Multi-Objective Differential Evolution with Adaptive Control of Parameters and Operators

Ke Li[1], Álvaro Fialho[2], and Sam Kwong[1]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong
jerryli3@student.cityu.edu.hk, cssamk@cityu.edu.hk
[2] LIX, École Polytechnique, Palaiseau, France
fialho@lix.polytechnique.fr

Abstract. Differential Evolution (DE) is a simple yet powerful evolutionary algorithm, whose performance highly depends on the setting of some parameters. In this paper, we propose an adaptive DE algorithm for multi-objective optimization problems. Firstly, a novel tree neighborhood density estimator is proposed to enforce a higher spread between the non-dominated solutions, while the Pareto dominance strength is used to promote a higher convergence to the Pareto front. These two metrics are then used by an original replacement mechanism based on a three-step comparison procedure; and also to port two existing adaptive mechanisms to the multi-objective domain, one being used for the autonomous selection of the operators, and the other for the adaptive control of DE parameters CR and F. Experimental results confirm the superior performance of the proposed algorithm, referred to as AdapMODE, when compared to two state-of-the-art baseline approaches, and to its static and partially-adaptive variants.

Keywords: Multi-Objective Optimization, Differential Evolution, Tree Neighborhood Density, Parameter Control, Adaptive Operator Selection.

1 Introduction

Differential Evolution, proposed by Storn and Price [15], is a popular and efficient population-based, direct heuristic for solving global optimization problems in continuous search spaces. The main benefits brought by DE are its simple structure, ease of use, fast convergence speed and robustness, which enables it to be widely applied to many real-world applications. For the generation of new solutions (trial vectors), each individual (target vector) is combined with others by means of different forms of weighted sums (mutation strategies). Originally, in case the newly generated solution has a better fitness value than its corresponding parent, it replaces its parent in the population for the next generation. The aim of these iterations is basically to find a proper direction for the search process towards the optimum, by following the quality distribution of the solutions in the current population.

C.A. Coello Coello (Ed.): LION 5, LNCS 6683, pp. 473–487, 2011.

One of the possible application domains of DE are the Multi-objective Optimization Problems (MOPs), which exist everywhere in real-world applications, such as engineering, financial, and scientific computing. The main difficulty in these cases lies in providing a way to compare the different solutions, as the involved multiple criteria might compete with one another, besides possibly not being directly comparable. Multi-Objective Evolutionary Algorithms (MOEAs) tackle this issue by searching for the set of optimal trade-off solutions, the so-called Pareto optimal set: the aim is not only to approach the Pareto optimal front as closely as possible, but also to find solutions that are distributed over the Pareto optimal front as uniformly as possible, in order to better satisfy all the different objectives considered. Needless to say, to be applied to MOPs, the DE original scheme needs to be adapted according to the mentioned aims.

Many different types of DE variants proposed to tackle MOPs can be found in the literature, such as GDE3 [12], and DEMO [17]. We refer the reader to [2] for a recent comprehensive survey of DE, including its application to MOPs. But the performance of DE largely depends on the definition of some parameters. Besides the crossover rate CR, and the mutation scaling factor F, there is the need of choosing which mutation strategies, from the many available ones, should be used for the generation of new solutions, and at which rate each of the chosen strategies should be applied. The setting of these parameters is usually a crucial and very time-consuming task: the optimal values for them do not only depend on the problem at hand, but also on the region of the search space that is being explored by the current population, while solving the problem. Following the intuition of the Exploration versus Exploitation (EvE) balance, exploration tends to be more beneficial in the early stages of the search (consequently more exploratory mutation strategies, high values for F and CR), while more exploitation should be promoted when getting closer to the optimum (respectively, more fine-tuning operators, and a smaller value for F).

A prominent paradigm to automate the setting of these parameters on-line, i.e., while solving the problem, is the so-called Adaptive parameter control. It constantly adapts the values of the parameters based on feedbacks received from the search process. Some algorithms have been recently proposed for the on-line adaptation of CR and F, and for the autonomous control of which of the strategies should be applied at each instant of the search, the latter being commonly referred to as Adaptive Operator Selection (AOS). Some DE algorithms using adaptive methods can be found in the literature, such as SaDE [16], JADE [21], jDE [1] and ISADE [11]. Regarding DE for MOPs, there also exists some pioneering works, such as JADE2 [20] and OW-MOSaDE [10]. However, to the best of our knowledge, the employment of both adaptive parameter control of CR and F, and adaptive operator (mutation strategy) selection, is still relatively scarce in the domain of MOPs.

In this work, we employ an adaptive parameter control of CR and F slightly different from the one employed by the JADE method [21], which adapts their values based on the recent success rate of the search process; and an AOS mechanism inspired from the PM-AdapSS-DE method [9], which uses the Probability

Matching mechanism to select between the available mutation strategies, based on the normalized relative fitness improvements brought by their recent applications. The main contribution of this work lies in the porting of these adaptive methods to the multi-objective domain. More specifically, a novel method is proposed to partially evaluate the fitness of the solutions, referred to as Tree Neighborhood Density (TND) estimator. The aggregation of the TND with the Pareto Dominance Strength (brought from the SPEA2 [23] method) is the information used by the AOS mechanism to keep its operator preferences up-to-date, and by a novel replacement mechanism based on a three-step comparison scheme. Lastly, the output of this replacement mechanism defines the success rates used for the adaptive parameter control of CR and F. The resulting algorithm, referred to as Adaptive Multi-Objective DE (Adap-MODE), is assessed in the light of a set of multi-objective benchmark functions, and shows to achieve significantly better results than other state-of-the-art approaches (NSGA-II [4] and GDE3 [12]) and than its static and partially-adaptive variants in most of the cases.

The remainder of this paper is organized as follows. Firstly, the background and some related work are briefly reviewed in Section 2. Then, our proposed algorithm is described in detail in Section 3. After that, some experimental results are analyzed in Section 4. Finally, Section 5 concludes this paper and gives possible directions for further work.

2 Related Work

The performance of an Evolutionary Algorithm (EA) strongly depends on the setting of some of its parameters. Section 2.1 will briefly overview the different ways of doing parameter setting in EAs, focusing on the kind of approach used in this work, referred to as Adaptive Parameter Control. Then, Section 2.2 will survey more specifically the Adaptive Operator Selection (AOS) paradigm.

2.1 Parameter Setting in Evolutionary Algorithms

There are different ways of doing parameter setting in EAs, as acknowledged by the well-known taxonomy proposed by Eiben et al. in [6]. In the higher level, there is the separation between Parameter Tuning and Parameter Control methods. Parameter Tuning methods set the parameters off-line, based on statistics over several runs; besides being computationally expensive, it provides a single parameter setting, that remains static during all the run. Parameter Control methods continuously adapt the parameters on-line, i.e., while solving the problem; these methods are further sub-divided into three branches, as follows.

The Deterministic methods adapt the parameter values according to predefined (deterministic) rules; but the definition of these rules already defines a complex optimization problem *per se*, besides hardly adapting to different problems. The Self-Adaptive methods adapt the parameter values *for free*, by encoding them within the candidate solution and letting the evolution take care of their control; in this case, however, the search space of the parameter values

is aggregated to that of the problem, what might significantly increase the overall complexity of the search process. Lastly, the Adaptive methods control the parameter values based on feedback received from the previous search steps of the current optimization process.

In this work, we use an adaptive method very similar to the one proposed in the JADE algorithm [21], which controls the values of DE crossover rate CR and mutation scaling factor F based on the recent success rate (more details in Section 3.4). Another example of adaptive method proposed for the same aim is the SaDE [16] algorithm. Furthermore, another kind of adaptive method is also used in our algorithm, the AOS, surveyed in the following.

2.2 Adaptive Operator Selection

A recent paradigm, referred to as Adaptive Operator Selection (AOS), proposes the autonomous control of which operator (or mutation strategy in the case of DE) should be applied at each instant of the search, while solving the problem, based on their recent performance. A general AOS method usually consists of two components: the Credit Assignment scheme defines how each operator should be rewarded based on the impacts of its recent applications on the search progress; and the Operator Selection mechanism decides which of the available operators should be applied next, according to their respective empirical quality estimates, which are built and constantly updated by the rewards received. Each of these components will now be briefly overviewed in turn.

Credit Assignment
The most common way of assessing the impact of an operator application is the fitness improvement achieved by the offspring generated by its application, with respect to a baseline individual. In [9], the fitness improvement with respect to its parent is considered, while [3] use as baseline individual the best individual of the current population.

Based on this impact assessment, different ways of assigning credit to the operators can be found, in addition to the common average of the recent fitness improvements. In [19], a statistical technique rewards the operators based on their capability of generating outlier solutions, arguing that rare but highly beneficial improvements might be more important than frequent small improvements. Along the same line, in [8] each operator is rewarded based on the extreme (or maximal) fitness improvement recently achieved by it. In the quest for a more robust rewarding, in [7] a rank-based scheme is proposed. In multi-modal problems, however, the diversity is also important; in [14], both diversity variation and fitness improvement are combined to evaluate the operator application.

Operator Selection
The Operator Selection mechanism usually keeps an empirical quality estimate for each operator, built by the received rewards, which is used to guide its selection. The most popular method for Operator Selection is referred to as Probability Matching (PM) [18]: basically, the probability of selecting each operator

is proportional to its empirical quality estimate with respect to the others; this is the method used in this work, more details in Section 3.3.

Other more complex Operator Selection methods worth to be mentioned are: the Adaptive Pursuit (AP) [18], originally proposed for learning automata, employs a winner-takes-all strategy to enforce a higher exploitation of the best operator; and the Dynamic Multi-Armed Bandit (DMAB) [8], which tackles the Operator Selection problem as yet another level of the Exploration vs. Exploitation dilemma, efficiently exploiting the current best operator, while minimally exploring the others, inspired from the multi-armed bandit paradigm.

3 Adaptive Multi-Objective DE

The general framework of the proposed adaptive Differential Evolution (DE) algorithm for multi-objective problems is illustrated in Fig. 1. As can be seen, it is divided into three modules. In the middle, there is the main cycle of the DE algorithm, represented here by only three steps for the sake of brevity: once after every generation, the fitness (see Section 3.1) of each offspring is evaluated by the sum of its Pareto Dominance (PD) strength and its Tree Neighborhood Density (TND). While the PD enforces convergence towards the Pareto front, the TND promotes diversification between the non-dominated solutions. These two measures are separately used by the Replacement mechanism, that decides which of the individuals should be maintained for the next generation by means of an original three-step comparison procedure (Section 3.2).

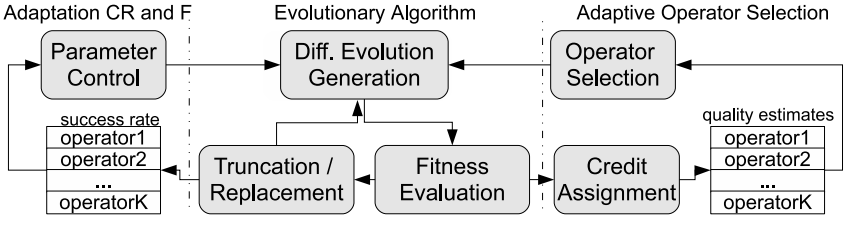

Fig. 1. The framework of the proposed adaptive Differential Evolution algorithm

Two adaptive mechanisms are employed in parallel. On the right side, there is the AOS module, inspired from the PM-AdapSS-DE algorithm [9]. And on the left side, there is the Adaptive Parameter Control module slightly modified from the JADE algorithm [21]. Both adaptive mechanisms are described, respectively, in Sections 3.3 and 3.4. Although these are adaptive mechanisms brought from the literature, it is worth noting that in this work they are originally ported to the multi-objective domain, by receiving inputs based on the special aggregation between the PD and the novel TND measures.

3.1 Fitness Evaluation

In multi-objective optimization, the aims of the search can be said to be two-fold. On the one hand, the solutions found should approach as much as possible to the Pareto front. On the other hand, the non-dominated solutions should be distributed over the Pareto front as uniformly as possible, in order to have satisfiable solutions for all the different objectives. In this work, we use the Pareto Dominance (PD) strength proposed in the SPEA2 algorithm [23] to enforce the first issue (convergence). For the second issue, we propose a novel measure to promote spread between the non-dominated solutions, referred to as the Tree Neighborhood Density (TND). The fitness of each individual is assessed by an aggregation of these two criteria, as described in the following.

Pareto Dominance Strength
In order to calculate the Pareto Dominance (PD) strength, we use the mechanism proposed in the SPEA2 algorithm [23]. The only difference is that the external archive to store elite individuals is not implemented here. Briefly, a strength value $S(i)$ is assigned to each individual i in the population P, representing the number of solutions it dominates. If solely based on this criterion, the fitness of each individual i, referred to as $PD(i)$ here, would be calculated as:

$$PD(i) = \sum_{j \in P, j \succ i} S(j) \tag{1}$$

i.e., the sum of the strengths of all the individuals that dominate individual i. Intuitively, the smaller the better, with $PD(i) = 0$ corresponding to a non-dominated solution; whereas a large $PD(i)$ means that the individual i is dominated by many others.

Tree Neighborhood Density
As previously mentioned, the Tree Neighborhood Density (TND) is a novel estimation proposed to enforce a higher level of spread between the non-dominated solutions. For the sake of a clearer discussion, some definitions and terminologies are firstly given as follows.

Definition 1 (Tree crowding density). *Let T be a minimum spanning tree connecting all the individuals of population P. For any individual i in P, let d_i be the degree of i in T, i.e., the number of edges of T connected to i; and let these edges be $\{l_{i,1}, l_{i,2}, \ldots, l_{i,d_i}\}$. The tree crowding density of i is estimated as:*

$$T_{crowd}(i) = \sum_{j=1}^{d_i} l_{i,j}/d_i \tag{2}$$

Definition 2 (Tree neighborhood). *Let $r_i = \max\{l_{i,1}, l_{i,2}, \ldots, l_{i,d_i}\}$. A circle centered in individual i, with radius r_i, is defined as the tree neighborhood of i.*

Definition 3 (Membership of individual on the tree neighborhood). *Let the Euclidean distance between individuals i and j be denoted as $dist_{i,j}$. The individual j is considered as a member of the tree neighborhood of i if and only if $dist_{i,j} \leq r_i$ (denoted as $i \rhd_T j$).*

Based on these definitions, the calculation procedure for the Tree Neighborhood Density (TND) is implemented as follows:

1. The Euclidean distance between each individual of the population P with the other $NP - 1$ individuals is calculated;
2. A minimal spanning tree T connecting all individuals is generated;
3. The tree crowding density for each individual i in T is assessed, and the corresponding tree neighborhood is generated;
4. For each individual i, the degrees of the individuals pertaining to its tree neighborhood are summed:

$$sumdegrees(i) = \sum_{j \in U} d_j, \text{ where } U = \{j | j \in P, i \rhd_T j\} \tag{3}$$

5. Then, the Tree Neighborhood Density of individual i is calculated as:

$$TND(i) = \frac{\sum_{j \in U}(1/Tcrowd_j)}{sumdegrees(i)} \tag{4}$$

6. Finally, the TND values of all individuals are normalized:

$$nTND(i) = \frac{TND(i) - TND_{min}}{TND_{max} - TND_{min}}. \tag{5}$$

   where $nTND(i)$ is the normalized TND of individual i, and TND_{max} and TND_{min} indicate, respectively, the maximum and minimum TND in the current population.

In the same way as for the PD measure, the smaller TND the better. The underlying motivation for its proposal can be explained as follows. The whole set of solutions in the population can be regarded as a connected graph, with the Euclidean minimum spanning tree of this graph being an optimized structure that reflects the distribution of the solutions of the current population in the search space. Then, for a given individual, the corresponding neighborhood can be defined by the other individuals connected to it, and finally, the crowdedness of this neighborhood can be said to represent its density.

Aggregated Fitness Evaluation

Based on the aforementioned discussion, the fitness value (to be minimized) of each individual i is calculated as the sum of both criteria:

$$f(i) = PD(i) + nTND(i) \tag{6}$$

It is worth noting that only the TND measure is normalized between 0 and 1. Hence, evolution proceeds by firstly minimizing PD, i.e., approaching the Pareto front; and then, as soon as some non-dominated solutions (i.e., with $PD = 0$) are found, $nTND$ becomes significant in the fitness evaluation, and a higher spread between the non-dominated solutions is promoted.

3.2 Replacement Mechanism

At each generation, each of the NP parental solutions is used to generate other NP offspring solutions. In the original DE algorithm, the offspring replaces its parent in the next generation if it has a better fitness value. In the case of multi-objective optimization, a different replacement mechanism is needed in order to incorporate the already mentioned properties of this kind of problem. To this aim, a three-step comparison method is proposed in this work, as follows.

Starting from the mixed population of size $2 \times NP$, containing the NP parental and the NP offspring individuals, firstly, the Pareto dominance relationship is considered: each pair (parent, offspring) is compared at a time, and the dominated one is immediately rejected.

In case the mixed population is still bigger than NP, the replacement mechanism proceeds to the second step, which uses the non-dominated sorting method proposed in the NSGA-II algorithm [4]. Briefly, at each round, the non-dominated individuals of the mixed population are chosen to survive to the next generation, and are removed from the mixed population. This is done iteratively up to the completion of the population for the next generation (i.e., NP chosen individuals after the first and second steps), or until there are no less than NP individuals with assigned rank values in the population.

If there are still individuals to be filtered for the next generation, the third step finally considers the TND values. At each iteration, the individual that has the lowest TND (i.e., the most crowded individual) is maintained, until the exact number of individuals for the completion of the new population is achieved.

3.3 Adaptive Operator Selection

As surveyed in Section 2.2, to implement the AOS paradigm, there is the need of defining two elements, the Credit Assignment and the Operator Selection mechanisms. The approaches used in this work will be now detailed in turn.

Credit Assignment: Normalized Relative Fitness Improvement
The Credit Assignment scheme is inspired from the one used in the PM-AdapSS-DE algorithm [9]; the differences are the use of a different and normalized calculation of the relative fitness improvements (which showed to perform better after some preliminary experiments) and in the already described fitness evaluation, specially designed for multi-objective optimization.

The impact of each operator application i is evaluated as the normalized relative fitness improvement η_i achieved by it, measured as:

$$\eta_i = \frac{|pf_i - cf_i|}{|f_{best} - f_{worst}|} \tag{7}$$

where f_{best} (respectively f_{worst}) is the fitness value of the best (respectively the worst) solution in the current population; pf_i and cf_i are the fitness values of the (parent) target vector and its offspring, respectively. As in [9], in case no improvement is achieved i.e., $pf_i - cf_i \geq 0$, η_i is set to zero.

All the normalized relative fitness improvements achieved by the application of operator (mutation strategy in this case) $a \in \{1, \ldots, K\}$ during each generation g are stored in a specific set R_a. Following [9], at the end of each generation g, a unique credit (or reward) is assigned to each operator, calculated as the average of all the normalized relative fitness improvements achieved by it:

$$r_a(g) = \sum_{i=1}^{|R_a|} \frac{R_a(i)}{|R_a|} \cdot \tag{8}$$

Operator Selection: Probability Matching
The Operator Selection mechanism used is the Probability Matching (PM) [18]. Formally, let the strategy pool be denoted by $S = \{s_1, \ldots, s_K\}$ where $K > 1$. The probability vector $P(g) = \{p_1(g), \ldots, p_K(g)\}(\forall t : p_{min} \leq p_i(g) \leq 1; \sum_{i=1}^{K} p_i(g) = 1)$ represents the selection probability of each operator at generation g. At the end of every generation, the PM technique updates the probability $p_a(g)$ of each operator a based on the received reward $r_a(g)$, as follows. Firstly, the empirical quality estimate $q_a(g)$ of operator a at generation g is updated as [18]:

$$q_a(g+1) = q_a(g) + \alpha \left[r_a(g) - q_a(g) \right] \tag{9}$$

where $\alpha \in (0, 1]$ is the adaptation rate; the selection probability is updated as:

$$p_a(t+1) = p_{min} + (1 - K \cdot p_{min}) \frac{q_a(g+1)}{\sum_{i=1}^{K} q_i(g+1)}. \tag{10}$$

where $p_{min} \in (0, 1)$ is the minimal selection probability value of each operator, used to ensure that all the operators have a minimal chance of being selected. The rationale for this minimal exploration is that the operators that are currently performing badly might become useful at a further moment of the search [18].

3.4 Adaptive Parameter Control of CR and F

The parameter adaptation method used here is similar to that used in the JADE algorithm [21]. Let CR_i^a denote the crossover rate for the individual i using operator $a \in \{1, \ldots, K\}$. At each generation, CR_i^a is independently generated according to a normal distribution with mean μ_{CR}^a and standard deviation 0.1:

$$CR_i^a = norm(\mu_{CR}^a, 0.1) \tag{11}$$

being regenerated whenever it exceeds 1. All successful crossover rates at generation g for operator a are stored in a specific set denoted as S_{CR}^a. The mean μ_{CR}^a is initialized to a user defined value and updated after each generation as:

$$\mu_{CR}^a = (1 - c) \cdot \mu_{CR}^a + c \cdot mean(S_{CR}^a) \tag{12}$$

where c is a constant and $mean(S_{CR}^a)$ is the arithmetic mean of values in S_{CR}^a.

An analogous adaptation mechanism is used for the scaling factor F_i^a. After some preliminary experiments, a difference with respect to the JADE algorithm [21] at this point is that the mean value μ_F^a is calculated by the root-mean-square of the values in S_F^a, instead of Lehmer mean.

4 Performance Comparison

In this section, three different empirical comparisons are presented. Firstly, the proposed Adap-MODE is compared with two state-of-the-art MOEAs, namely, NSGA-II [4] and GDE3 [12]. Then, in order to assess the benefits brought by the combined use of both adaptive parameter control modules, Adap-MODE is compared with four static variants, each using one of the four mutation strategies and a fixed values for control parameters ($CR = 0.5, F = 1.0$). Lastly, we compare Adap-MODE with its "partially-adaptive" variants, namely, the same MODE but using only AOS (and $CR = 0.5, F = 1.0$), and the same MODE but using only the adaptive parameter control of CR and F (and the mutation strategies being uniformly selected). This latter is done in order to evaluate the gain achieved by the combination of both modules, compared with each of the modules being independently applied.

4.1 Experimental Settings

For the sake of a fair empirical comparison, the parameters of the two state-of-the-art MOEAs are set as in the respective original papers. For the NSGA-II [4], $\eta_c = \eta_m = 20, p_c = 0.9, p_m = 1/D$, with D representing the dimension of the problem; and for GDE3 [12], $CR = 0.5, F = 1.0$. For the parameters of the proposed Adap-MODE method, the PM adaptation rate is set to $\alpha = 0.3$ and minimal probability $p_{min} = 0.05$, as in [9]; and the parameter c for the adaptive parameter control of CR and F is set to 0.1, as in [21], with CR and F being both initialized to 0.2. Lastly, the DE population size is set to 100.

In this work, the AOS mechanism implemented in Adap-MODE is used to select between the following four DE mutation strategies: (1) DE/rand/1/bin, (2) DE/current-to-rand/1/bin, (3) DE/rand/2/bin, and (4) DE/rand-to-best/2/bin. These are the same strategies used in some previous works [16,9]; no theoretical or empirical analysis was preliminary performed for their choice. It is worth highlighting that the AOS scheme is generic: any other set of mutation strategies could be considered here.

In order to compare the performance of the proposed and baseline approaches, ZDT [22] and DTLZ [5] test suites are considered as benchmark functions. The maximum number of generations is set to 300 for ZDT, and to 500 for DTLZ.

Two assessment metrics are used to quantitatively evaluate the performance of each algorithm at the end of each run, averaged over 50 runs. The Uniform Assessment (UA) metric [13] is used to evaluate the spread of the solutions, while the Hyper-Volume (HV) [24] is a comprehensive performance indicator. Generally, for the values of both UA and HV, the larger the better.

4.2 Experimental Results

The comparative results, for each of the are presented in Tables 1 to 3. Following the central limit theorem, we assume that the sample means are normally distributed; therefore, the paired t-test statistical test at 95% confidence level

Table 1. Comparative results of NSGA-II, GDE3 and Adap-MODE

		NSGA-II	GDE3	Adap-MODE	S
ZDT1	UA	4.433e-1/3.56e-2	2.359e-1/4.42e-2	**8.080e-1/1.62e-2**	†
	HV	3.65960/3.00e-4	3.65990/3.55e-4	**3.66193/3.15e-5**	†
ZDT2	UA	4.391e-1/4.68e-2	2.551e-1/4.98e-2	**8.069e-1/1.89e-2**	†
	HV	3.32618/3.21e-4	3.32673/3.06e-4	**3.32853/4.19e-5**	†
ZDT3	UA	4.252e-1/4.49e-2	2.069e-1/4.17e-2	**7.660e-1/1.98e-2**	†
	HV	4.80650/5.13e-2	4.81433/1.95e-4	**4.81463/4.81e-4**	†
ZDT4	UA	4.173e-1/4.69e-2	2.403e-1/4.64e-2	**8.055e-1/1.85e-2**	†
	HV	3.65413/4.04e-3	3.63033/1.84e-1	**3.66201/5.33e-4**	†
ZDT6	UA	4.529e-1/4.86e-2	2.226e-1/4.67e-2	**7.896e-1/2.27e-2**	†
	HV	3.03090/1.51e-3	3.04029/2.67e-4	**3.04183/1.62e-5**	†
DTLZ1	UA	3.742e-1/4.44e-2	5.256e-1/3.58e-2	**8.246e-1/1.48e-2**	†
	HV	0.967445/1.95e-3	0.965469/9.45e-4	**0.973582/2.75e-4**	†
DTLZ2	UA	3.688e-1/3.78e-2	4.868e-1/3.30e-2	**8.236e-1/1.84e-2**	†
	HV	7.33017/2.70e-2	7.31392/9.05e-3	**7.40523/1.14e-2**	†
DTLZ3	UA	3.353e-1/7.92e-2	4.857e-1/4.09e-2	**8.304e-1/1.72e-2**	†
	HV	6.41853/1.80e+0	5.85267/2.37e+0	**7.32465/5.76e-1**	†
DTLZ4	UA	**4.404e-1/9.19e-2**	2.532e-1/3.91e-2	2.654e-1/2.99e-2	‡
	HV	6.90792/7.55e-1	5.46000/1.10e+0	**7.02943/5.46e-1**	†
DTLZ5	UA	3.930e-1/4.63e-2	4.379e-1/3.85e-2	**7.866e-1/1.82e-2**	†
	HV	6.10048/1.42e-3	6.08543/1.83e-3	**6.10548/4.40e-3**	
DTLZ6	UA	2.939e-1/5.19e-2	2.652e-1/4.33e-2	**7.759e-1/2.18e-2**	†
	HV	5.86932/7.09e-2	6.10187/2.12e-3	**6.10732/4.88e-3**	†
DTLZ7	UA	4.102e-1/3.96e-2	4.491e-1/3.70e-2	**7.723e-1/1.86e-2**	†
	HV	13.15151/8.55e-2	13.19772/9.29e-2	**13.46486/7.43e-2**	†

is adopted to compare the significance between two competing algorithms, with the † indicating that Adap-MODE is significantly better than all its competitors in the corresponding Table, and ‡ representing that the best competitor significantly outperforms Adap-MODE. Moreover, the best results for each metric on each problem function are highlighted in **boldface**.

Starting with the comparison between Adap-MODE and the two state-of-the-art MOEAs, namely NSGA-II and GDE3, the results are presented in Table 1. These results clearly show that Adap-MODE is the best choice when compared to its competitors: it achieves the best results in 23 out of the 24 performance metrics, performing significantly better in 22 of them. The only exception is for the UA metric in the DTLZ4 problem, in which NSGA-II wins. It is worth noting that Adap-MODE performs around two times better than its competitors w.r.t. the uniformity metric UA in most of the functions, what might be largely attributed to the use of the proposed tree neighborhood density estimator by the fitness assignment.

Table 2 compares the performance of Adap-MODE with four static variants of it, each using one of the four available mutation strategies, without any adaptive parameter control. From these results, it becomes clear that there is no single

Table 2. Comparative results of Adap-MODE and its pure versions, following the same order of the problems as in Table 1

	Str.1	Str.2	Str.3	Str.4	Adap-MODE	S
UA	7.9e-1/1.9e-2	7.9e-1/1.9e-2	7.4e-1/2.6e-2	4.1e-1/5.2e-2	**8.1e-1/1.6e-2**	†
HV	3.662/3.4e-5	3.662/3.4e-5	3.656/2.2e-3	1.902/3.9e-1	**3.662/3.1e-5**	
UA	7.9e-1/1.5e-2	8.0e-1/1.9e-2	7.2e-1/2.7e-2	3.6e-1/6.3e-2	**8.1e-1/1.9e-2**	
HV	3.328/3.4e-5	**3.328/3.9e-5**	3.319/4.6e-3	1.905/2.4e-1	3.328/4.2e-5	‡
UA	**7.7e-1/2.1e-2**	7.5e-1/2.7e-2	5.1e-1/8.0e-2	3.4e-1/2.7e-2	7.6e-1/1.9e-2	
HV	**4.815/6.4e-5**	4.814/1.6e-3	4.775/1.3e-2	1.781/3.4e-1	4.814/4.8e-4	‡
UA	**8.1e-1/1.7e-2**	8.0e-1/1.6e-2	8.0e-1/1.9e-2	3.3e-1/3.9e-2	8.0e-1/1.8e-2	
HV	3.636/1.0e-1	3.662/3.8e-5	3.649/8.6e-2	0.0/0.0	**3.662/5.3e-4**	
UA	7.9e-1/1.9e-2	8.1e-1/2.0e-2	**8.2e-1/1.9e-2**	7.6e-1/4.6e-2	7.9e-1/2.2e-2	‡
HV	3.042/1.7e-5	3.042/2.4e-5	**3.042/1.5e-5**	3.041/3.1e-3	3.042/1.6e-5	
UA	**8.3e-1/2.1e-2**	8.2e-1/1.7e-2	8.2e-1/1.5e-2	4.7e-1/4.3e-2	8.2e-1/1.5e-2	
HV	0.97/1.0e-3	0.97/5.6e-4	0.969/7.1e-4	0.0/0.0	**0.973/2.7e-4**	†
UA	8.1e-1/1.8e-2	8.0e-1/2.0e-2	8.0e-1/1.9e-2	7.9e-1/2.5e-2	**8.2e-1/1.8e-2**	
HV	7.348/1.4e-2	7.337/1.4e-2	7.335/7.8e-3	7.303/8.8e-3	**7.405/1.1e-2**	†
UA	8.0e-1/1.8e-2	3.5e-1/3.9e-2	3.4e-1/3.2e-2	4.0e-1/3.9e-2	**8.3e-1/1.7e-2**	†
HV	6.538/2.0	0.0/0.0	0.0/0.0	0.0/0.0	**7.324/5.7e-1**	†
UA	2.5e-1/3.8e-2	2.4e-1/3.3e-2	2.5e-1/3.0e-2	2.3e-1/3.0e-2	**2.6e-1/2.9e-2**	†
HV	5.58/1.1	6.639/4.8e-1	6.359/7.7e-1	5.971/1.1	**7.029/5.4e-1**	†
UA	7.4e-1/2.2e-2	7.2e-1/2.3e-2	7.2e-1/2.1e-2	7.3e-1/2.8e-2	**7.8e-1/1.8e-2**	†
HV	6.073/3.4e-3	6.067/3.4e-3	6.065/3.9e-3	6.052/5.3e-3	**6.105/4.4e-3**	†
UA	7.9e-1/2.0e-2	7.9e-1/2.1e-2	**7.9e-1/1.7e-2**	7.6e-1/2.5e-2	7.7e-1/2.2e-2	‡
HV	6.107/4.4e-3	6.106/4.2e-3	**6.108/5.4e-3**	5.764/1.0	6.107/4.9e-3	
UA	7.6e-1/1.8e-2	7.7e-1/1.6e-2	7.4e-1/2.2e-2	5.1e-1/1.3e-1	**7.7e-1/1.8e-2**	
HV	13.412/5.6e-2	13.427/4.8e-2	13.346/7.3e-2	7.735/3.7	**13.46/7.4e-2**	†

mutation strategy that is the best over all the functions. For example, for the ZDT2 function, strategy 2 is the best in terms of HV, while strategy 1 is the winner for ZDT3. It is also worth noting that strategy 4 performs worst, while strategies 1 and 3 are the most competitive ones. This kind of situation motivates the use of the AOS paradigm. And indeed, Adap-MODE remains the best option in most of the functions, while achieving very similar performance in others.

The last comparative results, shown in Table 3, presents the performance of Adap-MODE compared with its "partially"-adaptive variants, one using only the AOS, and the other using only the adaptive parameter control of CR and F. From these results, it is not clear which of the adaptive modules is the most beneficial for the performance of Adap-MODE: at some functions, the "AOS only" method is better than the "parameter control only" one, while in others the opposite occurs. But these results clearly demonstrate that the combined use of both adaptive modules is better than their sole use, what is shown by the fact that Adap-MODE significantly outperforms them in most of functions, in terms of both UA and HV.

Table 3. Comparative results of Adap-MODE, Adap-MODE with AOS only and Adap-MODE with parameter control only

		CR/F(fixed)+AOS	CR/F(adapt.)+Unif.OS	Adap-MODE	S
ZDT1	UA	7.860e-1/2.08e-2	7.851e-1/2.42e-2	**8.080e-1/1.62e-2**	†
	HV	3.66162/2.97e-4	3.66066/2.69e-4	**3.66193/3.15e-5**	†
ZDT2	UA	7.809e-1/2.02e-2	7.793e-1/1.71e-2	**8.069e-1/1.89e-2**	†
	HV	3.32840/3.13e-4	3.32612/5.27e-4	**3.32853/4.19e-5**	†
ZDT3	UA	7.538e-1/2.83e-2	7.487e-1/1.52e-2	**7.660e-1/1.98e-2**	†
	HV	4.81448/1.18e-3	4.81228/1.18e-3	**4.81463/4.81e-4**	
ZDT4	UA	**8.127e-1/2.30e-2**	7.486e-1/6.12e-2	8.055e-1/1.85e-2	
	HV	3.64150/1.43e-1	3.65409/4.26e-2	**3.66201/5.33e-4**	†
ZDT6	UA	7.626e-1/2.34e-2	**8.078e-1/2.34e-2**	7.896e-1/2.27e-2	‡
	HV	3.04179/3.22e-5	3.04183/4.93e-5	**3.04183/1.62e-5**	
DTLZ1	UA	**8.247e-1/1.80e-2**	8.200e-1/1.73e-2	8.246e-1/1.48e-2	
	HV	0.969925/5.41e-4	0.917842/1.25e-1	**0.973582/2.75e-4**	†
DTLZ2	UA	8.096e-1/2.01e-2	8.224e-1/1.56e-2	**8.236e-1/1.84e-2**	
	HV	7.33762/1.10e-2	7.40368/9.20e-3	**7.40523/1.14e-2**	
DTLZ3	UA	6.365e-1/1.44e-1	8.289e-1/1.42e-2	**8.304e-1/1.72e-2**	
	HV	7.13704/3.70e-1	4.59535/2.92e+0	**7.32465/5.76e-1**	†
DTLZ4	UA	2.092e-1/3.32e-2	9.814e-2/4.33e-3	**2.654e-1/2.99e-2**	†
	HV	6.78321/6.03e-1	4.66216/1.09e+0	**7.02943/5.46e-1**	†
DTLZ5	UA	7.334e-1/2.26e-2	7.792e-1/1.95e-2	**7.866e-1/1.82e-2**	†
	HV	6.07005/3.69e-3	**6.10649/3.78e-3**	6.10548/4.40e-3	
DTLZ6	UA	7.739e-1/2.32e-2	**7.876e-1/2.00e-2**	7.759e-1/2.18e-2	‡
	HV	**6.10841/5.67e-3**	6.10640/4.14e-3	6.10732/4.88e-3	
DTLZ7	UA	7.621e-1/1.83e-2	7.634e-1/1.70e-2	**7.723e-1/1.86e-2**	†
	HV	13.42436/6.19e-2	13.43145/7.25e-2	**13.46486/7.43e-2**	†

5 Conclusion

In this paper, we propose a new DE algorithm for multi-objective optimization that uses two adaptive mechanisms in parallel: the Adaptive Operator Selection mechanism, to control which operator should be applied at each instant of the search; and the Adaptive Parameter Control, that adapts the values of the DE parameters CR and F while solving the problem. A tree neighborhood density estimator is proposed and, combined with the Pareto dominance strength measure, is used in order to evaluate the fitness of each individual. Additionally, a novel replacement mechanism is proposed, based on a three-step comparison procedure. As a consequence, the adaptive methods employed by the proposed algorithm, inspired from recent literature, are originally ported to the multi-objective domain.

Numerical experiments demonstrate that the proposed Adap-MODE is capable of efficiently adapting to the characteristics of the region that is currently being explored by the algorithm, by efficiently selecting appropriate operators and their corresponding parameters. Adap-MODE is shown to outperform two state-of-the-art MOEAs, namely NSGA-II [4] and GDE3 [12], in most of the

functions. It also performs significantly better, in most of the functions, than the same MODE with static parameters, and than the partially-adaptive variants using each of the two adaptive modules.

But there is still a lot of space for improvements. Firstly, for the fitness evaluation, more sophisticated schemes to control the balance between both convergence and spread could be analyzed. Regarding the AOS implementation, other schemes have already shown to perform better than PM in the literature and should also be analyzed in the near future, such as the Adaptive Pursuit [18] and the Dynamic Multi-Armed Bandit [8]; a more recent work, that also use bandits, reward the operators based on ranks [7], thus achieving a much higher robustness w.r.t. different benchmarking situations. In the same way, there are different alternatives for the adaptive parameter control of CR and F that could be further explored.

Another issue that deserves further exploration is related to the (hyper) parameters of the adaptive modules. In the case of Adap-MODE, the AOS requires the definition of the adaptation rate α and the minimum probability p_{min}, while the adaptive parameter control requires the setting of c. In this work, these parameters were set as in the original references, but further analysis of their sensitivity should be done. Ideally, Adap-MODE and the other methods used as baseline should also be all compared again, after a proper off-line tuning phase. Another important baseline would be the same MODE with off-line tuned CR, F, and mutation application rates.

Lastly, the extra computational time resulting from the use of these adaptive schemes should be further analyzed; although it is true to say that, in real-world problems, the fitness evaluation is usually the most computationally expensive step, all the rest becoming negligible.

Acknowledgement

This work is supported by Hong Kong RGC GRF Grant 9041353(CityU 115408).

References

1. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10, 646–657 (2006)
2. Das, S., Suganthan, P.N.: Differential evolution – a survey of the state-of-the-art. IEEE Trans. Evol. Comput. (in press)
3. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Proc. ICGA, pp. 61–69 (1989)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197 (2002)
5. Deb, K., Thiele, L., Laummans, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Abraham, A., et al. (eds.) Evolutionary Multiobjective Optimization, pp. 105–145. Springer, Heidelberg (2005)

6. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3, 124–141 (1999)
7. Fialho, Á., Ros, R., Schoenauer, M., Sebag, M.: Comparison-based adaptive strategy selection with bandits in differential evolution. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G., et al. (eds.) PPSN XI. LNCS, vol. 6238, pp. 194–203. Springer, Heidelberg (2010)
8. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)
9. Gong, W., Fialho, A., Cai, Z.: Adaptive strategy selection in differential evolution. In: Branke, J., et al. (eds.) Proc. GECCO. ACM, New York (2010)
10. Huang, V.L., Zhao, S.Z., Mallipeddi, R., Suganthan, P.N.: Multi-objective optimization using self-adaptive differential evolution algorithm. In: Proc. CEC, pp. 190–194. IEEE, Los Alamitos (2009)
11. Jia, L., Gong, W., Wu, H.: An improved self-adaptive control parameter of differential evolution for global optimization. In: Cai, Z., Li, Z., Kang, Z., Liu, Y. (eds.) Computational Intelligence and Intelligent Systems. CCIS, vol. 51, pp. 215–224. Springer, Heidelberg (2009)
12. Kukkonen, S., Lampinen, J.: GDE3: The third evolution step of generalized differential evolution. In: Proc. CEC, pp. 443–450. IEEE, Los Alamitos (2005)
13. Li, M., Zheng, J., Xiao, G.: Uniformity assessment for evolutionary multi-objective optimization. In: Proc. CEC, pp. 625–632. IEEE, Los Alamitos (2008)
14. Maturana, J., Lardeux, F., Saubion, F.: Autonomous operator management for evolutionary algorithms. J. Heuristics (2010)
15. Price, K.V.: An introduction to differential evolution. In: Corne, D., et al. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill, New York (1999)
16. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13, 398–417 (2009)
17. Robič, T., Filipič, B.: DEMO: Differential evolution for multiobjective optimization. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 520–533. Springer, Heidelberg (2005)
18. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Beyer, H.-G., et al. (eds.) Proc. GECCO, pp. 1539–1546. ACM, New York (2005)
19. Whitacre, J., Pham, T., Sarker, R.: Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Proc. GECCO, pp. 1345–1352. ACM, New York (2006)
20. Zhang, J., Sanderson, A.C.: Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions. In: Proc. CEC, pp. 2806–2815. IEEE, Los Alamitos (2008)
21. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13, 945–958 (2009)
22. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Comput. 8, 173–195 (2000)
23. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems, pp. 95–100. CIMNE (2002)
24. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. 3, 257–271 (1999)