Adaptive Multi-objective Genetic Algorithm using Multi-Pareto-Ranking

ABSTRACT

This paper extends an elitist multi-objective evolutionary algorithm, named GAME, based on several Pareto fronts corresponding to various fitness definitions. An additional operator is defined to create an adaptive version of this algorithm, called aGAME. This new operator alternates different modes of exploration of the search place all along aGAME execution. Mode switching is controlled according to the values of two performance indicators, in order to maintain a good compromise between quality and diversity of the returned solutions. aGAME is compared with the previous version (GAME) and with the three best ranked algorithms of the CEC 2009 competition, using five bi-objective benchmarks and the rules of this competition. This experimental comparison shows that aGAME outperforms these four algorithms, which validate both the efficiency of the proposed dynamic adaptive operator and the algorithm performance.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods

General Terms

Algorithms, Experimentation

Keywords

Adaptive multi-objective evolutionary algorithm, Quality and diversity indicators, Pareto ranking

1. INTRODUCTION

Over the last decade, many research works dealing with evolutionary algorithms (EA) focused on multi-objective algorithms. They are often based on very famous algorithms using Pareto ranking and non-dominated archives. Besides, designing adaptive algorithms is one of the most studied topics in this area. Section 2 gives a brief survey on these subjects. In the same way, this paper proposes a new adaptive multi-objective algorithm, called aGAME. It extends the previous version of an archive-based multi-objective algorithm called Genetic Algorithm with Multiple parEto fronts (GAME) [1]. The main parts of GAME are succinctly described in Section 3. GAME lies on a ranking process in which population is separated in several Pareto fronts. Associated selection strategies and several fitness definitions are also used. A parametric study based on design of experiments allowed GAME to outperform three algorithms in an experimental comparison based on the procedure of CEC 2009 competition. In aGAME, an adaptive mechanism is added in order to improve both performances and robustness. This new contribution is presented in section 4. Both selection probability of non-dominated individuals and mutation rate vary during the evolutionary process. This makes aGAME regularly switch between four behavior modes: initialization, normal search, exploration and exploitation. These changes are activated according to the values of two metrics measuring both the quality and the diversity of individuals. Finally, in section 5, the resulting algorithm behavior is analyzed and compared to the behavior of the previous static version GAME. This shows the impact of the proposed dynamic adaptation operator (DAO). Besides, all constrained bi-objective benchmarks of the CEC 2009 competition are used to measure the provided benefits and to validate aGAME performance. Section 6 ends the paper by giving the main conclusions of this work and developing several promising prospects.

2. RELATED WORK

2.1 Elitist Pareto-based MOEAs

Many multi-objective evolutionary algorithms (MOEAs) use the concept of Pareto dominance to rank solutions and to apply selection strategies based on non-domination ranks. Generally, such algorithms are elitists: the best solutions (i.e. non-dominated solutions) are kept either in the population itself or in a separate archive. In the first case, they participate to reproduction process which guides the exploration of the search space towards interesting areas. But the number of non-dominated solutions might greatly increase with the number of objectives, which limits the number of places reserved for new individuals. Therefore, such algorithms generally use a specific operator to preserve diversity. **Elitist Non-dominated Sorting Genetic Al-**

gorithm (NSGA-II) [12], is certainly one of the most famous algorithms belonging to this first category. The second class of MOEAs uses an archive of non-dominated solutions, which does not necessary takes part in reproduction. Besides, even if they use dominance to define the fitness of individuals, they usually use indicators rather than ranking in the strict sense of Pareto fronts. **Strength Pareto Evolutionary Algorithm 2 (SPEA2)** [30], as famous as NSGA-II, is certainly the reference algorithm in this class of MOEAs, known in literature as Indicator-based evolutionary algorithms [29]. Therefore, several recent MOEAs were inspired either by NSGA-II or by SPEA2. This subsection can not give an exhaustive survey of such approaches. It briefly describes some of them, in order to give the main elements required to position GAME, the algorithm on which this contribution is based, in section 3. **NSGA-II** uses an elitist selection strategy for replacement (with respect to fitness and spread). For each generation, parents and offsprings are gathered in a mating pool. A fast non-dominated sorting approach is used to rank these individuals in subsequent Pareto fronts. Then the next generation is built by prefering the individuals with the lower non-domination ranks. In case of rank equality, the selection depends on a diversity preservation mechanism. A density measure, called crowding distance, is computed in the objective space. For each point, it represents the largest cuboid enclosing it without including any other point in the population. Thus, when two individuals have the same rank, the selection process favors the individual with the largest cuboid. The tournament used to select parents for reproduction is also based on this density measure. **SPEA2** preserves elitism by using an archive of non-dominated solutions. This set stores the best individuals found by the algorithm. But it also participates to genetic operations in order to guide the exploration of the search space. For each generation, the non-dominated solutions of the population are copied in this archive. This may eliminate some other solutions in the archive (if they are dominated by the entering ones). The fitness used in SPEA2 is based both on dominance and neighborhood density. The algorithm assigns a weight to each solution S, based on the number of non-dominated individuals (in the archive and in the current population) which dominate S. A density estimation is computed using the k-th nearest neighbor method [12]. Finally, the fitness of S sums this distance and the weights of the solutions which dominate S. Only solutions of the archive participate to the reproduction selection. More recently, **FastPGA (Fast Pareto Genetic Algorithm)** [9] combined the ideas introduced in NSGA-II and SPEA2. FastPGA ranks parents and offsprings in two sets: non-dominated solutions and dominated ones. For non-dominated solutions, fitness is computed by using the crowding distance defined in NSGA-II. For dominated solutions, the calculation of fitness seems like the one used in SPEA2. FastPGA generalizes the weight computation: the fitness of any dominated solution S is the sum of the weights of individuals dominating S minus the sum of the weights of individuals dominated by S. This allows to limit the number of solutions which have the same fitness value. Both non-domination rank and fitness are used in the selection for reproduction. Besides, an additional mechanism dyamically controls the population size. This allows FastPGA to keep good performances for optimization problems with complex objective functions (particularly those the evaluation of which requires long computation times or simulation). **Indicator-based Evolutionary Algorithm (IBEA)** [29] is proposed in 2004. Its selection process explicitely lies on the computation of performance indicators. The optimization goal is defined as one or several metrics used to measure both quality and diversity of solutions (such as those presented in section 4). Initially, such metrics were defined to assess the performance of MOEAs at the end of their execution. In IBEA, two metrics are used to define individual fitness, but the authors indicate that any other indicator may be used. Given that these metrics assess both quality and diversity of solutions, no additional diversity preservation mechanism is needed. In keeping with IBEA, many EAs [28] and local search approaches [6] were developed, principally using hypervolume indicator to define their selection process. Indeed, this indicator allows to both estimate quality and diversity of individuals. Besides, it has the advantage that it is the only quality indicator known to be fully sensitive to Pareto dominance: if a set of solutions entirely dominates an other set of solutions, then the dominating set will have the best value of hypervolume indicator [3]. Unfortunately, computing it is NP-hard, and it is not easy to extend its computation to more than two objectives. Therefore, many authors search for efficient methods to compute or assess this indicator [3, 6, 7].

2.2 Adaptive MOEAs

An other way is explored in literature to increase both quality and diversity of solutions returned by EAs. Indeed, such algorithms are often criticized because they involve several parameters, the values of which are difficult to choose. Inadequate values might significantly decrease the algorithm performance. Therefore, many recent research works focus either on statistical tuning approaches [5, 16] or on adaptive algorithms. Such algorithms dynamically adapt their parameter values. They can be classified according to the type of adaptation they perform - i.e. how are the parameters modified ? - or to the adaptation level - i.e. where are the modifications performed (on each individual or on the whole population)? These two classes may be combined to get a hierarchical classification. Eiben et al. [14] first distinguish approaches which use parameter tuning from those which focus on parameter monitoring. Then, they separate this last category in three sub-classes: deterministic, adaptive and self-adaptive approaches. Deterministic monitoring modifies parameters independently from intermediate results provided by the algorithm. Typically, some indicators, such as the generation number, are used to make the values of parameters vary according to a given range of values or to a deterministic function. For instance, the population size periodically varies in Saw-Tooth GA, the algorithm proposed by Koumousis and Katsaras [17]. Tan et al. [20] also presented such a method for binary genetic algorithms. It associates a crossover rate and a mutation rate with each bit. Variations of most significant bits are favoured at the beginning of execution to allow a large exploration of the search space. This probability then decreases in the course of the execution. But according to De Jong [8], deterministic methods can hardly be applied, because predicting the behavior an algorithm will have at a given generation is a difficult task. Adaptive methods explore an other way. They modify the evolution process of the optimization algorithm by monitoring some of its characteristics. Deb et al. [13]

proposed a crossover operator which adapts the algorithm evolution according to both the fitness of parents and the fitness of the resulting offsprings. It is an adaptive version of SBX (Simulated Binary Crossover) [10]. Zeng et al. [25] extended this contribution in such a way that the probability distribution used by SBX is dynamically adjusted according to diversity measures. Hippolyte et al. propose a multi-agent based EA in which the population size results from interactions between the solutions (each of them being represented as an agent) [?]. Finally, self-adaptive algorithms insert the information which defines the adaptation strategy, the parameters to be controlled for instance, in the chromosomes themselves. These additional genes participate to the reproduction operations (in particular crossover). Solutions with efficient adaptation strategies are more likely to survive and to transmit this information to offsprings. Hinterding et al. [15] introduced the gaussian mutation, which is based on this principle: the parameter which permits to control the gaussian function is added to the chromosome.

The new MOEA proposed in this paper, aGAME, belongs to the second category: some of its parameters, such as mutation rate, vary in the course of its execution. These variations depend on performance indicators, which are periodically computed to assess the quality and the diversity of solutions. This is an adaptive version of GAME [1], a MOEA which includes several pareto fronts to rank individuals, a fitness function depending on the pareto set, and a specific selection mechanism. The following section gives a rundown on the main GAME's principles, required to clearly describe the improvements brought in the adaptive version.

3. GAME: STATIC MOEA WITH MULTIPLE PARETO FRONTS

GAME is an elitist MOEA which includes several pareto fronts to rank individuals, like NSGA-II. But it defines several fitness functions, like FastPGA. Indeed the fitness of individuals depends on the pareto set they belong to. Besides, a specific 2-step selection mechanism is also proposed. All these particularities of GAME are detailed in this section.

3.1 Ranking and fitness assignment

GAME (Genetic Algorithm with Multiple parEto sets) splits the population into several Pareto fronts. This permits to progressively rank individuals from non-dominated ones (first Pareto front PF_1) to dominated ones (last Pareto front). An archive PF_0 is also used to store non-dominated solutions found since the beginning of the execution in order to preserve elitism. When the optimization problem to be solved includes constraints, this is taken into account while building the successive Pareto fronts, using a penalty strategy. Solutions which do not satisfy constraints (i.e. unfeasible solutions) are not allowed to enter PF_1. Nevertheless, when this set is empty, such solutions may be accepted by default. Such a case may occur during the first generations of the algorithm. But this is fastly corrected, as soon as feasible solutions are found. They are more likely to survive, they transmit their characteristics to their descendants, and then unfeasible solutions naturally leave PF_1.

Individual fitness depends on the Pareto front the evaluated solution belongs to. GAME distinguishes three fitness definitions. Given that the algorithm must return solutions as close as possible to the optimal front PF^* while preserving their diversity, the fitness of the solutions contained in PF_0 and PF_1 is a diversity indicator: the crowding distance [12]. This indicator gives information about the distribution of solutions within the front. GAME uses a normalized *crowding distance* in order to avoid bias which might be involved by the sizes of the different objective ranges. In the second front PF_2, GAME favours the solutions which are closest to PF_1. The used indicator is *generational distance*. It measures the proximity between the solutions belonging to the two fronts. Using a fitness based on this indicator increases the selection probability of the solutions near PF_1, for reproduction and survival. Finally, for solutions in all the remaining fronts, the fitness is defined as an indicator called **gain**. The gain of solution $\overrightarrow{x_i}$ in relation to $\overrightarrow{x_j}$ for objective function f_k is called $gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k)$. It represents the improvement brought by $\overrightarrow{x_i}$ in comparison to $\overrightarrow{x_j}$ for the considered function. This gain, defined by equation 1, takes values between -1 and 1. A negative value indicates that the second solution is better than the first one for this objective. The gain equals zero when the two solutions are equivalent.

$$gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) = \frac{\lambda(f_k(\overrightarrow{x_i}) - f_k(\overrightarrow{x_j}))}{Max(f_k(\overrightarrow{x_i}), f_k(\overrightarrow{x_j}))} \quad (1)$$

where λ is a coefficient equal to 1 for maximization problems and -1 for minimization. The fitness of each individual is the sum of its gains (for all the objective functions) in relation to the other solutions belonging to the same front (see equation 2).

$$fitness(\overrightarrow{x_i}) = \sum_{j=1}^{|PF|} \sum_{k=1}^{m} gain(\overrightarrow{x_i}, \overrightarrow{x_j}, k) \quad (2)$$

where

- $|PF|$ is the size of the front to which $\overrightarrow{x_i}$ belongs;
- m is the number of objective functions of the tackled problem;

Each of these fitness definitions only makes sense within the same front. Selection works in two steps. The first one selects a front, while adapting to the number of fronts and to their sizes. Indeed, a great selection probability assigned to a front which size is low should give too much strength to the individuals it contains. This would reduce diversity rate in the next generation. The selection probability $Proba(PF_i)$ GAME assigns to each front PF_i takes into account both its rank i and its size, in order to overcome this problem. Finally, in the second selection step, a fitness-based tournament is used to select an individual among those belonging to the selected front.

An experimental study was conducted to show the usefulness and the efficiency of the proposed ranking and selection process. GAME compared favorably against the three best ranked algorithms of the CEC 2009 competition [27]. Nevertheless, for this static version, tuning of parameters was performed in a preliminary parametric study based on statistical tools. Although the tuned values provided promising results, they are certainly not adequate for all kinds of problems. Constant parameter values hardly allow algorithms to adapt to search spaces which might have very different characteristics. This motivated the design of an adaptive version of GAME, called aGAME, in which a new dynamic adaptation operator (DAO) is added to increase robustness. The following section describes this contribution.

4. DYNAMIC ADAPTATION OPERATOR

Dynamic adaptation must allow MOEAs to adapt to various search spaces, particularly to detect local optima and to climb out of them. It must also increase the algorithm performance by making it find final solutions that are both very various and as close as possible to global optima.

According to [9], an efficient MOEA must satisfy two conflicting goals: ensuring a wide exploration of the search space while guiding the algorithm towards one (or several) global optimum(a). In literature, many works [20, 21, 22, 4] show that a wide exploration should be favoured at the beginning of the algorithm execution. This strategy is known as ***explore first, exploit later***. Exploration permits to identify good solutions. Then, in the second phase, exploitation, the neighborhood of these solutions is more intensively examined. The performance of the various proposed approaches are generally compared using some indicators which measure both quality and diversity of solutions. This work proposes a new way to maintain the equilibrium between exploration and exploitation phases. The proposed dynamic adaptation operator periodically alternates various modes of exploration all along aGAME execution. Mode switching occurs according to the values of quality and diversity indicators. The following subsection presents the chosen metrics and motivates this choice.

4.1 Quality indicators

In literature, many indicators were proposed to assess the quality of the solutions returned by MOEAs. Like some previous adaptive approaches, aGAME uses such measures to control the algorithm behavior. But, the proposed dynamic adaptation operator uses two separate indicators, rather than using a single metric to measure both diversity and quality of solutions, like hypervolume indicator. First, this choice was made to ensure the scalability of aGAME. Indeed, since the computation of hypervolume indicator is NP-hard, using it would complicate the extension of aGAME to solve optimization problems involving more than two objective functions. Besides, Deb and Jain [11] explain that it is best to use an indicator to measure the diversity of solutions and an other one to assess the convergence of solutions to the Pareto-optimal front. Therefore, the proposed operator, DAO, uses the **Inverted generational distance** (IGD) to measure the proximity between solutions and the optimal Pareto front, when this one is known. Otherwise, the non-dominated archive, which constitutes the best estimation of this reference front, is used instead. IGD lies on the computation of an euclidean distance to assess the mean of the minimal distance between all the points of PF^* and those of PF (see equation 3).

$$IGD = \frac{\left(\sum_{i=1}^{|PF^*|} d_m^*(\overrightarrow{x_i^*})\right)^{1/m}}{|PF^*|} \qquad (3)$$

where

$$d_m^*(\overrightarrow{x_i^*}) = \min_{j=1}^{|PF|} \sqrt{\sum_{k=1}^{m} (f_k(\overrightarrow{x_i^*}) - f_k(\overrightarrow{x_j}))^2} \qquad (4)$$

$\overrightarrow{x_i^*}$ is a solution belonging to PF^* and $\overrightarrow{x_j}$ is a PF solution.

DAO also uses a diversity indicator proposed by Deb and Jain, and named ***diversity running performance metric*** (DM) [11]. It evaluates the distribution of solutions in a given front, in relation to one objective function. DM is computed, for each generation, by projecting the non-dominated solutions (belonging to the first Pareto front PF_1) onto an hyperplane. The latter is then divided in n hypercubes [1]. The diversity measure depends on the presence, or the absence, of solutions in each hypercube. The DM value is maximal when each hypercube contains at least one solution.

Zeng et al. [24, 26] propose to define n as the ratio between the size of the population and the number of objective functions.

DAO periodically calculates the values of IGD and DM. IGD is used to detect if aGAME does not manage to reduce the proximity between the population and the reference front. DM is used to check if this is due to a lack of diversity, which could indicate that aGAME is trapped in a local optimum. According to these elements, aGAME chooses the best mode of exploration to be used during the next period. This is detailed in the following subsection.

4.2 Adaptation mechanism

Keeping in the *explore first, exploit later* strategy, DAO first uses an **initial exploration** phase, widely exploring the search space, at the beginning of aGAME execution. This phase duration is t_{init}, expressed in number of generations. During this phase, the mutation rate (p_m) depends on the diversity level within the population (see equation 5).

$$p_m = 1 - DM(\textit{population}) \qquad (5)$$

$DM(\textit{population})$ takes values between 0 and 1. The closest to 1 they are, the best the diversity level of the population is. When $DM(\textit{population})$ is near 0, representing a low diversity within the population, p_m increases in order to favour mutations.

Beyond t_{init}, exploitation must be strengthened while preserving aGAME from being trapped in possible local optima. DAO regularly supervises (with a periodicity $t_{monitoring}$) the algorithm behavior. It guides its evolution, when necessary. Figure 1 presents the general working of DAO.

When the IGD of the archive did not vary between two successive monitorings, this indicates that a stagnation phase of IGD has begun [2]. The correction to be applied depends on the diversity level within the best solutions evaluated at

[1]Hypercubes which dimension is $m-1$, m being the number of objective functions of the tackled problem.

[2]Computing the IGD of the archive is only possible when the optimal solutions are known. Otherwise, the archive is used as a reference front, because it represents the best known assessment of the optimal Pareto front. Therefore, the instruction **If** $IGD(A(t)) = IGD(A(t_d))$ must be replaced by **If** $IGD(PF_1(t)) \leq IGD(PF_1(t_d))$ in the algorithm given in figure 1.

```
1: $t$: generation number
2: $t_{init}$: duration of the initialization phase
3: $t_{monitoring}$: monitoring period of the algorithm
4: $t_d$: number of the generation during which the latest
   monitoring occurred
5: $P(t)$: population at generation $t$
6: $PF_1(t)$: first Pareto front of the population at genera-
   tion $t$
7: $A(t)$: non-dominated archive built until generation $t$
8: $mode$: behavior the algorithm must choose ($normal$,
   $exploration$, $initial\ exploration$, or $exploitation$)
9: if $t < t_{init}$ then
10:    $mode \leftarrow initial\ exploration$
11: else
12:    if $t \% t_{monitoring} = 0$ then
13:       if $IGD(A(t)) = IGD(A(t_d))$ then
14:          if $DM(PF_1(t)) > DM(PF_1(t-1))$ then
15:             $mode \leftarrow exploitation$
16:          else
17:             $mode \leftarrow exploration$
18:          end if
19:       else
20:          $mode \leftarrow normal$
21:       end if
22:    end if
23:    $t_d \leftarrow t$
24: end if
25: return $mode$
```

Figure 1: Dynamic adaptation algorithm

time t. If these solutions are more various than those belonging to the previous generation, there is certainly no diversity loss in the current population. The behavior to adopt, **exploitation mode**, consists in intensifying the search of new solutions in the neighborhood of the best known solutions. The archive of non-dominated solutions then participates to reproduction operations. During the selection of parents, it has a greater selection probability than any other set of solutions. Consequently the solutions it contains have priority when aGAME selects some parents.

If IGD stagnates and the diversity of the best found solutions did not change or decreased, the algorithm might be trapped in an optimum, which may be either local or global. In case of global optimum, finding no improvement in the quality of the solutions anymore is normal. All the behavior corrections brought by DAO will be unsuccessful, but this will not damage the archive quality. Otherwise, if the algorithm is in a local optimum, increasing its mutation rate could allow it to escape from this trap. The strategy to be used is then the **exploration mode**. Equation 6 precises the formula used to control the mutation rate variation.

$$p_m = DM(PF_1) \tag{6}$$

If DAO does not detect any IGD stagnation at the time of monitoring, it selects the **normal mode** which works as GAME. In this case, the mutation rate equals a value usually used in literature (equation 7).

$$p_m = \frac{1}{number\ of\ variables} \tag{7}$$

Thus, periodical monitorings based on the proposed DAO permit to guide aGAME all along its execution.

5. VALIDATION AND PERFORMANCE ASSESSMENT

The specific characteristics of aGAME are validated in this section. The impact of multiple Pareto fronts was illustrated in previous work [1]. This paper focuses on the influence of the dynamic adaptation operator and the comparison of aGAME with other algorithms in the literature.

5.1 Experimental procedure

The experimentations carried out in this paper are based on all constrained bi-objective problems proposed for the CEC 2009 competition [27]. It consists in seven minimization problems with 10 real-valued decision variables.

In accordance with the experimental procedure indicated for the CEC 2009 competition, each evolutionary algorithm executes thirty independent runs for each addressed problem. The maximum number of evaluations is 30,000. In order to meet this requirement, GAME and aGAME use a population size equals to 100 and a number of generations equals to 300. Moreover, they use a 2-point crossover and a bitflip mutation operators. The crossover rate is 0.8 and the mutation rate is $\frac{1}{number\ of\ variables}$ (it may vary during aGAME's execution, due to DAO).

The parameters of each algorithm should be the same for all the problems with the same number of objective functions.

The performance indicator to be used as a comparison criterion is the IGD (see Equation 3). The IGD is to be minimized.

5.2 Influence of the dynamic adaptation operator

Since evolutionary algorithms are stochastic processes, they are usually carried out several times before drawing up conclusions based on statistically reliable results. The first constrained problem of the CEC 2009 competition has been solved in thirty independent runs by GAME (non-adaptive algorithm) and aGAME, in order to assess the impact of the proposed dynamic adaptation operator. The evolution of the IGD over generations for the two algorithms is shown in Figure 2. In this figure, each point represents the average of thirty IGD (30 runs) for the corresponding generation number.

At the end of 300 generations, aGAME provides a lower value of IGD than GAME. The curve of the IGD provided by aGAME decreases slower than the one given by GAME. This is due to the initial phase of exploration which favors a quick scan of the search space. The two curves in Figure 2 are obtained by computing the average of the IGD (of each generation) for thirty executions. A detailed examination of the progress of GAME for a randomly selected execution shows that the curve of the IGD is a staircase curve, with relatively long periods of stagnation. This is illustrated in Figures 3 to 8, which are representative of the evolution of the IGD along 300 generations for 6 independent trials (with 5 Pareto fronts).

The experience that allowed to plot Figure 3 was taken with aGAME (using the same algorithmic parameters and the same set of random numbers) in order to illustrate the

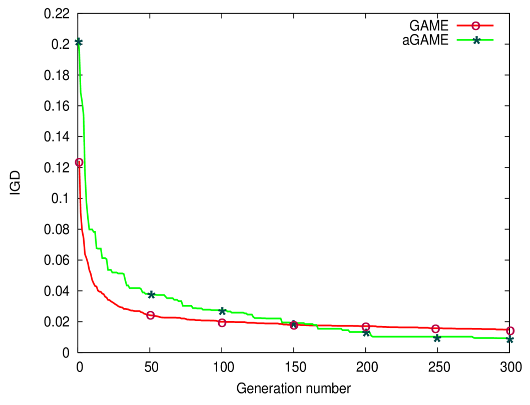

Figure 2: Average IGD for 30 independent runs of GAME and aGAME on the CEC 2009 first constrained problem

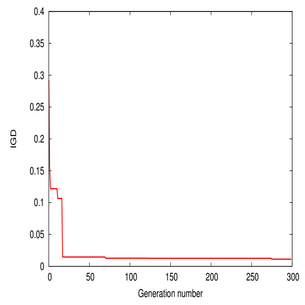

Figure 3: Trial 1

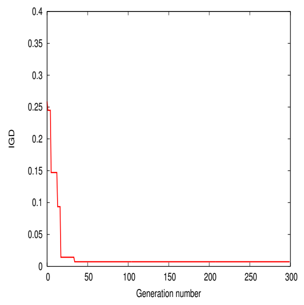

Figure 4: Trial 2

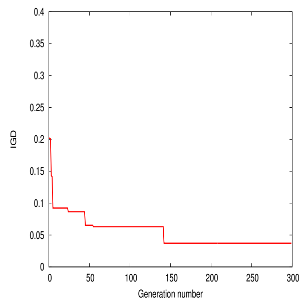

Figure 5: Trial 3

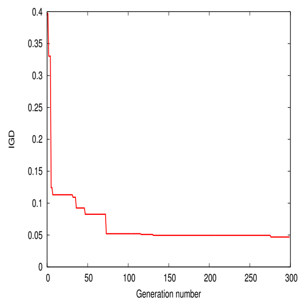

Figure 6: Trial 4

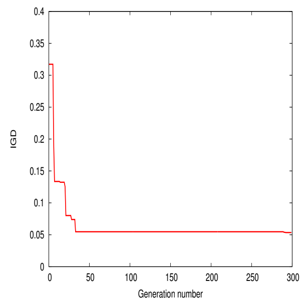

Figure 7: Trial 5

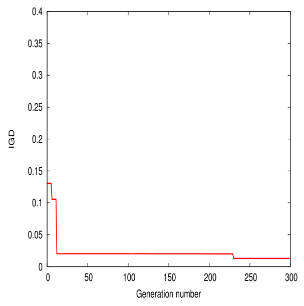

Figure 8: Trial 6

influence of DAO on the progress of aGAME. The following values were arbitrarily set to the parameters required by DAO:

- t_{init} = 15% of the total number of generations
- $t_{monitoring}$ = 5% of the total number of generations

The IGD of the archive of non-dominated solutions of aGAME (Figure 9) does not decline as rapidly as in the case of GAME (Figure 3). Indeed, aGAME begins with an initial exploration phase during which the probability of mutations is generally higher than in the GAME version. The goal is to enable a broad scan of the search space during the early generations and gather as much information as possible. The best solutions are added to the archive and their characteristics can be used during the exploitation phase.

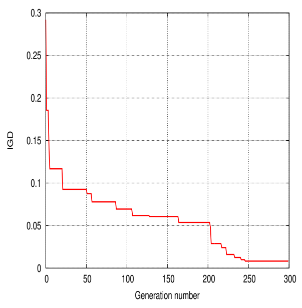

Figure 9: Trial 1 with DAO

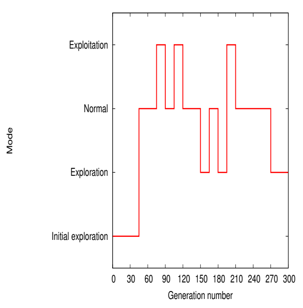

Figure 10: Dynamic mode switching

For this example, the initial exploration phase is 45 generations long. After the expiration of t_{init}, aGAME alternates between three modes: normal, exploitation and exploration (see Figure 10). This dynamic change strategy allows aGAME to steady improve its convergence indicator. This enables aGAME to reach lower IGD values at the end of the optimization process.

5.3 Comparison with other MOEAs

aGAME is compared to some algorithms of the literature. This comparative study is based on constrained bi-objective problems proposed for the CEC 2009 competition. Thus, it is compared to the three best ranked algorithms in this category (*Constrained problems*)during the CEC 2009 competition: DMOEA-DD [19], LiuLi [18] and MTS [23].

DMOEAD-DD (Dynamical Multiobjective Evolutionary Algorithm - Domain Decomposition) improves DMOEA [31] that used an aggregated *fitness* function including the notion of Pareto dominance, entropy and density (based on *crowding distance*). In the improved variant, authors split the search space into several subsets. DMOEA computes Pareto fronts for each of them. Genetic operators permit information exchange between these subsets.

LiuLi (concatenation of the authors'name: Hai-Lin Liu and Xueqiang Li) splits the search space into sub-areas in order to reduce algorithm complexity. Genetic operations, particularly reproduction, are performed in a single sub-area. Information exchange between areas is based on children, because they may be assigned to other areas.

MTS (Multiple Trajectory Search) is an algorithm based on three local search methods. For each solution, MTS determines the method which corresponds to its neighborhood.

This algorithm begins with a large search. The size of neighborhood is progressively reduced until it reaches a given minimal size. Then its size is set to its initial value and the regression re-starts.

To illustrate the gain brought by the dynamic adaptation operator (DAO), aGAME performance is also compared to those of its non-adaptive version (GAME). The comparison is made in accordance to the experimental conditions defined in the competition. For each algorithm, Table 1 presents the average of the IGD values computed in the sets of final solutions, in 30 independent executions. This study shows that, globally, GAME reaches solutions which are quite close to the reference front. This results in low IGD values in table 1. Moreover the adaptive version, aGAME, performs better than GAME.

In Table 1, the values in brackets represent the gain of the associated algorithm with respect to aGAME. A negative value indicates that aGAME outperforms that algorithm.

This study shows that, globally, GAME reaches solutions which are quite close to the reference front. This results in low IGD values in table 1. Moreover the adaptive version, aGAME, performs better that GAME

6. CONCLUSION

To sum up, GAME is an elitist multi-objective genetic algorithm, based on the building of multiple Pareto fronts. This ranking strategy and the associated 2-step selection provided gains both in terms of proximity with optimal solutions and in terms of diversity in the set of final solutions returned by GAME.

Moreover, aGAME improves the results of GAME adding an adaptive strategy (DAO). Each loop is more efficient by selecting the right mode (exploration or exploitation) at the right time.

Finally, using the experimental conditions of the CEC 2009 competition showed that aGAME would have been quite well ranked in this competition, which constitutes a promising result. In addition, aGAME uses a parallel evaluation procedure based on a master-slave model. This model allowed to reduce the experimentation duration in previous work in the field of mobile networks [2]. In this context, a previous version of the algorithm allowed to solve constrained problems with four objectives, the evaluation of which was done using a network simulator. Nevertheless, designing an asynchronous version of aGAME would be very interesting. Such a version would no longer rely on an architecture where the master must wait until all the slaves have finished their task before building the next generation.

7. REFERENCES

[1] W. Abdou, C. Bloch, D. Charlet, and F. Spies. Multi-pareto-ranking evolutionary algorithm. In *EvoCOP 2012, 12th European Conference on Evolutionary Computation in Combinatorial Optimisation*, LNCS, Malaga, Spain, 2012. Springer. To appear.

[2] W. Abdou, A. Henriet, C. Bloch, D. Dhoutaut, D. Charlet, and F. Spies. Using an evolutionary algorithm to optimize the broadcasting methods in mobile ad hoc networks. *Journal of Network and Computer Applications*, 34(6):1794 – 1804, 2011.

[3] J. Bader and E. Zitzler. Hype: an algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[4] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 8(2):137–155, 2004.

[5] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Germany, November 2010.

[6] M. Basseur, R.-Q. Zeng, and J.-K. Hao. Hypervolume-based multi-objective local search. *Neural Computing & Applications*, pages 1–13, 2011. 10.1007/s00521-011-0588-4.

[7] K. Bringmann and T. Friedrich. Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. *Theoretical Computer Science*, (0):–, 2010.

[8] K. De Jong. Parameter Setting in EAs: a 30 Year Perspective. pages 1–18. 2007.

[9] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[10] K. Deb and R. B. Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9(2):115–148, 1995.

[11] K. Deb and S. Jain. Running Performance Metrics for Evolutionary Multi-Objective Optimization. In L. Wang, K. C. Tan, T. Furuhashi, J.-H. Kim, and X. Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 1, pages 13–20, Orchid Country Club, Singapore, November 2002. Nanyang Technical University.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[13] K. Deb, K. Sindhya, and T. Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1187–1194, New York, NY, USA, 2007. ACM.

[14] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transations on Evolutionary Computation*, 3(2):124–141, July 1999.

[15] R. Hinterding, Z. Michalewicz, and T. C. Peachey. Self-adaptive genetic algorithm for numeric functions. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, PPSN IV, pages 420–429, London, UK, 1996. Springer-Verlag.

[16] J.-L. Hippolyte, C. Bloch, P. Chatonnay, C. Espanet, D. Chamagne, and G. Wimmer. Tuning an evolutionary algorithm with taguchi methods. In R. Chbeir, Y. Badr, A. Abraham, D. Laurent, and F. Ferri, editors, *CSTST'08, 5th Int. Conf. on Soft Computing as Transdisciplinary Science and Technology*, pages 265–272. ACM Press, 2008.

Problem	**DMOEADD**	**LiuLi**	**MTS**	**GAME**	**aGAME**
Constrained Problem 1	0.01131	0.00085	0.01918	0.01489	0.00828
Constrained Problem 2	0.0021	0.004203	0.02677	0.00042	0.00039
Constrained Problem 3	0.056305	0.182905	0.10446	0.03462	0.03385
Constrained Problem 4	0.00699	0.014232	0.01109	0.00742	0.00070
Constrained Problem 5	0.01577	0.10973	0.02077	0.01227	0.01129
Constrained Problem 6	0.01502	0.013948	0.01616	0.00181	0.00138
Constrained Problem 7	0.01905	0.10446	0.02469	0.00545	0.00373
Average Ranking	3rd	4th	5th	2nd	1st

Table 1: Mean values of IGD returned by aGAME, GAME and the 3 best ranked algorithms in CEC 2009 competition

[17] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Trans. Evolutionary Computation*, 10(1):19–28, 2006.

[18] H.-L. Liu and X. Li. The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 1928–1934, Piscataway, NJ, USA, 2009. IEEE Press.

[19] M. Liu, X. Zou, Y. Chen, and Z. Wu. Performance assessment of dmoea-dd with cec 2009 moea competition test instances. In *IEEE Congress on Evolutionary Computation*, pages 2913–2918, 2009.

[20] K. C. Tan, S. C. Chiam, A. A. Mamun, and C. K. Goh. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2):701–713, 2009.

[21] K. C. Tan, C. K. Goh, Y. J. Yang, and T. H. Lee. Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2):463–495, 2006.

[22] K. C. Tan, T. H. Lee, and E. F. Khor. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Trans. Evolutionary Computation*, 5(6):565–588, 2001.

[23] L.-Y. Tseng and C. Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 1951–1958, Piscataway, NJ, USA, 2009. IEEE Press.

[24] F. Zeng, J. Decraene, M. Y. H. Low, S. Zhou, and W. Cai. Diversity-driven self-adaptation inăevolutionary algorithms. 70:95–106, 2011.

[25] F. Zeng, M. Y. H. Low, J. Decraene, S. Zhou, and W. Cai. Self-Adptive Mechanism for Multi-objective Evolutionary Algorithms. In *Proceedings of the InternationalMultiConforence of Engineers and Computer Scientists 2010, IMECS 2010.*

[26] F. Zeng, M. Y. H. Low, J. Decraene, S. Zhou, and W. Cai. Self-adaptive mechanism for multi-objective evolutionary algorithms. In S. I. Ao, O. Castillo, C. Douglas, D. D. Feng, and J.-A. Lee, editors, *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2010)*, pages 7–12. Newswood Limited, March 2010.

[27] Q. Zhang, A. Zhou, S. ZhaoâĂă, P. N. SuganthanâĂă, W. Liu, and S. Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. Technical report, The School of Computer Science and Electronic Engineering, 2009.

[28] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49, 2011.

[29] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842. Springer, 2004.

[30] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[31] X. Zou, Y. Chen, M. Liu, and L. Kang. A new evolutionary algorithm for solving many-objective optimization problems. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 38(5):1402–1412, Oct. 2008.