. Also, the convergence rate is an indicator that still has to be implemented. This new operator could be used with the evolution of the hypervolume throughout the generations, or the inverted generationnal distance.

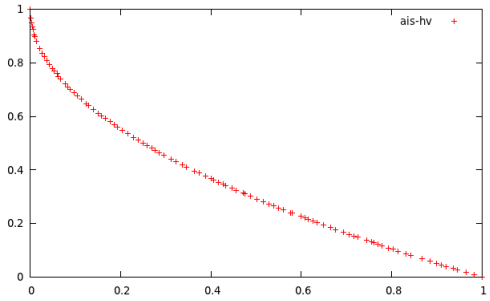
7. Aknowledgments

This work gave me the great opportunity to come to Mexico, to take advantage of the experience of the Multi objective Optimization team in CINVESTAV, and also to discover a beautiful country, full of contrasts and so different from what I knew before. I would like to thank Dr Carlos A. Coello Coello for his invitation, his disponibility, and for providing me with such an original and interesting subject. Thanks again to Dr Carlos A. Coello Coello and the UMI LAFMIA 3175 CNRS for providing financial support, which was of great help during my stay. I would also like to thank Alfredo Arias, Dr A. López Jaimes, and Saúl Zapotecas for our discussions about my work, for their constant kindness and their help for many little things that made my stay in Mexico a real pleasure. Finally, I would like to thank Eric RendallDay, who reviewed this document more than once and my mother for her support, moral as well as material.

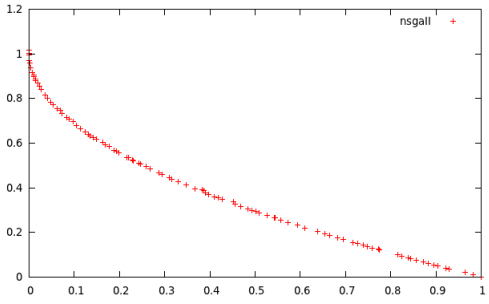
Appendix A. Problem ZDT1

Table A.4: ZDT1 problem with parameters $a = 0.6$, $b = 1.0$, $c = 1.0$, $d = 0.0$

PROB=zdt1, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	8.972159e-01	-
HYPERVOLUME (A1):	8.889857e-01	5.097441e-04
HYPERVOLUME (A2):	**8.924710e-01**	**1.428605e-04**
Inv-Gen-Dst (A1):	1.981111e-04	1.007348e-05
Inv-Gen-Dst (A2):	**1.678874e-04**	**8.989136e-06**
COV2SETS (A1/A2):	7.100000e-01	**8.977193e-01**
COV2SETS (A2/A1):	**3.166000e+01**	6.638102e+00
COV2SETS (PF/A1):	3.295500e+02	4.710082e+01
COV2SETS (PF/A2):	**2.716000e+01**	**1.405327e+01**
SPREAD (A1):	2.543682e-01	5.710559e-04
SPREAD (A2):	**1.916463e-01**	**2.677102e-04**



(a) Ais-HV



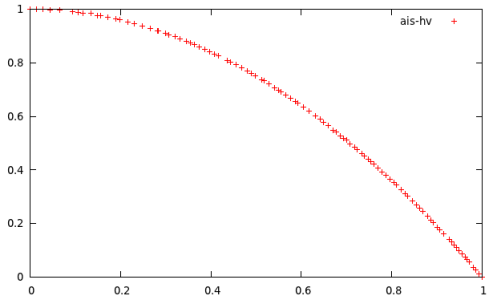
(b) NSGA-II

Figure A.7: Generation 200 on Ais-HV (a), and NSGA-II (b)

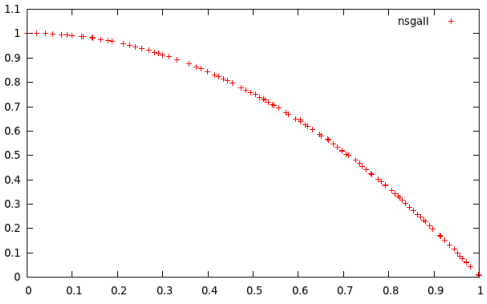
Appendix B. Problem ZDT2

Table B.5: ZDT2 problem with parameters $a = 0.4$, $b = 0.8$, $c = 0.4$, $d = 0.0$

PROB=zdt2, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	5.764303e-01	-
HYPERVOLUME (A1):	5.675995e-01	5.716423e-04
HYPERVOLUME (A2):	**5.717070e-01**	**1.715964e-04**
Inv-Gen-Dst (A1):	1.966293e-04	**8.388726e-06**
Inv-Gen-Dst (A2):	**1.632170e-04**	9.104609e-06
COV2SETS (A1/A2):	7.000000e-01	**9.110434e-01**
COV2SETS (A2/A1):	**4.265000e+01**	7.250345e+00
COV2SETS (PF/A1):	4.211600e+02	6.013281e+01
COV2SETS (PF/A2):	**2.152000e+01**	**1.253434e+01**
SPREAD (A1):	2.511818e-01	4.924860e-04
SPREAD (A2):	**1.773486e-01**	**2.244342e-04**



(a) Ais-HV



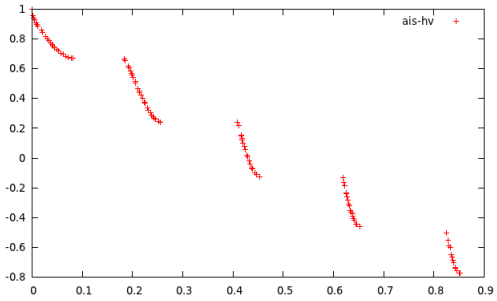
(b) NSGA-II

Figure B.8: Generation 200 on Ais-HV (a), and NSGA-II (b)

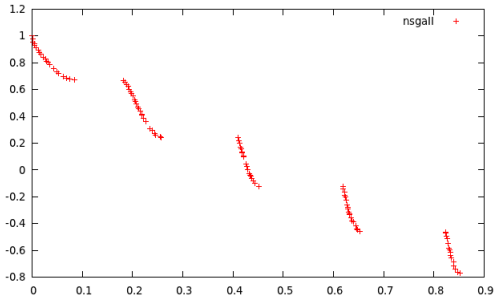
Appendix C. Problem ZDT3

Table C.6: ZDT3 problem with parameters $a = 0.6$, $b = 1.0$, $c = 0.0$, $d = 0.6$

PROB=zdt3, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	1.121989e+00	-
HYPERVOLUME (A1):	1.115808e+00	3.356911e-03
HYPERVOLUME (A2):	**1.119018e+00**	**2.652612e-04**
Inv-Gen-Dst (A1):	**2.619783e-04**	3.233972e-04
Inv-Gen-Dst (A2):	3.662419e-04	**1.237625e-04**
COV2SETS (A1/A2):	2.640000e+00	**3.024963e+00**
COV2SETS (A2/A1):	**3.014000e+01**	5.856654e+00
COV2SETS (PF/A1):	3.061900e+02	4.153834e+01
COV2SETS (PF/A2):	**7.464000e+01**	**3.494897e+01**
SPREAD (A1):	**4.426133e-01**	**3.096580e-04**
SPREAD (A2):	5.278173e-01	1.216346e-03



(a) Ais-HV



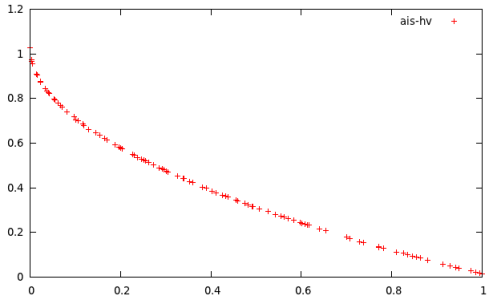
(b) NSGA-II

Figure C.9: Generation 200 on Ais-HV (a), and NSGA-II (b)

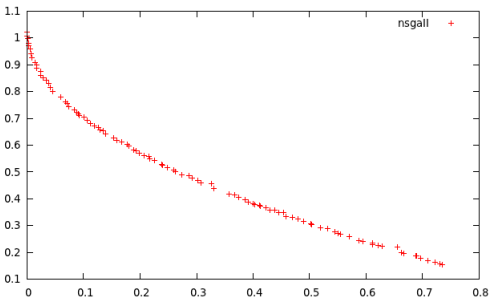
Appendix D. Problem ZDT4

Table D.7: ZDT4 problem with parameters $a = 1.0$, $b = 0.4$, $c = 0.6$, $d = 0.0$

PROB=zdt4, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	4.243026e+00	-
HYPERVOLUME (A1):	**4.218502e+00**	**1.852282e-02**
HYPERVOLUME (A2):	4.181431e+00	2.965293e-02
Inv-Gen-Dst (A1):	**6.799886e-04**	8.813995e-04
Inv-Gen-Dst (A2):	1.287022e-03	**6.623777e-04**
COV2SETS (A1/A2):	**3.742900e+02**	2.757433e+02
COV2SETS (A2/A1):	1.759000e+01	**6.398517e+01**
COV2SETS (PF/A1):	**1.458580e+03**	**1.131187e+03**
COV2SETS (PF/A2):	5.003750e+03	2.625216e+03
SPREAD (A1):	**3.025505e-01**	2.326102e-02
SPREAD (A2):	4.640457e-01	**9.048357e-03**



(a) Ais-HV



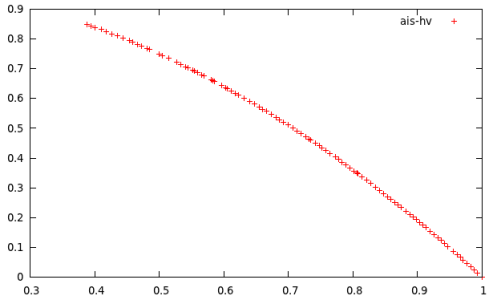
(b) NSGA-II

Figure D.10: Generation 200 on Ais-HV (a), and NSGA-II (b)

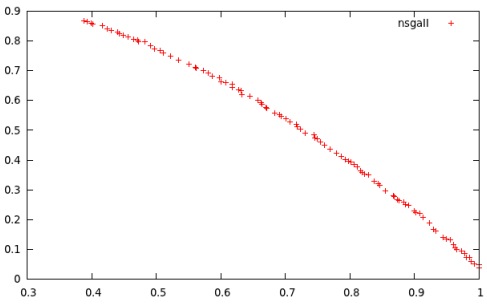
Appendix E. Problem ZDT6

Table E.8: ZDT6 problem with parameters $a = 0.2$, $b = 0.4$, $c = 0.6$, $d = 0.4$

PROB=zdt6, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	3.681806e-01	-
HYPERVOLUME (A1):	3.479779e-01	2.363901e-03
HYPERVOLUME (A2):	**3.662232e-01**	**4.861708e-05**
Inv-Gen-Dst (A1):	4.907628e-04	6.094925e-05
Inv-Gen-Dst (A2):	**1.107655e-04**	**2.752997e-06**
COV2SETS (A1/A2):	0.000000e+00	**0.000000e+00**
COV2SETS (A2/A1):	**3.119400e+02**	3.995768e+01
COV2SETS (PF/A1):	3.071790e+03	4.026863e+02
COV2SETS (PF/A2):	**0.000000e+00**	**0.000000e+00**
SPREAD (A1):	2.455140e-01	4.993801e-04
SPREAD (A2):	**1.682690e-01**	**2.354683e-04**



(a) Ais-HV



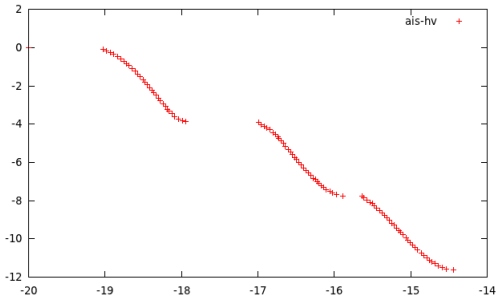
(b) NSGA-II

Figure E.11: Generation 200 on Ais-HV (a), and NSGA-II (b)

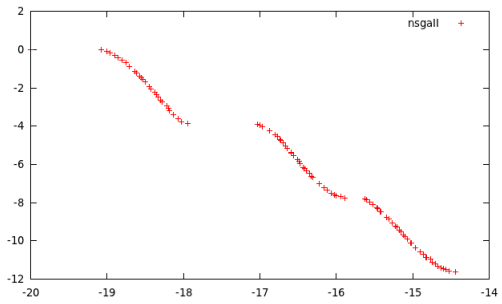
Appendix F. Problem KUR

Table F.9: KUR problem with parameters $a = 1.0$, $b = 0.4$, $c = 0.0$, $d = 0.8$

PROB=kur, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	4.038610e+01	-
HYPERVOLUME (A1):	4.005438e+01	1.835585e-02
HYPERVOLUME (A2):	**4.015592e+01**	**8.009187e-03**
Inv-Gen-Dst (A1):	1.629555e-03	5.685489e-05
Inv-Gen-Dst (A2):	**1.259808e-03**	**2.280523e-05**
COV2SETS (A1/A2):	4.190000e+00	**2.571750e+00**
COV2SETS (A2/A1):	**1.917000e+01**	4.384187e+00
COV2SETS (PF/A1):	2.316300e+02	3.525667e+01
COV2SETS (PF/A2):	**9.240000e+01**	**1.690207e+01**
SPREAD (A1):	3.399824e-01	3.880348e-04
SPREAD (A2):	**2.489354e-01**	**1.503382e-04**



(a) Ais-HV



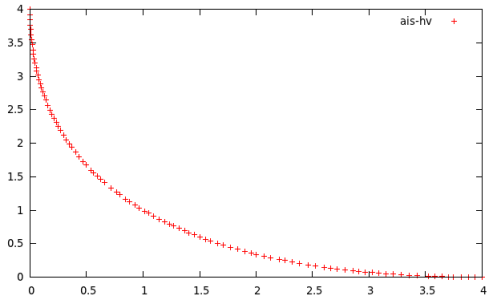
(b) NSGA-II

Figure F.12: Generation 200 on Ais-HV (a), and NSGA-II (b)

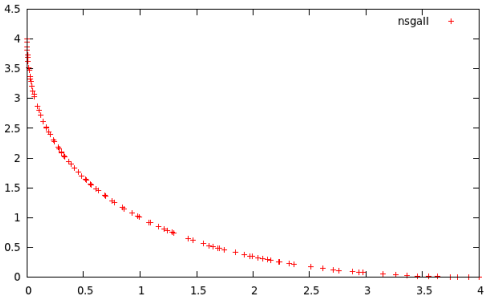
Appendix G. Problem SCH1

Table G.10: SCH1 problem with parameters $a = 0.2$, $b = 0.8$, $c = 0.0$, $d = 0.0$

PROB=sch1, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	1.673083e+01	-
HYPERVOLUME (A1):	1.666395e+01	3.756141e-03
HYPERVOLUME (A2):	**1.667902e+01**	**6.385606e-04**
Inv-Gen-Dst (A1):	8.566620e-04	4.678536e-05
Inv-Gen-Dst (A2):	**6.259442e-04**	**6.406047e-06**
COV2SETS (A1/A2):	**5.000000e-01**	7.141428e-01
COV2SETS (A2/A1):	3.500000e-01	**6.538348e-01**
COV2SETS (PF/A1):	1.030000e+00	8.769835e-01
COV2SETS (PF/A2):	**7.800000e-01**	**6.720119e-01**
SPREAD (A1):	2.848059e-01	7.347945e-04
SPREAD (A2):	**1.124959e-01**	**8.237493e-05**



(a) Ais-HV



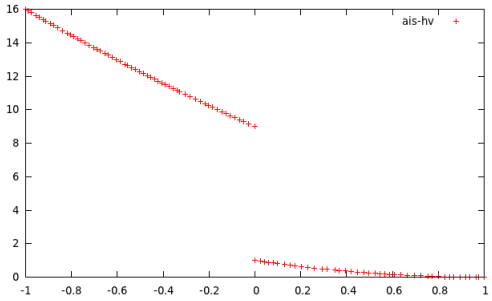
(b) NSGA-II

Figure G.13: Generation 200 on Ais-HV (a), and NSGA-II (b)

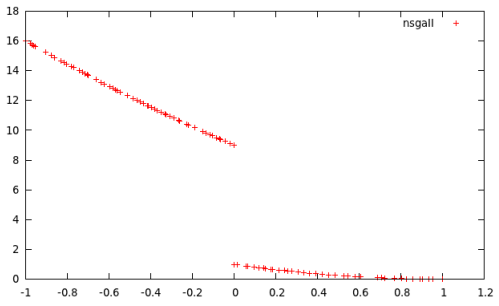
Appendix H. Problem SCH2

Table H.11: SCH2 problem with parameters $a = 0.6$, $b = 0.6$, $c = 0.0$, $d = 0.6$

PROB=sch2, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	2.606859e+01	-
HYPERVOLUME (A1):	2.598037e+01	4.319898e-03
HYPERVOLUME (A2):	**2.600114e+01**	**1.003242e-03**
Inv-Gen-Dst (A1):	1.266999e-03	8.120261e-05
Inv-Gen-Dst (A2):	**9.617711e-04**	**2.013877e-05**
COV2SETS (A1/A2):	3.900000e-01	**6.147357e-01**
COV2SETS (A2/A1):	**1.320000e+00**	1.340746e+00
COV2SETS (PF/A1):	1.950000e+00	1.251998e+00
COV2SETS (PF/A2):	**1.250000e+00**	**8.645808e-01**
SPREAD (A1):	9.707172e-01	1.542839e-04
SPREAD (A2):	**9.529271e-01**	**1.520728e-05**



(a) Ais-HV



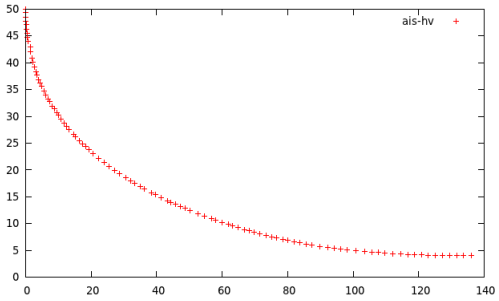
(b) NSGA-II

Figure H.14: Generation 200 on Ais-HV (a), and NSGA-II (b)

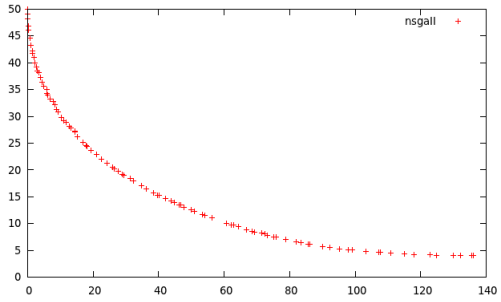
Appendix I. Problem BNH

Table I.12: BNH problem with parameters $a = 0.2$, $b = 0.8$, $c = 1.0$, $d = 0.8$

PROB=bnh, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	6.411704e+03	-
HYPERVOLUME (A1):	6.379973e+03	1.819580e+00
HYPERVOLUME (A2):	**6.389734e+03**	**4.076710e-01**
Inv-Gen-Dst (A1):	2.068579e-02	1.247548e-03
Inv-Gen-Dst (A2):	**1.568725e-02**	**2.579401e-04**
COV2SETS (A1/A2):	1.520000e+00	**1.268700e+00**
COV2SETS (A2/A1):	**1.565000e+01**	3.592701e+00
COV2SETS (PF/A1):	1.572200e+02	2.875990e+01
COV2SETS (PF/A2):	**2.544000e+01**	**7.864248e+00**
SPREAD (A1):	3.959604e-01	1.256399e-03
SPREAD (A2):	**3.222677e-01**	**2.594123e-04**



(a) Ais-HV



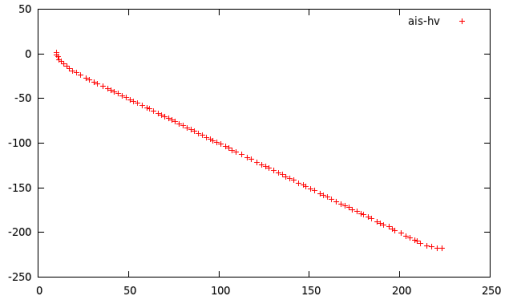
(b) NSGA-II

Figure I.15: Generation 200 on Ais-HV (a), and NSGA-II (b)

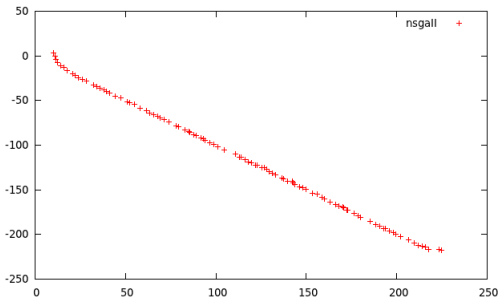
Appendix J. Problem SRN

Table J.13: SRN problem with parameters $a = 1.0$, $b = 0.2$, $c = 0.2$, $d = 1.0$

PROB=srn, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	3.703809e+04	-
HYPERVOLUME (A1):	3.671138e+04	1.609865e+01
HYPERVOLUME (A2):	**3.681206e+04**	**3.998687e+00**
Inv-Gen-Dst (A1):	4.003212e-02	1.856587e-03
Inv-Gen-Dst (A2):	**3.041670e-02**	**3.686325e-04**
COV2SETS (A1/A2):	1.160000e+00	**1.036533e+00**
COV2SETS (A2/A1):	**1.348000e+01**	3.474133e+00
COV2SETS (PF/A1):	1.438800e+02	2.573608e+01
COV2SETS (PF/A2):	**2.937000e+01**	**7.929256e+00**
SPREAD (A1):	2.508401e-01	3.914518e-04
SPREAD (A2):	**1.190118e-01**	**9.635186e-05**



(a) Ais-HV



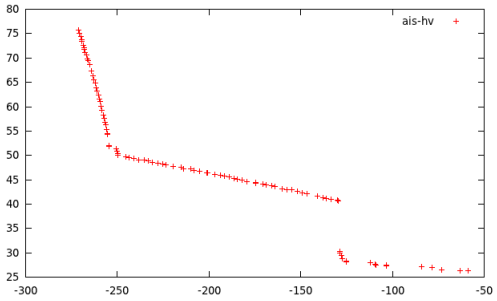
(b) NSGA-II

Figure J.16: Generation 200 on Ais-HV (a), and NSGA-II (b)

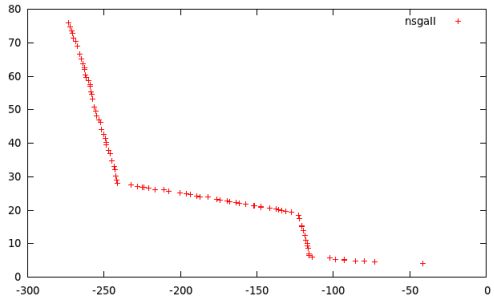
Appendix K. Problem OSY

Table K.14: OSY problem with parameters $a = 1.0$, $b = 0.8$, $c = 0.6$, $d = 0.6$

PROB=osy, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	2.121310e+04	-
HYPERVOLUME (A1):	**2.090325e+04**	**2.443592e+02**
HYPERVOLUME (A2):	1.986941e+04	1.785109e+03
Inv-Gen-Dst (A1):	**1.270094e-01**	**8.061315e-02**
Inv-Gen-Dst (A2):	2.090655e-01	1.647553e-01
COV2SETS (A1/A2):	**5.675200e+02**	8.568820e+02
COV2SETS (A2/A1):	1.713000e+01	**1.303507e+01**
COV2SETS (PF/A1):	**6.410300e+02**	**2.242667e+02**
COV2SETS (PF/A2):	5.463970e+03	7.502850e+03
SPREAD (A1):	**5.510363e-01**	3.033283e-03
SPREAD (A2):	5.571633e-01	**2.665801e-03**



(a) Ais-HV



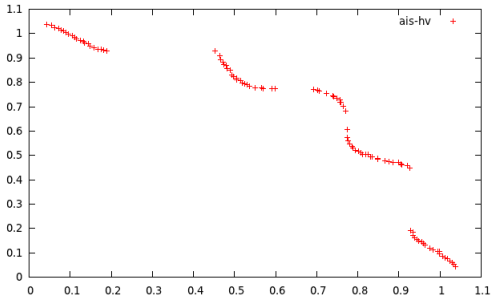
(b) NSGA-II

Figure K.17: Generation 200 on Ais-HV (a), and NSGA-II (b)

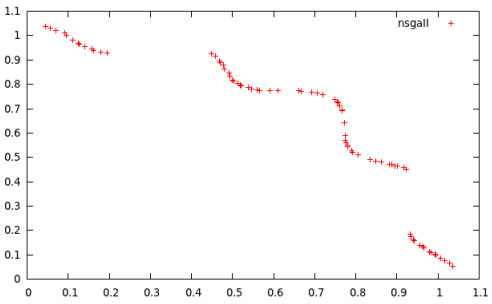
Appendix L. Problem TNK

Table L.15: TNK problem with parameters $a = 1.0$, $b = 0.2$, $c = 0.0$, $d = 0.8$

PROB=tnk, A1=NSGAII, A2=AIS-HV POPSIZE=100, PFSIZE = 1000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	5.335408e-01	-
HYPERVOLUME (A1):	5.273578e-01	8.032265e-04
HYPERVOLUME (A2):	**5.296092e-01**	**5.341263e-04**
Inv-Gen-Dst (A1):	**2.870605e-04**	**5.937704e-05**
Inv-Gen-Dst (A2):	3.477924e-04	2.503430e-04
COV2SETS (A1/A2):	1.648000e+01	**4.290641e+00**
COV2SETS (A2/A1):	**2.360000e+01**	5.734108e+00
COV2SETS (PF/A1):	3.573000e+02	4.961965e+01
COV2SETS (PF/A2):	**2.929100e+02**	**4.547771e+01**
SPREAD (A1):	**7.239077e-01**	**1.604295e-03**
SPREAD (A2):	7.843828e-01	2.038829e-03



(a) Ais-HV



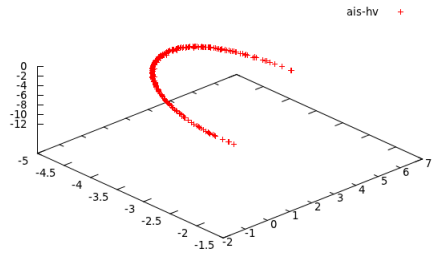
(b) NSGA-II

Figure L.18: Generation 200 on Ais-HV (a), and NSGA-II (b)

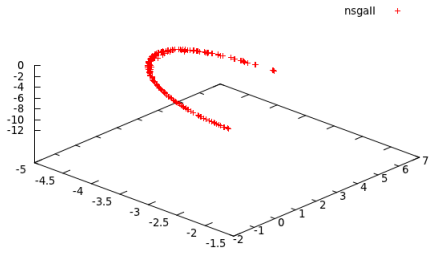
Appendix M. Problem BNH4

Table M.16: BNH4 problem with parameters $a = 1.0$, $b = 0.0$, $c = 0.0$, $d = 1.0$

PROB=bnh4, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	1.666199e+02	-
HYPERVOLUME (A1):	1.642451e+02	1.365045e-01
HYPERVOLUME (A2):	**1.650376e+02**	**6.789279e-02**
Inv-Gen-Dst (A1):	1.208248e-03	7.449092e-05
Inv-Gen-Dst (A2):	**8.074322e-04**	**3.771509e-05**
COV2SETS (A1/A2):	2.494000e+01	**5.408919e+00**
COV2SETS (A2/A1):	**5.672000e+01**	7.973807e+00
COV2SETS (PF/A1):	8.554600e+02	7.874178e+01
COV2SETS (PF/A2):	**5.345600e+02**	**4.548809e+01**
SPREAD (A1):	3.629387e-01	3.498537e-04
SPREAD (A2):	**3.054108e-01**	**2.902681e-04**



(a) Ais-HV



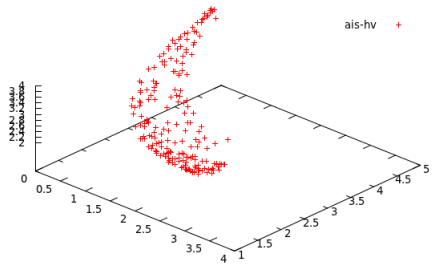
(b) NSGA-II

Figure M.19: Generation 500 on Ais-HV (a), and NSGA-II (b)

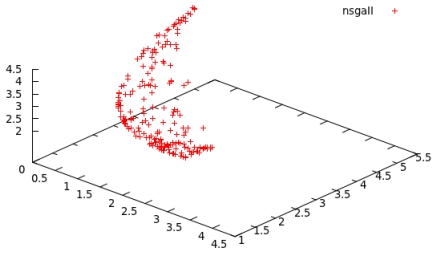
Appendix N. Problem VNT1

Table N.17: VNT1 problem with parameters $a = 0.0$, $b = 1.0$, $c = 1.0$, $d = 0.8$

PROB=vnt1, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	3.915465e+01	-
HYPERVOLUME (A1):	3.843549e+01	4.400205e-02
HYPERVOLUME (A2):	**3.846757e+01**	**4.315937e-02**
Inv-Gen-Dst (A1):	2.492565e-03	**9.407104e-05**
Inv-Gen-Dst (A2):	**2.443735e-03**	9.419610e-05
COV2SETS (A1/A2):	4.490000e+00	**2.670187e+00**
COV2SETS (A2/A1):	**8.540000e+00**	3.389454e+00
COV2SETS (PF/A1):	7.466000e+01	**1.849823e+01**
COV2SETS (PF/A2):	**4.618000e+01**	2.083861e+01
SPREAD (A1):	3.682286e-01	**1.205947e-02**
SPREAD (A2):	**3.194453e-01**	1.368506e-02



(a) Ais-HV



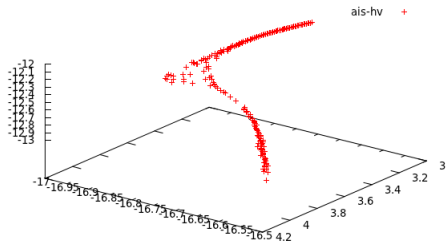
(b) NSGA-II

Figure N.20: Generation 500 on Ais-HV (a), and NSGA-II (b)

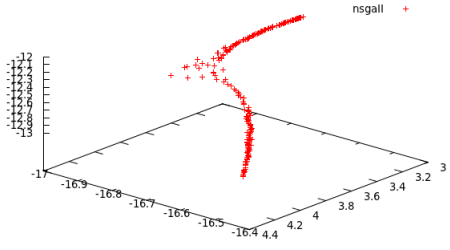
Appendix O. Problem VNT2

Table O.18: VNT2 problem with parameters $a = 0.0$, $b = 0.4$, $c = 0.8$, $d = 0.8$

PROB=vnt2, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	9.753846e-01	-
HYPERVOLUME (A1):	9.713098e-01	5.181028e-04
HYPERVOLUME (A2):	**9.725243e-01**	**3.363723e-04**
Inv-Gen-Dst (A1):	**2.117062e-04**	**1.616032e-05**
Inv-Gen-Dst (A2):	7.047082e-04	2.868213e-04
COV2SETS (A1/A2):	1.009000e+01	**4.730951e+00**
COV2SETS (A2/A1):	**1.811000e+01**	4.890593e+00
COV2SETS (PF/A1):	3.126100e+02	4.518582e+01
COV2SETS (PF/A2):	**1.468300e+02**	**4.346747e+01**
SPREAD (A1):	2.683721e-01	**2.784821e-04**
SPREAD (A2):	**2.620417e-01**	3.959834e-04



(a) Ais-HV



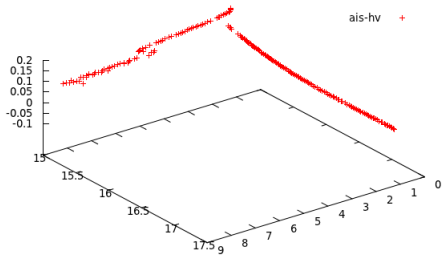
(b) NSGA-II

Figure O.21: Generation 500 on Ais-HV (a), and NSGA-II (b)

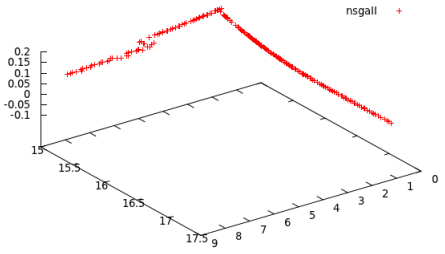
Appendix P. Problem VNT3

Table P.19: VNT3 problem with parameters $a = 0.6$, $b = 0.2$, $c = 0.2$, $d = 0.8$

PROB=vnt3, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	5.362486e+00	-
HYPERVOLUME (A1):	5.343830e+00	1.001768e-03
HYPERVOLUME (A2):	**5.347744e+00**	**9.577516e-04**
Inv-Gen-Dst (A1):	7.165877e-04	1.512673e-04
Inv-Gen-Dst (A2):	**5.077640e-04**	**3.849537e-05**
COV2SETS (A1/A2):	5.810000e+00	**3.022234e+00**
COV2SETS (A2/A1):	**2.534000e+01**	5.503126e+00
COV2SETS (PF/A1):	2.896800e+02	4.698678e+01
COV2SETS (PF/A2):	**9.728000e+01**	**2.612014e+01**
SPREAD (A1):	6.957154e-01	7.535019e-04
SPREAD (A2):	**6.638902e-01**	**6.407286e-04**



(a) Ais-HV



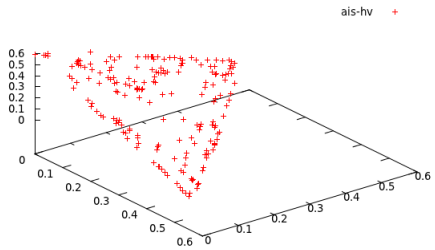
(b) NSGA-II

Figure P.22: Generation 500 on Ais-HV (a), and NSGA-II (b)

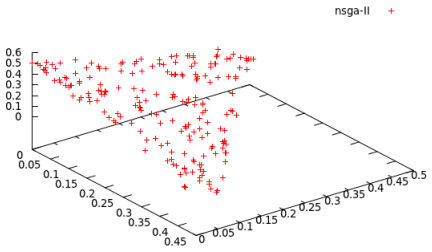
Appendix Q. Problem DTLZ1

Table Q.20: DTLZ1 problem with parameters $a = 0.0$, $b = 1.0$, $c = 1.0$, $d = 0.8$

PROB=dtlz1, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	1.456069e+00	-
HYPERVOLUME (A1):	**1.425056e+00**	3.987393e-02
HYPERVOLUME (A2):	1.422801e+00	**1.218025e-02**
Inv-Gen-Dst (A1):	**1.216457e-03**	1.201047e-03
Inv-Gen-Dst (A2):	1.997958e-03	**5.495831e-04**
COV2SETS (A1/A2):	**1.922010e+03**	**1.076029e+03**
COV2SETS (A2/A1):	2.183000e+02	1.080882e+03
COV2SETS (PF/A1):	**3.723810e+03**	1.791160e+04
COV2SETS (PF/A2):	2.046786e+04	**1.006530e+04**
SPREAD (A1):	**3.517348e-01**	7.443500e-03
SPREAD (A2):	4.163033e-01	**2.365465e-03**



(a) Ais-HV



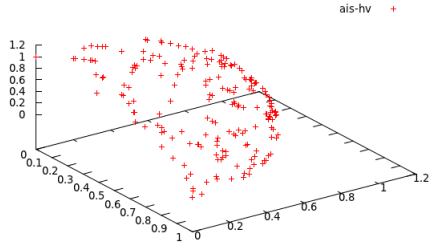
(b) NSGA-II

Figure Q.23: Generation 500 on Ais-HV (a), and NSGA-II (b)

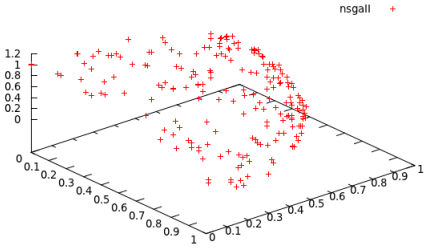
Appendix R. Problem DTLZ2

Table R.21: DTLZ2 problem with parameters $a = 0.0$, $b = 1.0$, $c = 0.2$, $d = 1.0$

PROB=dtlz2, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	1.204409e+00	-
HYPERVOLUME (A1):	1.144841e+00	**4.590454e-03**
HYPERVOLUME (A2):	**1.161533e+00**	8.947158e-03
Inv-Gen-Dst (A1):	**1.230231e-03**	**4.686928e-05**
Inv-Gen-Dst (A2):	1.257585e-03	6.905440e-05
COV2SETS (A1/A2):	1.070000e+00	**1.274794e+00**
COV2SETS (A2/A1):	**1.221000e+01**	4.554767e+00
COV2SETS (PF/A1):	8.351000e+01	2.201794e+01
COV2SETS (PF/A2):	**1.514000e+01**	**8.122832e+00**
SPREAD (A1):	2.564649e-01	**2.455185e-04**
SPREAD (A2):	**2.055064e-01**	3.284170e-04



(a) Ais-HV



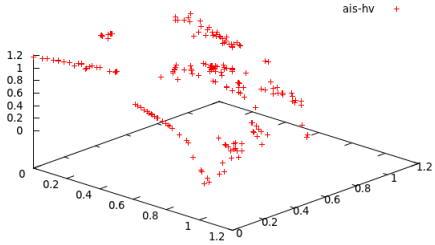
(b) NSGA-II

Figure R.24: Generation 500 on Ais-HV (a), and NSGA-II (b)

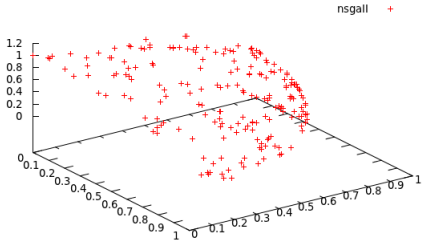
Appendix S. Problem DTLZ3

Table S.22: DTLZ3 problem with parameters $a = 0.8$, $b = 0.6$, $c = 0.8$, $d = 0.6$

PROB=dtlz3, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	3.100326e+05	-
HYPERVOLUME (A1):	3.100281e+05	**2.691994e+00**
HYPERVOLUME (A2):	**3.100302e+05**	1.161351e+01
Inv-Gen-Dst (A1):	**1.285379e-03**	**6.793484e-05**
Inv-Gen-Dst (A2):	8.142901e-03	4.594056e-03
COV2SETS (A1/A2):	**4.008970e+03**	3.367163e+03
COV2SETS (A2/A1):	2.130000e+00	**8.877674e+00**
COV2SETS (PF/A1):	**2.408900e+02**	**2.803561e+02**
COV2SETS (PF/A2):	4.157050e+04	3.382124e+04
SPREAD (A1):	**3.529938e-01**	**7.582437e-02**
SPREAD (A2):	8.583327e-01	1.190171e-01



(a) Ais-HV



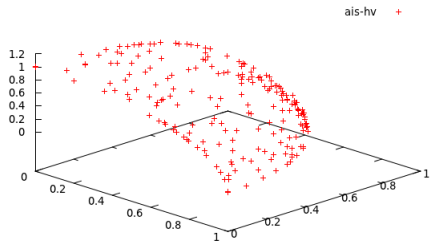
(b) NSGA-II

Figure S.25: Generation 500 on Ais-HV (a), and NSGA-II (b)

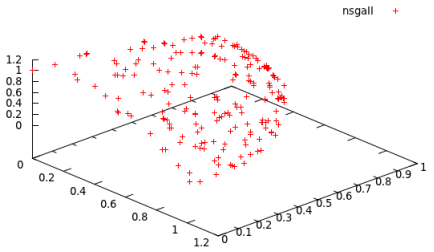
Appendix T. Problem DTLZ4

Table T.23: DTLZ4 problem with parameters $a = 0.8$, $b = 0.8$, $c = 0.8$, $d = 0.2$

PROB=dtlz4, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	9.796691e-01	-
HYPERVOLUME (A1):	**9.239092e-01**	**4.660974e-03**
HYPERVOLUME (A2):	9.171880e-01	5.370789e-02
Inv-Gen-Dst (A1):	**1.194994e-03**	**4.561336e-05**
Inv-Gen-Dst (A2):	3.289257e-03	3.309546e-03
COV2SETS (A1/A2):	1.410000e+00	**1.667903e+00**
COV2SETS (A2/A1):	**2.292000e+01**	1.185216e+01
COV2SETS (PF/A1):	7.976000e+01	1.893416e+01
COV2SETS (PF/A2):	**1.917000e+01**	**1.508844e+01**
SPREAD (A1):	2.746856e-01	**4.038541e-04**
SPREAD (A2):	**1.924270e-01**	8.777892e-04



(a) Ais-HV



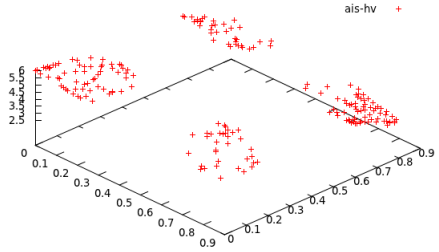
(b) NSGA-II

Figure T.26: Generation 500 on Ais-HV (a), and NSGA-II (b)

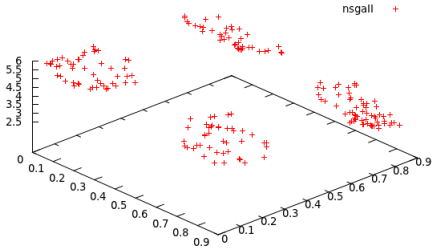
Appendix U. Problem DTLZ7

Table U.24: DTLZ7 problem with parameters $a = 0.0$, $b = 1.0$, $c = 1.0$, $d = 0.4$

PROB=dtlz7, A1=NSGAII, A2=AIS-HV POPSIZE=200, PFSIZE = 2000, RUNS=100		
INDICATOR	MEAN	Std. Dev.
HYPERVOLUME (PF)	2.130413e+00	-
HYPERVOLUME (A1):	2.045854e+00	**6.288492e-03**
HYPERVOLUME (A2):	**2.058288e+00**	9.128645e-03
Inv-Gen-Dst (A1):	**1.259613e-03**	**7.820159e-05**
Inv-Gen-Dst (A2):	1.334508e-03	1.389308e-04
COV2SETS (A1/A2):	3.210000e+00	**2.346465e+00**
COV2SETS (A2/A1):	**2.769000e+01**	6.778931e+00
COV2SETS (PF/A1):	2.295600e+02	4.502406e+01
COV2SETS (PF/A2):	**4.558000e+01**	**2.256598e+01**
SPREAD (A1):	5.607393e-01	**1.161393e-03**
SPREAD (A2):	**5.151642e-01**	1.216795e-03



(a) Ais-HV



(b) NSGA-II

Figure U.27: Generation 500 on Ais-HV (a), and NSGA-II (b)

References

[1] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.

[2] F. Campelo, F. Guimarães, H. Igarashi, Overview of artificial immune systems for multi-objective optimization, in: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization, Vol. 4403 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 937–951.

[3] H. Mo, Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies (Volume 1) (Handbook of Research On...), Medical Information Science Reference, 2008.

[4] C. A. C. Coello, N. C. Cortés, Solving multiobjective optimization problems using an artificial immune system, Genetic Programming and Evolvable Machines 6 163–190.

[5] F. Freschi, M. Repetto, Multiobjective optimization by a modified artificial immune system algorithm, in: International Conference on Artificial Immune Systems, 2005, pp. 248–261.

[6] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Tech. Rep. 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001).

[7] L. Jiao, M. Gong, R. Shang, H. Du, B. Lu, Clonal selection with immune dominance and anergy based multiobjective optimization, in: C. Coello Coello, A. Hernndez Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization, Vol. 3410 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, pp. 474–489.

[8] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, Evol. Comput. 16 (2008) 225–255.

[9] B. Lu, L. Jiao, H. Du, M. Gong, Ifmoa: Immune forgetting multiobjective optimization algorithm., in: ICNC (3)'05, 2005, pp. 399–408.

[10] G. Coelho, F. Von Zuben, omni-ainet: An immune-inspired approach for omni optimization, in: H. Bersini, J. Carneiro (Eds.), Artificial Immune Systems, Vol. 4163 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006, pp. 294–308.

[11] N. Beume, B. Naujoks, M. Emmerich, Sms-emoa: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653 – 1669.

[12] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms - a comparative case study, in: A. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN V, Vol. 1498 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1998, pp. 292–301.

[13] M. Fleischer, The measure of pareto optima applications to multi-objective metaheuristics, in: C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, K. Deb (Eds.), Evolutionary Multi-Criterion Optimization, Vol. 2632 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 74–74.

[14] K. Bringmann, T. Friedrich, An efficient algorithm for computing hypervolume contributions, Evol. Comput. 18 (2010) 383–402.

[15] K. Bringmann, T. Friedrich, Don't be greedy when calculating hypervolume contributions, in: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms, FOGA '09, ACM, New York, NY, USA, 2009, pp. 103–112.

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

[17] J. L. Bentley, K. L. Clarkson, D. B. Levine, Fast linear expected-time algorithms for computing maxima and convex hulls, Algorithmica 9 (1993) 168–183.

[18] S. Yang, Statistics-based adaptive non-uniform mutation for genetic algorithms, in: E. Cant-Paz, J. Foster, K. Deb, L. Davis, R. Roy, U.-M. OReilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A. Schultz, K. Dowsland, N. Jonoska, J. Miller (Eds.), Genetic and Evolutionary Computation GECCO 2003, Vol. 2724 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 208–208.

[19] M.-R. Chen, Y.-Z. Lu, A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization, European Journal of Operational Research 188 (3) (2008) 637 – 651.

[20] M. López-Ibáñez, J. Knowles, M. Laumanns, On sequential online archiving of objective vectors, in: R. Takahashi, K. Deb, E. Wanner, S. Greco (Eds.), Evolutionary Multi-Criterion Optimization, Vol. 6576 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 46–60.

[21] C. M. Fonseca, L. Paquete, M. López-Ibáñez, An improved dimension-sweep algorithm for the hypervolume indicator, pp. 1157–1163.

[22] A. Liefooghe, L. Jourdan, E.-G. Talbi, A software framework based on a conceptual unified model for evolutionary multiobjective optimization: Paradiseo-moeo, European Journal of Operational Research 209 (2) (2011) 104 – 112.

[23] J. Durillo, A. Nebro, E. Alba, The jmetal framework for multi-objective optimization: Design and architecture, in: CEC 2010, Barcelona, Spain, 2010, pp. 4138–4325.