Combining Gradient Techniques for Numerical Multi–Objective Evolutionary Optimization

Peter. A.N. Bosman
Centre for Mathematics and Computer Science
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
Peter.Bosman@cwi.nl

Edwin D. de Jong
Institute of Information and Computing Sciences
Utrecht University
P.O. Box 80089
3508 TB Utrecht
The Netherlands
dejong@cs.uu.nl

ABSTRACT

Recently, gradient techniques for solving numerical multi–objective optimization problems have appeared in the literature. Although promising results have already been obtained when combined with multi–objective evolutionary algorithms (MOEAs), an important question remains: what is the best way to integrate the use of gradient techniques in the evolutionary cycle of a MOEA. In this paper, we present an adaptive resource–allocation scheme that uses three gradient techniques in addition to the variation operator in a MOEA. During optimization, the effectivity of the gradient techniques is monitored and the available computational resources are redistributed to allow the (currently) most effective operator to spend the most resources. In addition, we indicate how the multi–objective search can be stimulated to also search *along* the Pareto front, ultimately resulting in a better and wider spread of solutions. We perform tests on a few well–known benchmark problems as well as two novel benchmark problems with specific gradient properties. We compare the results of our adaptive resource–allocation scheme with the same MOEA without the use of gradient techniques and a scheme in which resource allocation is constant. The results show that our proposed adaptive resource–allocation scheme makes proper use of the gradient techniques only when required and thereby leads to results that are close to the best results that can be obtained by fine–tuning the resource allocation for a specific problem.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Gradient methods*; I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance, Experimentation, Theory

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007 ...$5.00.

Keywords

Evolutionary Algorithms, Memetic Algorithms, Multi–Objective Optimization, Numerical Optimization, Gradients

1. INTRODUCTION

In many problems of interest, particularly in continuous optimization problems, local features of the search space can be highly informative of the directions in which better solutions may be found. Non–evolutionary machine–learning methods therefore often employ gradient descent. Gradient information has also been used in single–objective evolutionary methods under the name of *memetic algorithms* [15]. More recent is the use of multi–objective gradient techniques either alone or in combination with multi–objective evolutionary algorithms (MOEAs) [2, 5, 8, 13, 18, 19].

While some problems may benefit from the use of gradient information, in other problems the additional cost required to calculate gradients does not weigh up to the benefits. It is therefore important to determine for *which problems* the use of a gradient technique brings advantage, as is also implied by the No Free Lunch notion that no single method is optimal for all problems.

The idea of determining the utility of gradient techniques can be taken one step further by not only selectively applying the technique to certain problems, but also varying the probability of applying the technique during optimization.

Adaptively choosing the probability of operators has several advantages in addition to the potential improvements in efficiency. First, a practitioner wishing to apply a method with adaptive operator probabilities is relieved of the need to select and tune the different operator probabilities. Furthermore, by leaving the choice of the operator probabilities to the search algorithm, a larger part of the optimization task is automated. Finally, adapting the probabilities of the operators can render optimization methods more robust, as any unfavorable choices of parameters can be corrected during the course of the run.

In this paper, we investigate the combined use of three gradient techniques for numerical multi–objective optimization by an adaptive means of choosing how often to use each gradient technique. In addition to the related work on gradient techniques for numerical multi–objective optimization as cited above, related work therefore concerns the use of adaptive operator probabilities. An interesting find in biology that may appear related is that the E. Coli bac-

terium increases its probability of mutation when it runs out of food; as E.Coli cells begin to starve, the expression of DNA Polymerase IV is quadrupled, so that a mutation–causing enzyme is activated [14]. This biological mechanism appears to be aimed at guaranteeing survival. Our focus here however is on the engineering question of how efficient optimization may be achieved.

Perhaps the first use of adaptive operator probabilities is by Davis [6]. In this work, an operator receives credit if it produces offspring that improves over the best fitness in the current population, and additional credit is assigned to earlier operators along the lineage. The operator probability is adapted based on the ratio of received credit to the number of times the operator was applied.

Rather than adapting the probability of operators that apply to the whole genome, the probability of applying an operator can also be considered per bit. This idea has been used with the aim of preserving relevant schemata, based on uniform crossover [23]. Bitwise operator probabilities have also been used with mutation [1].

More recent work on adaptive operator probabilities is given by Julstrom's ADOPP [10]. ADOPP sets the probability of mutation and crossover operators proportional to their recent success in producing improved offspring. In more recent work, the resulting development of the operator probabilities is studied [11].

In the above mentioned works, adaptive resource–allocation is performed for single–objective optimization. In this paper, we specifically aim at multi–objective optimization.

The remainder of this paper is organized as follows. In Section 2 we briefly review existing gradient techniques for numerical multi-objective optimization. In Section 3 we present our adaptive resource–allocation scheme. In Section 4 we present the results from our experiments. Finally, concluding remarks are presented in Section 5.

2. GRADIENT TECHNIQUES

In earlier work we described three techniques to compute and use gradient information in multi–objective optimization [2]. Two of these techniques only use the gradient information of a single objective function at a time. A third approach was specifically designed on the basis of an extension of the notion of gradient to the multi–objective setting. Because in this paper we will make use of these three gradient techniques, we briefly recall them.

2.1 The techniques

2.1.1 Random–objective conjugate gradients

In this straightforward approach the conjugate gradients algorithm is applied to a randomly chosen objective. The resulting technique is called ROCG (Random–Objective Conjugate Gradients). It depends completely on the correlation between the objectives whether the best local improvement in a single objective also leads to an improvement in the other objectives. Since this is typically not the case, the percentage of local searches that leads to an improvement (i.e. a solution that dominates the solution from where the local search started) is typically small.

2.1.2 Alternating–objective repeated line–search

To reduce the probability of improving a single objective while making the objective value in the other objective worse, the objective that is searched locally can be altered during search. Care should still be taken not to let the local search in a single objective converge to a minimum. Doing so results in the same approach as ROCG. Hence a single line–search in the direction of the negative gradient of that objective can be performed in a single, alternatingly chosen objective. This process is repeated until a multi–objective local minimum is found. We refer to this technique as AORL (Alternating–Objective Repeated Line–search).

2.1.3 Combined–objectives repeated line–search

This final technique is best described as a multi–objective version of gradient descent. A line–search is performed in a promising direction. When the line–search terminates, a new line–search is initiated in a new promising direction found at the new location. It was shown that a set of directions can be computed such that each of these directions improves both objectives simultaneously [2]. This result is more general than other related work in the literature where only a single direction is found. Moreover, these directions are maximal in the sense that they do not dominate each other. Note that a line–search in this case does not aim to minimize a single objective but aims to find the best non–dominated solution in the given direction, i.e. the line–search in this strategy is itself a multi–objective search algorithm. We refer to this strategy as CORL (Combined–Objectives Repeated Line–search).

2.2 MOEA integration: fixed ratio

We employ a generational hybridization scheme. A gradient technique is applied to one or more solutions in a candidate set at the end of each generation. From earlier work we found that taking this candidate set to be the entire population (instead of the selected solutions or only the non–dominated solutions) gives the best results.

To control the ratio of search effort spent by the MOEA and by the gradient technique we use a ratio parameter ρ_e. We aim to keep the ratio of the number of evaluations required by the gradient technique and the total number of evaluations required so far equal to ρ_e. To this end, the gradient technique is applied only as long as the current actual ratio is smaller than ρ_e.

3. COMBINING GRADIENT TECHNIQUES

3.1 The technique

Although the CORL technique is often the most effective one, this isn't always the case [2]. In some cases the gradients of the individual objectives may already allow for improvements. The three different gradient techniques all have their strengths and weaknesses. Together however, they offer a diverse spectrum of ways to exploit the multi–objective gradient. To have the benefits of all three gradient techniques we propose to combine them.

3.2 MOEA integration: fixed ratios

The simplest way to combine the gradient techniques is to use a straightforward extension of the integration of a single gradient technique as presented in Section 2.2. A generational scheme is used in which now all gradient techniques are applied to one or more solutions at the end of each generation. Each technique is only applied as long as the current ratio of evaluations for that specific technique is smaller than

ρ_e. Note that since we now have three gradient techniques, we have $0 \leq \rho_e < \frac{1}{3}$. If, for instance, we set $\rho_e = 25\%$, then all operators, i.e. both the variation operator and the three gradient techniques, are allowed to spend an equal share, i.e. a quarter, of all evaluations.

3.3 MOEA integration: adaptive ratios

A fixed–ratio approach cannot in general be optimal. If one gradient technique is clearly superior to another gradient technique for a particular problem, it is more efficient to allow the more superior technique to spend more search effort. An intuitively more favorable integration is thus a combination of the gradient techniques such that during search the most effective gradient technique is assigned the largest probability. Or, if no technique is efficient compared to the base MOEA, to reduce the use of the gradient techniques to a minimum. This amounts to the adaptive allocation of resources to operators of interest [20].

Since the three operators all have a different way of exploiting gradient information, it makes sense not to allocate *all* resources to the operator that creates the *most* non–dominated solutions. In other words, the reward of applying a local search method is not to be measured only in the efficiency of the operator concerning the resources it requires per new non–dominated solution. An operator is additionally favorable if it is able to generate non–dominated solutions in a sparsely populated location, even if this requires more resources than another operator requires to generate new non–dominated solutions in a more crowded location. In the absence of an all–encompassing reward mechanism we therefore opt for a method that attempts to match the probability of applying an operator (i.e. a probability–matching approach [6, 9]) to its relative reward instead of maximizing the probability of the most rewarding operator (i.e. an adaptive pursuit approach [21, 22]).

To compute the number of solutions that each of the gradient techniques is to be applied to in the next generation we redistribute the total number of evaluations that was used previously. Redistribution will be done proportionally to the observed efficiency of the gradient techniques in terms of average number of improvements (i.e. generation of new non–dominated solutions) per evaluation.

3.3.1 Number of evaluations: $\mathcal{E}_o(t)$

We denote the number of evaluations that are to this end taken into consideration for a specific operator by $\mathcal{E}_o(t)$ in generation t, $o \in \{VAR, ROCG, AORL, CORL\}$. We denote the *actual* number of evaluations that an operator used in generation t by $E_o(t)$. For the gradient techniques, we take the number of evaluations for a specific technique to be just the number of evaluations that the technique actually used in the recent most generation, i.e. $\mathcal{E}_o(t) = E_o(t)$. For the variation operator of the base MOEA we sum all evaluations backwards over previous generations until the number of evaluations is at least as large as the largest number of evaluations used by any gradient technique, i.e. $\mathcal{E}_{VAR}(t) = \sum_{t'=t_{min}}^{t} E_{VAR}(t)$ where t_{min} is chosen as large as possible such that $\mathcal{E}_{VAR}(t) \geq \mathcal{E}_o(t)$ for all $o \in \{ROCG, AORL, CORL\}$ holds, keeping in mind that $t_{min} \geq 0$. The reason for doing this is twofold:

- *Fairness of comparison*
  The population size in the base MOEA is fixed and so is the number of offspring it generates in each generation. Unless large population sizes are used, the number of evaluations used by the gradient techniques in a single generation is typically much larger than the number of evaluations used by the base MOEA. Computing efficiency over the same number of evaluations is a much more fair basis of comparison.
- *Allowing an increase in calls to gradient techniques*
  If the base MOEA gets stuck and the efficiency of one or more gradient techniques is higher, we want to allow the number of evaluations per generation allocated to the gradient techniques to grow to facilitate an increase in calls. If we only take the number of evaluations by the base MOEA from a single generation, the total number of evaluations to redistribute per generation remains rather constant. For instance, assume that the variation operator resulted in 100 evaluations and each gradient technique used 200 evaluations in a single call. Furthermore, assume that only one gradient technique generates improvements. Then 700 evaluations are redistributed to that gradient technique. This leads to 3 calls to the best gradient technique in the next generation, which in turn leads to an expected number of evaluations of 600 by the gradient techniques and 100 by the variation operator. Hence the number of calls to the best gradient technique stays at most 3. However, because the gradient technique is truly superior, we would like the number of calls to that operator to be allowed to grow. In our approach, the counted number of evaluations of the base MOEA would have been 200 instead of 100. This would lead to 800 evaluations to be allotted to the best gradient technique, resulting in 4 calls in the next generation. Moreover, in a subsequent generation we would count back 800 evaluations for the EA to match the 800 evaluations used by the gradient technique. If the gradient technique is again superior, it will now be assigned 1600 evaluations, leading to an exponential growth of the number of calls to that gradient technique.

3.3.2 Number of improvements: $\mathcal{I}_o(t)$

Let $I_o(t)$ be the number of improvements obtained by an operator in generation t. It is computed as follows:

- *Variation operator*
  Count the number of offspring solutions that
  1. are not dominated by any solution in the set of selected solutions and
  2. dominate at least one solution in the set of selected solutions.
- *Gradient technique*
  Count the number of solutions that were subjected to the gradient technique and resulted in a solution that
  1. is not dominated by any solution in the population and
  2. dominates the solution that the gradient technique started from

This notion of improvement is *strict*. In addition to not being dominated, new solutions must dominate the solution(s) it was created from. However, to ensure a diverse

front during the search and also to ensure that the front may be expanded sideways once the search gets near the Pareto–optimal front, the second requirement can be dropped. This allows the search to perform "sideway" steps in addition to "forward" or "domination" steps.

Analogous to the number of evaluations, for the gradient techniques we have that the number of improvements $\mathcal{I}_o(t)$ that is used to compute the redistribution of the evaluations is just the number of improvements in the recent most generation, i.e. $\mathcal{I}_o(t) = I_o(t)$. For the variation operator on the other hand, the same summation is used as for the evaluations, i.e. $\mathcal{I}_{VAR}(t) = \sum_{t'=t_{min}}^{t} I_{VAR}(t)$.

3.3.3 Redistribution of evaluations

The reward $\mathcal{R}_o(t)$ that an operator o is assigned at the end of generation t is its efficiency in terms of the number of improvements per evaluation:

$$\mathcal{R}_o(t) = \frac{\mathcal{I}_o(t)}{\mathcal{E}_o(t)} \tag{1}$$

Moreover, we define $\mathcal{R}_o(t)$ to be 0 if $\mathcal{E}_o(t) = 0$. The higher the value of $\mathcal{R}_o(t)$, the more cost-effective operator o is. The proportional redistribution of the total number of evaluations $\mathcal{E}(t) = \sum_o \mathcal{E}_o(t)$ equals:

$$\mathcal{E}_o^{Redist}(t) = \frac{\mathcal{R}_o(t)}{\sum_{o'} \mathcal{R}_{o'}(t)} \mathcal{E}(t) \tag{2}$$

Now, let $\mathcal{C}_o(t)$ be the number of times gradient technique o was called in generation t. The number of calls $\mathcal{C}_o^{Redist}(t)$ that gradient technique o should be called in generation $t+1$ on the basis of the recent most evidence is just the redistribution of the evaluations divided by the average number of evaluations per call for that particular gradient technique:

$$\mathcal{C}_o^{Redist}(t) = \frac{\mathcal{C}_o(t)}{\mathcal{E}_o(t)} \mathcal{E}_o^{Redist}(t) \tag{3}$$

To compute the number of calls $\mathcal{C}_o(t+1)$ to gradient technique o in generation $t+1$, we use memory decay to ensure a smooth decrease in calls if the efficiency of o drops in subsequent generations. To implement memory decay, we use a running average $\mathcal{C}_o^{Run}(t)$ from which the discrete value for $\mathcal{C}_o(t)$ is derived. Because we want to stimulate the use of gradient techniques, we propose to not use memory decay if the redistribution in equation 3 indicates that the number of calls to a gradient technique should increase, i.e.:

$$\mathcal{C}_o^{Run}(t+1) = \begin{cases} \mathcal{C}_o^{Redist}(t) & \text{if } \mathcal{C}_o^{Redist}(t) \geq \mathcal{C}_o^{Run}(t) \\ \eta^{Decay}\mathcal{C}_o^{Run}(t) + & \\ \quad (1-\eta^{Decay})\mathcal{C}_o^{Redist}(t) & \text{otherwise} \end{cases} \tag{4}$$

In principle, we will use $\mathcal{C}_o(t) = \lfloor \mathcal{C}_o^{Run}(t) \rfloor$. Now, if at some point $\mathcal{C}_o(t) = 0$ holds, then gradient technique o will no longer be used throughout the run. However, its non–usefulness may be only temporary. Therefore, we propose to force a call to gradient technique o after a while. The number of generations to wait in generation t, denoted $\mathcal{W}_o(t)$, is just the inverse of the running average of the number of calls to perform in the next generation, i.e.:

$$\mathcal{W}_o(t+1) = \begin{cases} \lfloor 1/\mathcal{C}_o^{Run}(t+1) \rfloor & \text{if } \mathcal{W}_o(t) = 0 \\ \mathcal{W}_o(t) - 1 & \text{otherwise} \end{cases} \tag{5}$$

with $\mathcal{W}_o(0) = 0$. To ensure that after waiting one call to the gradient technique is performed, $\mathcal{C}_o(t)$ is defined as follows:

$$\mathcal{C}_o(t) = \begin{cases} 1 & \text{if } \mathcal{W}_o(t-1) = 1 \\ \lfloor \mathcal{C}_o^{Run}(t) \rfloor & \text{otherwise} \end{cases} \tag{6}$$

with $\mathcal{C}_o(0) = 1$.

4. EXPERIMENTS

4.1 Setup

4.1.1 Multi–objective optimization problems

Five of the problems we have used for testing have been taken from the literature on designing difficult and interesting multi–objective optimization problems and on comparing various MOEAs [7, 24]. Specifically, we have used the problems known as EC_i, $i \in \{1, 2, 3, 4, 6\}$. For specific details regarding the difficulty of these problems we refer the interested reader to the indicated literature. Their definitions are presented in Table 1.

We have designed two additional problems. These problems are labeled BD_i, $i \in \{1, 2\}$ in Table 1. The main reason for adding these two problems is because of their gradient properties. Both problems make use of Rosenbrocks function, which is a challenging function that requires proper gradient–exploitation in order to find the minimum. In function BD_1 it is easy to distribute points along the front. For any value that is obtained from Rosenbrocks function that is defined over all variables except the first one, a linear front is uniformly obtained by sampling x_0 uniformly in its domain of $[0; 1]$. To move such a front to Pareto optimality, all that is required is the minimization of Rosenbrocks function. Function BD_2 is harder than BD_1 in the sense that the objective functions overlap in all variables instead of only in the first one. This automatically decreases the usefulness of gradient techniques that only take into account one objective at a time. An additional difficulty of BD_2 is that the first objective is much easier to minimize than the second objective. As a consequence, the search is likely to converge first towards a small part of the Pareto front, i.e. an area near the solution that minimizes the first objective. In order to obtain the full Pareto front, the MOEA must be able to traverse along the Pareto front. Although in itself this is already a typical problem for most MOEAs, for BD_2 this additionally requires proper gradient exploitation because the second objective is Rosenbrocks function.

4.1.2 Performance indicator

To measure performance we only consider the subset of all non–dominated solutions in the population upon termination. We call such a subset an approximation set and denote it by $\mathcal{S}$. A performance indicator is a function of approximation sets $\mathcal{S}$ and returns a real value that indicates how good $\mathcal{S}$ is in some aspect. More detailed information regarding the importance of using good performance indicators for evaluation may be found in literature [3, 12, 25].

Name	Objectives	Domain
EC_1	$f_0 = x_0, \quad f_1 = \gamma\left(1-\sqrt{f_0/\gamma}\right)$ $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} x_i/(l-1)\right)$	$[0;1]^{30}$ $(l=30)$
EC_2	$f_0 = x_0, \quad f_1 = \gamma\left(1-(f_0/\gamma)^2\right)$ $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} x_i/(l-1)\right)$	$[0;1]^{30}$ $(l=30)$
EC_3	$f_0 = x_0$ $f_1 = \gamma\left(1-\sqrt{f_0/\gamma}-(f_0/\gamma)\sin(10\pi f_0)\right)$ $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} x_i/(l-1)\right)$	$[0;1]^{30}$ $(l=30)$
EC_4	$f_0 = x_0, \quad f_1 = \gamma\left(1-\sqrt{f_0/\gamma}\right)$ $\gamma = 1 + 10(l-1) + \sum_{i=1}^{l-1}\left(x_i^2 - 10\cos(4\pi x_i)\right)$	$[-1;1]\times$ $[-5;5]^9$ $(l=10)$
EC_6	$f_0 = 1 - e^{-4x_0}\sin^6(6\pi x_0)$ $f_1 = \gamma\left(1-(f_0/\gamma)^2\right)$ $\gamma = 1 + 9\left(\sum_{i=1}^{l-1} x_i/(l-1)\right)^{0.25}$	$[0;1]^{10}$ $(l=10)$
BD_1	$f_0 = x_0$ $f_1 = 1 - x_0 + \gamma$ $\gamma = \sum_{i=1}^{l-2}\left(100(x_{i+1}-x_i^2)^2 + (1-x_i)^2\right)$	$[0;1]\times$ $[-5.12;5.12]^9$ $(l=10)$
BD_2	$f_0 = \sum_{i=0}^{l-1} x_i^2$ $f_1 = \sum_{i=0}^{l-2}\left(100(x_{i+1}-x_i^2)^2 + (1-x_i)^2\right)$	$[-5.12;5.12]^{10}$ $(l=10)$

Table 1: The benchmark problems used for testing.

Here we use a performance indicator that uses knowledge of the optimum, i.e. the Pareto–optimal front. We define the distance $d(\boldsymbol{x}^0, \boldsymbol{x}^1)$ between two multi–objective solutions $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$ to be the Euclidean distance between their objective values $f(\boldsymbol{x}^0)$ and $f(\boldsymbol{x}^1)$. The performance indicator we use computes the average of the distance to the closest solution in an approximation set $\mathcal{S}$ over all solutions in the Pareto–optimal set $\boldsymbol{\mathcal{P}_S}$. We denote this indicator by $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}$ and refer to it as the distance from the Pareto–optimal front to an approximation set. A smaller value for this performance indicator is preferable and a value of 0 is obtained if and only if the approximation set and the Pareto–optimal front are identical. This indicator is ideal for evaluating performance if the optimum is known because it describes how well the Pareto–optimal front is covered and thereby represents an intuitive trade–off between the diversity of the approximation set and its proximity (i.e. closeness to the Pareto–optimal front). Even if all points in the approximation set are on the Pareto–optimal front the indicator is not minimized unless the solutions in the approximation set are spread out perfectly.

Because the Pareto–optimal front may be continuous, a line integration over the entire Pareto front is required in the definition of the performance indicator. In a practical setting, it is easier to compute a uniformly sampled set of many solutions along the Pareto–optimal front and to use this discretized representation of $\boldsymbol{\mathcal{P}_F}$ instead. We have used this approach using 5000 uniformly sampled points. The performance indicator now is defined as follows:

$$\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}(\mathcal{S}) = \frac{1}{|\boldsymbol{\mathcal{P}_S}|} \sum_{\boldsymbol{x}^1 \in \boldsymbol{\mathcal{P}_S}} \min_{\boldsymbol{x}^0 \in \mathcal{S}} \{d(\boldsymbol{x}^0, \boldsymbol{x}^1)\} \qquad (7)$$

4.1.3 Base MOEA

The base MOEA we use is the naive MIDEA [4]. This MOEA is an EDA specifically designed for multi–objective optimization. It has been shown to give good results on a wide variety of problems defined in both discrete and continuous parameter spaces. Moreover, it is fast and easy to understand, making it a good baseline algorithm. The following gives a brief overview of its main features. For specific details the interested reader is referred to the literature [4].

The naive MIDEA maintains a population of size n. In each generation it selects a subset of this population of size $\lfloor \tau n \rfloor$, $\tau \in [\frac{1}{n}; 1[$, to perform variation with. By means of variation $n - \lfloor \tau n \rfloor$ new solutions are generated which replace the solutions in the population that were not selected.

Selection is performed using a diversity–preserving selection operator. Since the goal in multi–objective optimization is both to get close to the Pareto–optimal front and to get a good diverse representation of that front, a good selection operator must exert selection pressure with respect to both aspects. The selection operator in the naive MIDEA does this by using truncation selection on the basis of domination count (i.e. the number of times a solution is dominated). If the number of non–dominated solutions exceeds the targeted selection size $\lfloor \tau n \rfloor$, a nearest–neighbour heuristic in the objective space is used to ensure that a well–spread, representative subset of all non–dominated solutions is chosen.

The variation operator is geometrical in nature and is specifically designed to provide an advantage over traditional variation operators. The selected solutions are first clustered in the objective space. Subsequently, the actual variation takes place only between individuals in the same cluster, i.e. a mating restriction is employed. The rationale is that variation inside each cluster can process specific information about the different regions along the Pareto front. Such a parallel exploration automatically gives a better probability of obtaining a well–spread set of offspring solutions. To further stimulate diversity along the Pareto front each new offspring solution is constructed from a randomly chosen cluster. Variation inside a single cluster is done by estimating a one–dimensional normal–distribution for each variable separately and subsequently drawing new samples from the estimated distribution.

4.1.4 General algorithmic setup

For selection we set τ to 0.3, conforming to earlier work [4] and the rule–of–thumb for FDA [16]. We allowed gradient techniques to perform 10 iterations each time they were called. Gradient information was approximated when required using $\Delta x_i = 10^{-13}$. Furthermore, we have used the Polak–Ribiere variant of the conjugate gradient algorithm [17]. We set ρ_e to 0.5 for the single-gradient MOEA integration, i.e. divide the number of evaluations equally among the gradient technique and the base MOEA. For the combined-gradient MOEA integration we set ρ_e to 0.25. Finally, we used $\eta^{Decay} = 0.75$.

It is important to note that all variables have a bounded range. If the variables move outside of this range, some objective values can become non–existent. It is therefore important to keep the variables within their ranges. However, a simple repair mechanism that changes a variable to its boundary value if it has exceeded this boundary value gives artifacts that may lead us to draw false conclusions about the performance of the tested MOEAs. If for instance the

search on problem EC_6 probes a solution that has all negative values for each of the variables x_i with $i \geq 1$, then the repair mechanism sets all these variables to 0. This is especially well possible during a gradient–search procedure because the gradient with respect to the second objective points in the direction of all negative values for variables x_i with $i \geq 1$. It is not hard to see that the solution resulting after boundary repair lies on the Pareto front. We have therefore adapted the local search operators such that local search never changes a solution into one that lies out of the problem range. Similarly, the sampling procedure of the naive MIDEA is changed to prevent the generation of solutions that are out of bounds.

4.2 Results

Adaptive operator allocation scheme

Figures 2 and 3 show convergence graphs, averaged over 100 independent runs, of the $\boldsymbol{D}_{\boldsymbol{\mathcal{P}_F \to \mathcal{S}}}$ indicator for all problems and all MOEAs with a population size of 500. To reduce the number of individual lines and because the individual gradient techniques were already addressed in depth in earlier work [2], we combined the results of the individual gradient techniques into a grey area that covers the range between the best and the worst results of the individual gradient techniques. In that earlier work, it was already found and explained that for problems EC_1, EC_2, EC_3 and EC_6, the relative contribution of the gradient techniques is inferior to the contribution by the base MOEA, especially in the beginning of the run. The base MOEA is already able to move the Pareto front by improving many solutions simultaneously. Gradient search can then not provide additional help fast enough because the number of evaluations required before a solution is improved is relatively large, making local search much more expensive. In addition, on problem EC_6 the contribution by gradient techniques was found to be inferior not only because of the parallel efficiency of the base MOEA, but also because the multi–objective gradient is simply harder to exploit in this problem. It can be seen from the convergence graphs that our adaptive resource–allocation scheme reduces the calls to the gradient techniques to a minimum for these problems and therefore results in a performance close to that of the base MOEA alone, outperforming all fixed–ratio resource–allocation schemes. The minimal application of the gradient techniques by our resource–allocation scheme on EC_6 is additionally illustrated in Figure 1.

On the EC_4 problem, especially the CORL gradient technique is known from our earlier work to be a good search operator for finding new non–dominated solutions. From the design of problems BD_1 and BD_2 we also know that gradient techniques can greatly aid the base MOEA. It can be seen from the results that our adaptive resource–allocation scheme now is better than the base MOEA, similar to the fixed–ratio resource–allocation schemes. In addition, with exception of the BD_1 problem, our adaptive resource–allocation scheme again outperforms the 25%–each fixed–ratio resource–allocation scheme. The adaptive application of the gradient techniques by our resource–allocation scheme on EC_4, on average stimulating especially the use of CORL, is additionally illustrated in Figure 1. On the BD_1 problem, the 25%–each fixed–ratio resource–allocation scheme is probably close to optimal when taking into account only the *number* of improvements per evaluation. Indeed, our adaptive resource–allocation scheme was found to spend on average 75% of all evaluations on gradient techniques on the BD_1 problem. The 50% CORL-only resource–allocation scheme is however still better (it is the lower–bound of the grey area). The reason why this specific division of resources is not picked up by our adaptive resource–allocation scheme is that the *size* of the improvements is not taken into account (i.e. the length of the improvement in the objective space). Taking this into account would lead to a weighted sum of improvements. It should however be noted that a proper weighting of the size of improvements is not straightforward since small improvements may still be extremely valuable when other techniques aren't able to make improvements.

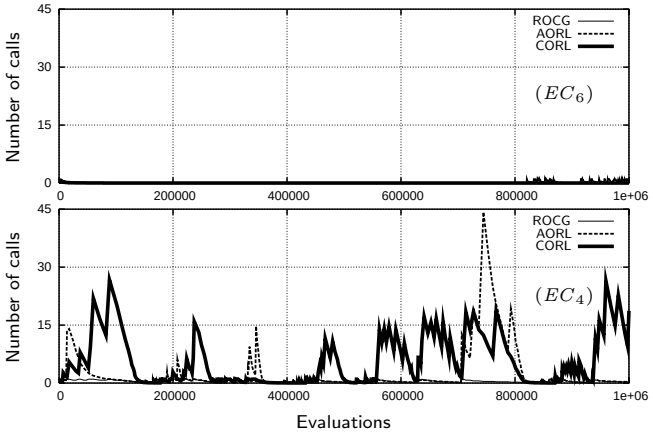

Figure 1: Typical runs on problems EC_6 and EC_4 showing the number of calls to gradient techniques.

Alternative improvement definition (sideway steps)

From the results it becomes immediately clear that by counting improvements also when a new non–dominated solution is created that does not necessarily itself dominate the solution(s) it was created from, i.e. allowing sideway steps, is beneficial. By design, we already knew that this property was likely to be strongly required for the BD_2 problem to efficiently find a good spread along the entire Pareto–optimal front. Indeed, the improvement obtained is remarkably large on this problem. The gradient technique is now able to traverse the Pareto–optimal front sideways and thereby obtain a good spread along the entire Pareto–optimal front. Even when the Pareto–optimal front can be reached in many parts, it may still be beneficial to explicitly allow sideway steps to obtain a nice spread along the Pareto front. In addition, having a good spread improves diversity, which in turn may stimulate exploration, possibly even leading to finding solutions that are closer to the Pareto–optimal front. The experimental results confirm this hypothesis as allowing sideway steps leads to results that are never worse. On the EC_4 problem, the results are even significantly better although this problem is not specifically designed to require movement along the Pareto–optimal front.

5. CONCLUSIONS

When applying EAs to specific, e.g. real–world, problems, it is often found that the addition of local search operators can aid in finding good solutions quickly and reliably. It is however an important question how these local search operators should be incorporated in the evolutionary cycle. To avoid specific heuristic decisions, an adaptive approach can be taken. This has been well documented in the literature

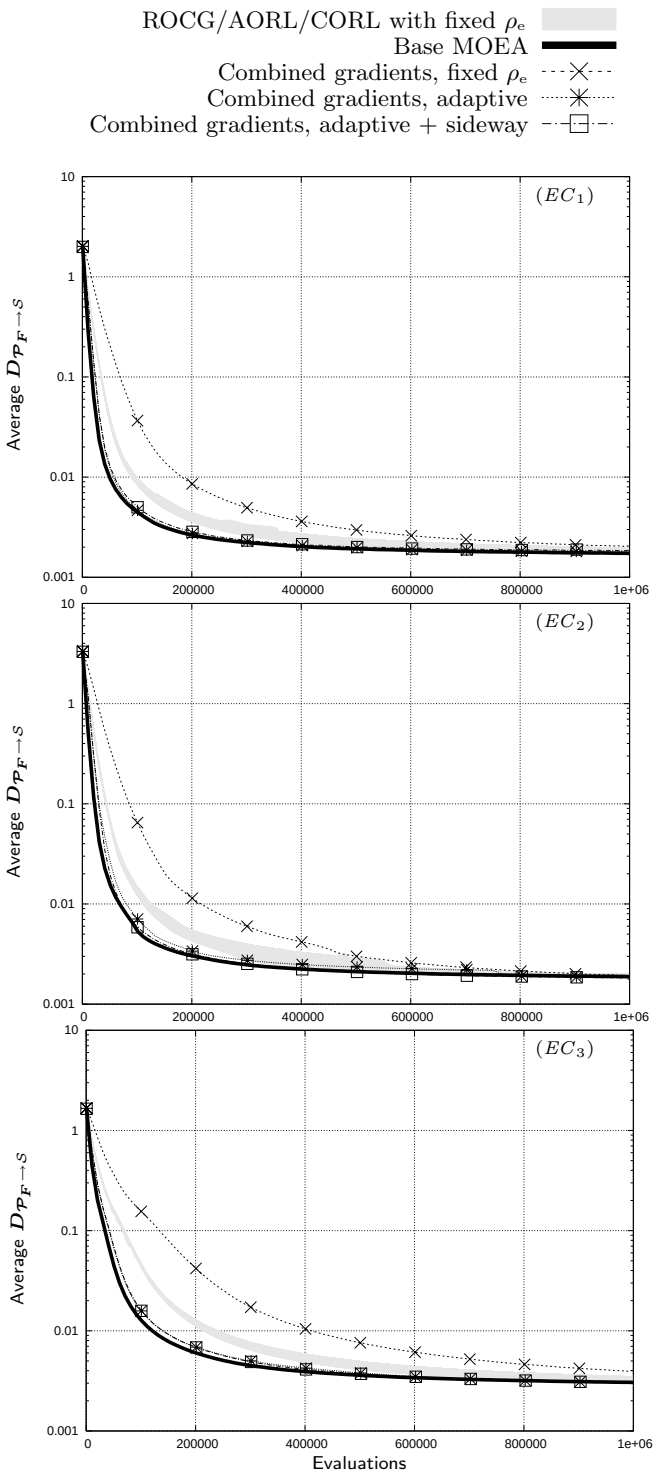

Figure 2: Convergence graphs for all MOEAs on problems EC_1, EC_2 and EC_3.

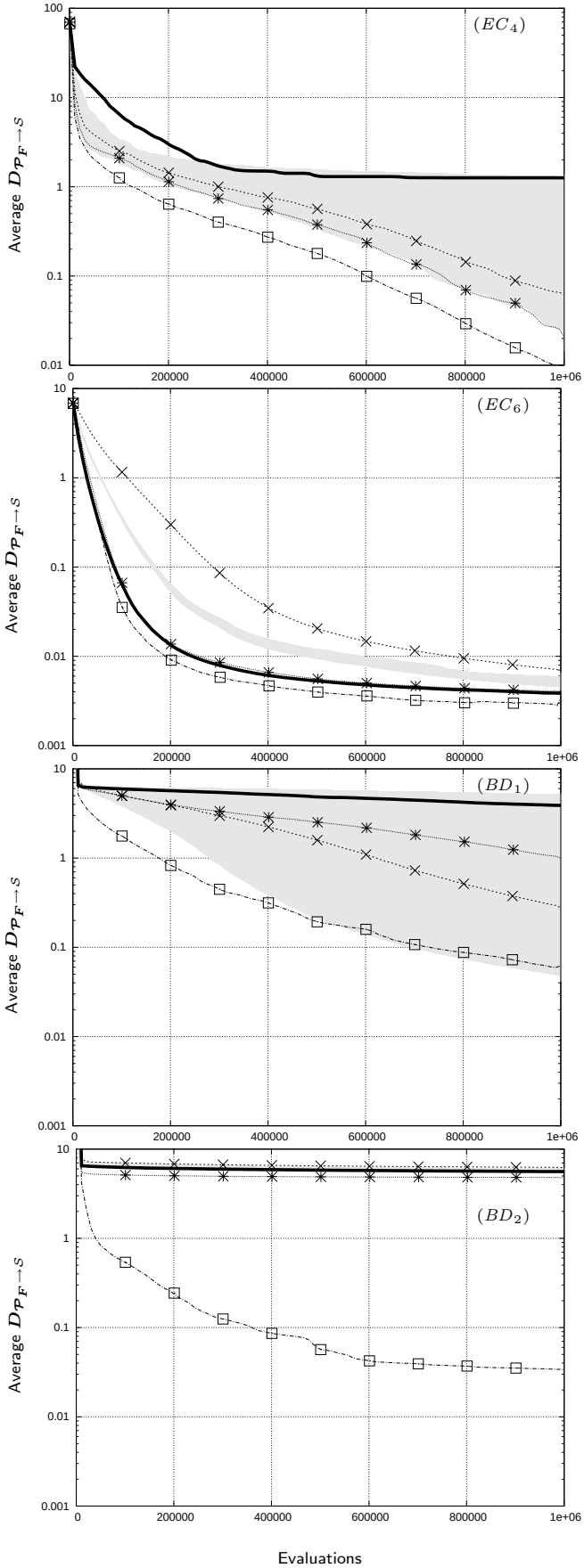

Figure 3: Convergence graphs for all MOEAs on problems EC_4, EC_6, BD_1 and BD_2. The legend is the same as used in Figure 2.

for single–objective optimization. In this paper we have presented an adaptive resource–allocation scheme for the use of three gradient techniques in numerical multi–objective evolutionary optimization. In each generation, the effectivity of each gradient technique and the variation operator of the MOEA is assessed in terms of the number of improvements per evaluation. The total number of evaluations used in the recent most generation is then redistributed proportionally to the effectivity of each operator for the next generation.

Our experiments show that our scheme allows for proper adaptive allocation of computing resources to the various operators. For problems where the gradient techniques contribute little, the scheme minimizes the use of the gradient techniques. For problems where the gradient techniques do contribute significantly, the scheme allows the contributing gradient techniques to be applied much more often.

In addition to the adaptive operator allocation scheme, we have shown that by counting (and accepting) improvements not as solutions that dominate the solutions they were constructed from, but as solutions that are not–dominated (i.e. a less strict notion of improvement), the application of gradient techniques allows the traversal of the Pareto front in a sideway fashion. This property can be greatly beneficial to the overall search efficiency of the MOEA, especially on problems where it is easy to converge to a specific part of the Pareto–optimal front, but it is hard to find substantially larger parts of the Pareto–optimal front. In our experiments, the results for using this notion of improvement were never worse than for using the more strict notion of improvement.

We conclude that the adaptive resource–allocation scheme that we proposed is a robust and flexible method for numerical multi–objective evolutionary optimization. Although specific choices for specific problems may yet lead to slightly better results, our scheme is effective enough to validly argue that there is no obvious need for practitioners to fine–tune these choices but that instead, our scheme can be used.

6. REFERENCES

[1] T. Bäck. Self-adaptation in genetic algorithms. In F. J. Varela and P. Bourgine, editors, *Proceedings of the 1st European Conference on Artificial Life*, pages 227–235, Cambridge, MA, 1992. MIT Press.

[2] P. A. N. Bosman and E. D. de Jong. Exploiting gradient information in numerical multi-objective evolutionary optimization. In H.-G. Beyer et al., editors, *Proceedings of the GECCO–2005 Genetic and Evolutionary Computation Conference*, pages 755–762. ACM Press, New York, New York, 2005.

[3] P. A. N. Bosman and D. Thierens. The balance between proximity and diversity in multi–objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7:174–188, 2003.

[4] P. A. N. Bosman and D. Thierens. The naive MIDEA: a baseline multi–objective EA. In C. A. Coello Coello et al., editors, *Evolutionary Multi–Criterion Optimization – EMO'05*, pages 428–442, Berlin, 2005. Springer–Verlag.

[5] M. Brown and R. E. Smith. Effective use of directional information in multi–objective evolutionary computation. In E. Cantú-Paz et al., editors, *Proceedings of the GECCO–2003 Genetic and Evolutionary Computation Conference*, pages 778–789, Berlin, 2003. Springer–Verlag.

[6] L. Davis. Adapting operator probabilities in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 61–69, San Francisco, California, 1989. Morgan Kaufmann.

[7] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[8] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[9] T.-P. Hong, H.-S. Wang, and W.-C. Chen. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, 6:439–455, 2000.

[10] B. A. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87, San Francisco, CA, 1995. Morgan Kaufmann.

[11] B. A. Julstrom. Adaptive operator probabilities. In J. Alander, editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, 1996.

[12] J. D. Knowles and D. Corne. On metrics for comparing non–dominated sets. In *Proceedings of the 2002 Congress on Evol. Comp. CEC 2002*, pages 666–674, Piscataway, New Jersey, 2002. IEEE Press.

[13] M. Lahanas, D. Baltas, and S. Giannouli. Global convergence analysis of fast multiobjective gradient based dose optimization algorithms for high-dose-rate brachytherapy. *Phys. Med. Biol.*, 48:599–617, 2003.

[14] J. Layton and P. Foster. Error-prone DNA polymerase IV is controlled by the stress–response sigma factor, RpoS, in Escherichia coli. *Molec. Microbiology*, 50(2):549–561, 2003.

[15] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, 1989.

[16] H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7:353–376, 1999.

[17] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes In C: The Art Of Scientific Computing*. Cambridge University Press, Cambridge, 1992.

[18] S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

[19] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multilevel evolutionary subdivision techniques. In Carlos M. Fonseca et al., editors, *Evolutionary Multi–Criterion Optimization – EMO'03*, pages 118–132, Berlin, 2003. Springer–Verlag.

[20] R. S. Sutton and A. G. Bargo. *Reinforcement Learning: an introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[21] M. A. L. Thathachar and P. S. Sastry. A class of rapidly converging algorithms for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15:168–175, 1985.

[22] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In H.-G. Beyer et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2005*, pages 1539–1546, New York, New York, 2005. ACM Press.

[23] T. White and F. Oppacher. Adaptive crossover using automata. In Y. Davidor et al., editors, *Parallel Problem Solving from Nature — PPSN III*, pages 229–238, Berlin, 1994. Springer–Verlag.

[24] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Computation*, 8(2):173–195, 2000.

[25] E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. Grunert da Fonseca. Why quality assessment of multiobjective optimizers is difficult. In W. B. Langdon et al., editors, *Proceedings of the GECCO–2002 Genetic and Evolutionary Computation Conference*, pages 666–674, San Francisco, California, 2002. Morgan Kaufmann.