Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing
No. B 20/2008

Efficient Evolutionary Optimization Algorithm: Filtered Differential Evolution

Timo Aittokoski

University of Jyväskylä 2008

Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing

Editor Raino A. E. Mäkinen
Technical editor Paula Takala

Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing
No. B 20/2008

Efficient Evolutionary Optimization Algorithm: Filtered Differential Evolution

Timo Aittokoski

University of Jyväskylä
Department of Mathematical Information Technology
P.O. Box 35 (Agora)
FI–40014 University of Jyväskylä
FINLAND
fax +358 14 260 2731
http://www.mit.jyu.fi/

URN:ISBN:978-951-39-9036-7
ISBN 978-951-39-9036-7 (PDF)
ISSN 1456-436X

Jyväskylän yliopisto, 2022

ISBN 978-951-39-3354-8
ISSN 1456-436X

Efficient Evolutionary Optimization Algorithm: Filtered Differential Evolution

Timo Aittokoski*

Abstract

Solving many real-life engineering problems requires often global and efficient (in terms of objective function evaluations) treatment, because function values involved are produced via time consuming simulations. In this study, we consider optimization problems of this type by discussing some drawbacks of the current surrogate assisted methods and then introduce a new population based optimization algorithm, which borrows features of the well-known Differential Evolution algorithm, but improves its efficiency by filtering away ineffective trial points.

1 Introduction

Many real-life industrial optimization problems are *computationally expensive*, i.e. they involve time consuming steps, for example black box -simulations, in evaluating the objective or constraint functions. These problems are often nondifferentiable, at least in practice due to various difficulties in calculating the derivatives, as well as multimodal, which prevents from using the classical (e.g. derivative based) optimization methods. Therefore, there is a high demand for easy-to-use and efficient (in terms of objective function evaluations needed) global optimization algorithms. To this end, in Section 3 we propose a new efficient optimization algorithm, Filtered Differential Evolution (FDE). It is based on Differential Evolution [38, 31] and improves its efficiency by filtering away ineffective trial points.

We are interested in solving a global single objective optimization problem formulated as

$$minimize \ f(x) \tag{1}$$

subject to $x \in S$.

Function $f : \mathbb{R}^n \to \mathbb{R}$ to be minimized is called an *objective function*. Minimization of the objective function is done by altering values of the decision or design variables forming the vector $x \in \mathbb{R}^n$. The decision variable values will lie then within the

*timo.aittokoski@jyu.fi

Department of Mathematical Information Technology, University of Jyväskylä, PO Box 35 (Agora), FI-40014 University of Jyväskylä, Finland

search (a.k.a. decision or design) space, i.e. in box constrained domain in $\mathbb{R}^n$ in our case. Sometimes all the points in the search space are not acceptable, and acceptable subset of the search space is called as the *feasible region* S. If only box constraints are used, the search space equals to the feasible region. The aim of the minimization process is to find the point with the minimum value of the objective function [1], i.e. optimum, and the corresponding design variable values. Point x^* is a global minimum, if $f(x^*) \leq f(x)$ with all $x \in S$. If there exists $\delta > 0$ so that $f(x^*) \leq f(x)$ with all $x \in S$, for which is valid $\|x - x^*\| \leq \delta$, point x^* is a local minimum.

There are several different methods to solve problem (1), and Evolutionary Algorithms (EA's) are a family of widely employed algorithms suitable for the task. EA's are general purpose stochastic and global optimization techniques inspired by the phenomena of evolution in the nature. EA's have been used with success to solve several different types of real life engineering optimization problems [11]. Differential Evolution (DE) [16, 31, 38] is an often cited and widely used population based evolutionary algorithm. It has been successfully applied to a wide range of engineering optimization problems, (e.g., ranging from optimization of water pumping systems [4] to internal combustion engine design [2]). Also in comparison with several types of evolutionary algorithms, for example such as Stochastic Genetic Algorithm (StGA) [39], Fast Evolutionary Strategies (FES) [43], Fast Evolutionary Programming (FEP) [44], Particle Swarm Optimization (PSO) [3], a Simple Evolutionary Algorithm (SEA) [41] and the Evolutionary Optimization (EO) [3] algorithm, the DE algorithm has performed favorably. For a summary of the results, see [31].

However, when dealing with computationally expensive problems, the use of even well developed EAs, such as DE, may not be reasonable, as the number of objective function evaluations required may prove to be prohibitive. This situation has led researches to consider how the efficiency of EAs could be improved. As a result, so-called meta-modelling or surrogate assisted schemes have been developed. In these, the computationally expensive original objective function is replaced with an inexpensive, lower fidelity surrogate model, which is utilized by EA (or in a more general level, by any other type of optimization algorithm) in solving the optimization problem. The surrogate model [23] may be implemented for example by using kriging [10], artificial neural networks (ANN) [21], radial basis function networks (RBFN) [7], support vector machines (SVM) [9],[40] etc. The surrogate is created by sampling, i.e., by evaluating the function values of an initial set of points within the search space. After the initial sampling, a surrogate model of the objective function based on the sampled points is computed. In the actual optimization process, the surrogate model is used by the optimization algorithm instead of the real expensive objective function, and the surrogate model may be updated occasionally to improve its accuracy.

The simplest meta-modelling schemes utilize merely a static surrogate, which is

[1] It may be useful to point out that the global optimum is not necessarily unique; in some cases there may exist several equally good global optima in the different parts of the search space. However, in the literature it is often implicitly assumed that the global optimum is a single point. In this study we rely on the similar assumption.

built in the beginning of the optimization process. No further updates are made of the surrogate [5, 23]. The selected optimization algorithm then uses the surrogate instead of the real objective function until the stopping condition of the optimization algorithm used is met. The use of a static surrogate may obviously and very easily lead the algorithm to converge to some false global optimum, i.e., optimum of surrogate, which is not the optimum of the real expensive objective function.

A more realistic approach is to update the surrogate during the process, which leads to improved accuracy of the surrogate, and thus the optimization algorithm should avoid converging to a false optimum. Several algorithms with some surrogate update scheme have been proposed. These include the assisting Genetic Algorithm (GA) with ANN [8], GA with kriging [32], GA with RBFN [28], and ES with kriging [15], to mention a few. With a surrogate updating approach it is not a trivial task to decide when and how should the surrogate be updated so that the optimization algorithm converges correctly with as few expensive objective function evaluations as possible. These issues are referred to as *model management* or in the context of evolutionary algorithms as *evolution control*, and they are discussed for example in [12], [24] and [33]. For a more profound discussion of meta-model assisted evolutionary algorithms, the reader is referred, for example, to [17, 23, 30].

A higher level of sophistication within a meta-modelling scheme is achieved with methods that reject the framework of the surrogate assisted evolutionary algorithms, and instead use the meta-model itself to determine at what point should the next expensive objective function evaluation be made, and thus exploit all the information available to the full extent. The point is determined by maximizing an *utility function* (known also as a figure of merit) reflecting the rewards of continuing sampling in a particular region [25], [35]. The purpose of the utility function is to balance local and global search by finding a trade-off between sampling in known, promising regions of the search space versus sampling in under-explored regions or regions where the variation in function values and the uncertainty of the prediction are both high. In this paper we refer to methods of this type as utility function based algorithms.

It is worth mentioning that the computational overhead (due to running the algorithm itself) of utility function based algorithms may be high as the number of evaluated points increases. This is due to the fact that maximizing the utility function is itself a global optimization problem, as is the fitting of the meta-model. So, in order to select a point to be sampled next, two global optimization problems must be solved, and this must be iterated as many times as samples are taken.

Probably the two most well-known utility function based algorithms are Efficient Global Optimization (EGO) [25] and *a radial basis function method for global optimization* [19], which both bear some resemblance to a method introduced more than a decade earlier, namely the P-algorithm [45]. The roots of the methods of this type can be traced back as far as to 1964 and work of H. Kushner [26]. Both EGO and the radial basis function method for global optimization have been shown to be very efficient in terms of objective function evaluations with known test problems [19].

Based on the discussion above, we can note that each type of the algorithms is

best suited for a certain type of problems. Problems where objective function evaluations take virtually no time at all, can usually be solved using known EA's, for example, DE. More expensive problems, where one objective function evaluation takes time from a few seconds to some minutes, can be solved for example by using surrogate assisted evolutionary algorithms. And finally, very expensive problems, where one function evaluation may take several hours, benefit most of more sophisticated (and also more complex) utility function based surrogate approaches, such as EGO, where all sampled information is exploited to its full extent.

In this work, we consider an approach which can be characterized to combine ideas of surrogate assisted approaches and standard evolutionary algorithms. The approach proposed is based on the DE algorithm, which we refine using some ideas borrowed from the meta-modelling community. Although we do not explicitly construct the surrogate, we use the information of previously sampled points to filter away some of the trial points so that expensive function evaluations are avoided in uninteresting areas of the search space. With this approach we aim to tackle problems of mediocre expense (where one objective function evaluation takes time from a few seconds to some minutes), where the performance of the traditional EA approach may not suffice.

The remainder of this study is organized as follows. In Section 2 we discuss some difficulties of the current EAs and meta-modelling algorithms. Then, in Section 3, we propose some alternate ways to treat these difficulties and formulate a corresponding algorithm. In Section 4 we show some experimental results, comparing the variants of the new algorithm with variants of DE using test problems from the literature. Finally, we conclude in Section 5.

2 Some drawbacks of the current approaches

2.1 Evolutionary algorithms

Population based evolutionary algorithms, such as DE, use a population of points to keep track of good solutions and to attain a more thorough search to avoid getting trapped in local optima. The basic idea of such algorithms, as we see it, is to exploit the information available in the population to focus the search to certain, promising areas of the search space. In other words, it is assumed that the population somehow approximates the distribution of the good solutions in the search space, and thus, during the optimization process the population should spatially shrink towards the global optimum.

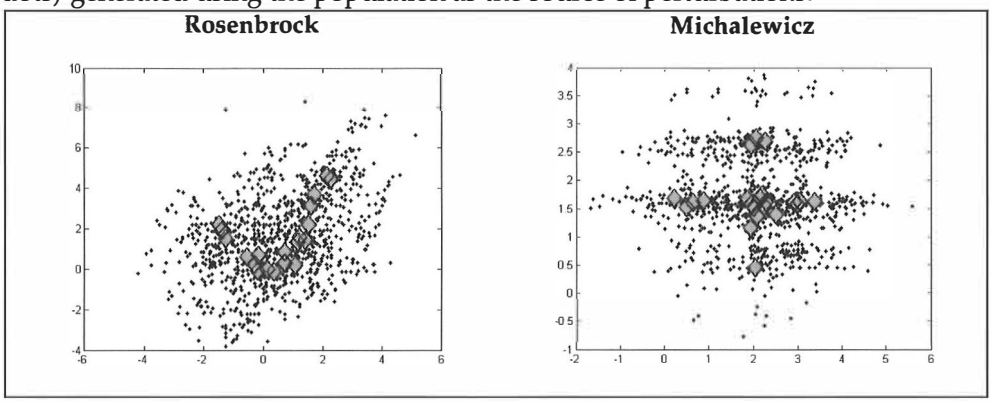
The idea that the population somehow approximates the distribution of the good solutions in the search space can be expressed also by utilizing the concept of a *sublevel set*, which is a subset of the search space where the function attains values less than or equal to a given threshold value. The population P with its near neighborhood can be considered as a rough approximation of the sublevel set of the objective function f, where the threshold value is the objective function value of the worst population member. For a more thorough discussion, the reader is referred to [34].

With the concept where the population is seen as an approximation of a certain sublevel set, it seems plausible that the algorithm should ideally sample only the area inside and near the given sublevel set, i.e., near the current population. This idea may be further clarified by noting that an optimization process begins with an initial population (with a sufficient size and diversity) which is somewhat evenly distributed around the search space, but then the population starts to adapt to some shape based on the objective function landscape. This particular shape is assumed, because the points from other areas are detected as inferior by the objective function value and excluded from the population. In this sense, when a population adapts and shrinks to a certain shape, it happens for a reason, and it makes little sense to evaluate more points from the areas which supposedly caused the contraction to a certain shape in the first place.

Unfortunately, several EA's may severely fail to sample the search space inside or near a given sublevel set in certain cases. If the sublevel set is disconnected, i.e., the population forms separate clusters (as it is often the case with multimodal functions where global optimization is needed in the first place), or if the sublevel set is strongly curved, elongated, or otherwise of irregular shape, the trial point generation strategy is inefficient because of the high number of the generated trial points that do not coincide with the given sublevel set, but are instead located outside of it. This behavior is clearly demonstrated in Figure 1, where the distribution of the population of 22 individuals in the search space is shown (as gray diamonds) after 300 objective function evaluations using DE with trial point generation strategy *DE/rand/1* (see Section 3) to solve the Rosenbrock and Michalewicz test functions (see Section 4). Using this static population of 22 individuals (no updates were made to population), 1100 trial points (black dots in the figure) were produced with the help of the DE's trial point generation strategy to illustrate which regions of the search space are attainable. In both of the cases the population, after 300 objective function evaluations, follows already the shape of the sublevel set of the given function, which definitely is not the case with the generated trial points. Indeed, when the objective function values of the generated trial points are checked using the worst objective function value in the population as the threshold level (which defines the sublevel set of the current population), it is to be noted that with the Rosenbrock function only 31% of the points have a better value than the chosen threshold, i.e., reside inside the given sublevel set, and with the Michalewicz function 60%. The better percentage with the Michalewicz function is explained by the fact that the sublevel set of the function consists of perpendicular straight regions, and these regions are easier to reach by linear combinations (used in the DE point generation mechanism) of the current population members. With the population distributed to separate clusters, the percentages would be even worse. This example shows that a remarkable number of objective function evaluations is unnecessarily wasted.

As seen in the figures above, the actual behavior of DE contradicts with the intuitive need to follow the shape of a given sublevel set. Within this framework of thought it also seems obvious that the well-known exploration/exploitation (i.e.,

Figure 1: Population after 300 evaluations (gray diamonds), and 1100 points (black dots) generated using the population as the source of perturbations.



balance between global and local search) dilemma [13, 27, 29] within the EA community originates from the use of less developed trial point generation strategies. Once we have been able to shrink the sublevel set justifiably, we should no longer explore farther locations of the search space, as we already implicitly know that these areas contain only inferior solutions.

Irrational behaviour of DE, as well as several other EA's, is mainly due to an inefficient point generation strategy and lack of memory; only points in the current population are kept in memory, and all previously evaluated points are completely neglected by the trial point generation mechanism. For these reasons, the algorithm may repeatedly sample trial points at regions where only solutions with bad objective function values have been found earlier during the optimization process, and intuitively it seems clear that on these regions there is nothing to gain. Further, there is no mechanism to prevent sampling very close to, or even at the location of the already sampled solutions. This behaviour leads to splurging of objective function evaluations, which is not acceptable in the case of computationally expensive problems.

2.2 Surrogate assisted evolutionary algorithms

As mentioned earlier in Section 1, with a surrogate assisted approach, model management, i.e., evolution control, is not a trivial task; it is difficult to decide when and how should the surrogate be updated so that the EA converges correctly with as few expensive objective function evaluations as possible. In [24], two basic methods for evolution control are identified: individual based control, and generation based control.

In the individual based control, a certain number of individuals in a population is chosen and evaluated with the expensive objective function, and the surrogate is updated with this new additional information. Individuals may be chosen randomly,

or by selecting the best ones based on the predicted function value of the surrogate. This approach is used for example in [18] and [15]. An obvious problem in this approach is to select the most efficient number for individuals to be controlled, and also the strategy to choose the individuals.

In the generation based control, the whole population is evaluated with the expensive objective function after a certain number of generations. This approach is used for example in [8]. The surrogate may also be updated after some local convergence criterion for running the EA on the surrogate has been fulfilled. This approach is used in [32], where convergence is obtained when the best solution of a predetermined number of generations has not changed by more than a user-defined value. Also with the generation based control it may be problematic to decide the most efficient schedule for controlling the population. Further, it may be problematic to define the problem specific user defined values for the change in solutions.

Obviously, there may be also combinations of individual and generation based control (e.g. [14]), or the surrogate may even be updated incrementally, after each evaluated point (e.g., [28]). Some evolution control methods are discussed and compared in [33], but there are no obvious answers on how to manage evolution control in the most efficient manner.

Unless all sampled points are used in the creation of the surrogate model, it may also be difficult to select a proper set of points to be used. If all sampled points are used, the surrogate is globally most accurate, but may lack some local details of the objective function landscape if the surrogate model is not flexible enough. Further, creation of the surrogate model may become prohibitively time consuming as the number of sample points grows [23], even to the point where the original and expensive objective function is more economical to evaluate. One example of this was encountered in [1] where EGO was used to solve an engine optimization problem.

Usually not all sampled points are used in the surrogate, as it is sufficient to model some neighborhood of the current population, as shown by, e.g., [18] and [8]. It is interesting to note that the idea of modelling the neighborhood of the current population coincides with the concept of sublevel sets: the accuracy of the surrogate model is emphasized only in the areas near the current population. A common way to select a subset (i.e., a training set) of all sampled points to fit the surrogate is to use some or all points of the population immediately after generation based control, or in the case of individual based control, to replace some number of the worst or randomly selected sample points from the current training set with freshly evaluated better individuals. Also more intelligent approaches exist, see e.g., [15] where local surrogates are constructed using k-nearest neighbors and kriging. In [14], mutual distance and objective function values of the points are taken into account, in order to keep the points of the training set scattered and yet sufficiently close to the current population.

Based on the issues discussed in this subsection, we can conclude that evolution control is far from being a self-evident task, and may require serious tweaking with various parameters. Further, unless local approximations are used, fitting the surro-

gate may become very time consuming as the optimization process progresses and the number of evaluated sample points grows.

2.3 Surrogate algorithms

Although utility function based surrogate algorithms, i.e., the ones that use the meta-model itself to determine at what point should the next expensive objective function evaluation be made, are the most efficient in terms of number of objective function evaluations. They are also the most complex ones to implement. For example, in case of RBFs, it may not be obvious how to select the parameters of the algorithm. In [19], issues such as selection of the type of radial basis functions and polynomial degree, the decision to trust the model (no need to explore the unsearched areas) or not to trust it (need to explore the unsearched area) and how to maximize the utility function are considered as open problems.

Especially with kriging based approaches, e.g., EGO [25], a cumulative number of sample points may make the fitting of the surrogate overly expensive at some point of the optimization process. This is in contradiction with the fact that with these approaches all evaluated points should be used to fit the surrogate in order to get globally as accurate surrogate as possible.

We conclude this section by noting that traditional EA's suffer from the lack of memory and inefficient trial point generation strategies, and it may be difficult to set proper values for the control parameters (e.g., population size, F and CR in the case of DE). On the other hand, these algorithms themselves are rather easy to implement and use, and their computational overhead is negligible.

With surrogate assisted EA's, evolution control may be a problematic issue and it may require some tedious parameter tuning. Also, it is not obvious which type of surrogate should be selected. Further, implementing the surrogate model may not be straightforward, as it may require some profound understanding.

Utility function based surrogate algorithms are obviously the most efficient ones of the three types discussed, but they are complex in their implementation. Running them may require more computational power, as two global optimization problems need to be solved in order to decide the location for the next sample.

From these premises we propose our algorithm, striving for simplicity and efficiency, in the following section.

3 Filtered Differential Evolution, FDE

In the previous section, we discussed some problems of the current, widely used approaches. Here we propose a complementary algorithm as a partial response to the presented difficulties.

Although our general approach of filtering presumably inefficient trial points away could in principle be adapted to enhance the performance of several different

population based algorithms (such as GA, DE, PSO, etc.), we chose to utilize the basic mechanisms of DE, because it is rather efficient and widely utilized. Thus, the basic structure of our algorithm is similar to that of DE, except the fact that we filter out some inefficient trial points, and try to select only better ones for expensive evaluation. For this reason, we refer to the proposed algorithm as Filtered Differential Evolution, FDE.

As we use the DE as an integral part of our proposed algorithm, it is in order to briefly describe the basic functioning of it. DE is a stochastic, population-based, direct search algorithm for global optimization, and it employs the population to produce the perturbations necessary for the *trial point generation*. In each iteration of the DE, the new trial points must compete against a prescribed target i.e. parent point. Each member of the population serves in turn as a target point, and the better one of the target and trial points is selected for the population in the next generation. During the years several different trial point generation strategies have been proposed, see, for example, [31], but in this study we utilize two trial point generation strategies: one originating from a so-called *classic DE*, denoted as *DE/rand/1* strategy, and another one, *DE/local-to-best/1* strategy [37].

In DE, each point in the population is selected in turn as a target point x_i, and a new trial point is generated. In the *DE/rand/1* trial point generation strategy, three distinct points, x_{r_0}, x_{r_1}, and x_{r_2}, are randomly selected from the population to create a perturbed, i.e., mutated point v_i,

$$v_i = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}), \tag{2}$$

where the scale factor F is a positive real number that controls the rate at which the population evolves. In the *DE/local-to-best/1* strategy, the perturbed point is created as,

$$v_i = x_i + F \cdot (x^* - x_i) + F \cdot (x_{r_1} - x_{r_2}), \tag{3}$$

where x^* is best individual in the population.

The perturbed point v_i may be recombined with the target point x_i to create the trial point t_i using uniform crossover, where the trial point is constructed by selecting vector components from either the perturbed point or the target point. The user-defined crossover probability CR controls the fraction of the vector components that are copied from the perturbed point; for more information, see [38]. If the crossover is not used, the perturbed point v_i itself becomes the trial point. The trial point is allowed to replace the target point in the population for the following generation if and only if it yields an equal or better objective function value than the target point. Otherwise, the target point x_i remains in the population. The population is updated once at the end of each generation.

As shown in Subsection 2.1, the point generation mechanism of DE is wasting some function evaluations as the points produced by it do not necessarily follow the shape of the sublevel set, partly due to lack of memory. To change this behavior, in FDE, all evaluated points are kept in memory, with the corresponding objective function value, and this data is used to filter out the trial points that presumably would not be accepted to the next generation. More specifically, for each parent

in the population one or more trial points are generated, the one with the best predicted value is selected, and if the best predicted value is not better than the parent's objective value, the trial point is filtered away, i.e., excluded from consideration, and the expensive objective function evaluation is avoided.

The rationale to generate more than one trial point for each parent is to extract more information about the search space, and thus make the search more effective. In some algorithms, such as EGO, an explicit optimization run is executed on the surrogate to select the point which maximizes the expected improvement in the objective function value. In our approach, we simply select the trial point with best predicted objective function value. It may be worth pointing out that the computational overhead of our algorithm increases relative to the number of trial points generated, and we suppose that a higher number of trial points produces also better efficiency. Thus, the user may balance between efficiency of the search and cost of the objective function evaluation by selecting a different number of trial points to be generated; with expensive objective function evaluations it is reasonable to explore the surrogate more extensively by selecting a higher number of trial points to be generated.

To filter out bad trial points, previous samples are used to predict the objective function value. In contrast to all methods discussed earlier, no explicit surrogate model is constructed to predict the objective function value, but in FDE we rather use a simple nearest neighbor interpolation as a base of the prediction. Further, also in contrast to most surrogate assisted EA's, we take into account the uncertainty of the prediction (in a somewhat similar manner as in the EGO algorithm) by checking how far the trial point is from the closest known sample point and multiply this distance with factor L, which depicts the variation in the objective function landscape, and is derived (see below) from the current population.

After this short introduction, we now can present the steps of the FDE algorithm as follows:

1. Initialize the population (randomly or using some space filling pseudo random sequence to guarantee a sufficient diversity of solutions).

2. While stopping criterion is not met:

   (a) For each parent in the population:

      i. generate the user specified number (one or more) of trial points (using mutation and crossover operators of DE),

      ii. predict the objective function values of all the trial points, select the best value, and set this value as a trial point value,

      iii. filtering: if the trial point value is better than the parent's objective value, evaluate the trial point using the real objective function, and set this value as a trial point value,

      iv. add the current trial point to the child population.

(b) Select the members for the next population by choosing in a pairwise manner the better member from the parent and the child populations, using the respective parent and child pairs in comparison

As predicting the objective function value of the new trial point is a focal point of the proposed algorithm, we give here a more detailed explanation of it. Because the prediction of the trial point value will inevitably be more or less inaccurate, this must be somehow taken into account. There are two factors affecting the accuracy of the prediction: the distance to the closest evaluated sample point, and irregularity of the objective function landscape. Obviously, the closer we are to some evaluated sample point, the more accurate will the prediction be. Also, when the objective function landscape is very flat (low irregularity), there won't be much error. Thus, it seems logical to assume that the predicted objective function value may deviate from the objective function value of the closest already evaluated point by the product of the distance (from the trial point to the closest evaluated sample point) and the measure of irregularity.

As a measure of irregularity we use factor L, which is calculated based on the information contained in the current population. For each point, i.e., member m in the population, a distance d_m to the closest neighbor m_c is computed, and the slope L_m between these two points is calculated with respective objective function values as $L_m = \frac{|f(m)-f(m_c)|}{d_m}$. Now, L is chosen as the maximum of all L_m values, and thus it reflects the maximum irregularity in the given population. The value for L is computed once for every generation. To accomplish this, the computation of a point-to-point distance matrix of the whole population is required to identify the nearest neighbors for each point. It may be noted that in some sense L is a rough estimate of a local Lipschitz constant (a measure of the maximum rate of change of a function in some given region, see, for example, [20] and [42]) for the objective function.

One may argue that L should be calculated based on all sampled points, instead of only points of the current population. Unfortunately, this approach would not work, because we wish to estimate the irregularity of the search landscape in the neighborhood of the given sublevel set, i.e., locally. Essentially this means that in the beginning of the optimization process the sublevel set defined by the population covers almost the whole search space, and thus the value of L may be expected to be very high. When the optimization process progresses, the sublevel set shrinks, and also the value of L decreases. Finally, in the end of the process, when the population has concentrated around the global optimum, the search landscape occupied by the population is (and actually must be) very flat, the corresponding sublevel set is very small, and the value of L is zero or very close to it.

If the value of L was calculated based on all sampled points, it would remain high throughout the optimization process, and hinder the convergence to a global optimum. This is due to the fact that the population should concentrate around the global optimum during the process, and in this phase the points in the population are more densely distributed than the other evaluated points surrounding the

population. In this case, the predicted objective function values would get good values around the population (because of the large inter-point distances), i.e., outside the sublevel set defined by the population, and thus the search would concentrate mainly outside the desired sublevel set. This explains why the value of L should approach zero as the optimization process progresses, and this is realized by computing L from the current population.

With the factor L, and given the trial point t we can now compute the prediction of the objective function value. As we are dealing with minimization problems, we are interested to know how much the objective function value can improve at the maximum at a given point. Thus, we are making an optimistic prediction. First we compute the distance d_t from t to its nearest neighbor t_{nn} among all so far evaluated sample points. This requires computation of distances between t and all so far evaluated samples, which is a negligible cost (compared to that of one expensive objective function evaluation) if the number of evaluated points remains in the order of thousands. The predicted objective function value is now $f_{pred} = f(t_{nn}) - (L * d_t)$. In short, we use the objective function value of the nearest already evaluated neighbor as an estimate, and this estimate is reduced by the product of the current L and the distance to the evaluated neighbor. In this way, the possible inaccuracy of the prediction is incorporated into a predicted value in an optimistic way.

The prediction of the objective function value by the aforementioned method possesses some intuitively appealing features. First, in the beginning of the optimization process while the population is very diverse, also the irregularity (and the value of L) is high. In this phase, it is obviously possible, by the predicted objective function values, to exclude only few solutions at very bad areas of the search space, thus allowing good global search properties. When the process progresses, the value of L decreases, and the prediction becomes a little bit less optimistic, thus reducing the allowable search area, wherein the trial points are not directly filtered away, more to a shape of the sublevel set defined by the current population. This greatly improves its efficiency over the basic DE. Anyhow, if the trial point at this phase appears at the large unsearched area (where the distance to the closest evaluated sample is large, and thus the predicted objective function value is good), it still has a chance to get evaluated with the real expensive objective function. Finally, when the population concentrates more and more, the value of L further decreases, and the filtering by the predicted objective function values allows the search only very close to the current population, thus changing emphasis more in to direction of local search. At the same time, the proposed method avoids sampling at, or very near, already known points, as the predicted value attains its maximum value in between two already known solutions (because this maximizes the distance to the closest evaluated point).

4 Numerical results

In this section, we compare the results of our new algorithm to other DE implementations. More specifically, we use the Matlab implementation of the original DE

(ODE) from [36], our own implementations of DE without filtering (DE), with filtering using only a single trial point (FDE), and with filtering with four trial points (FDE-4). All algorithms are tested with a similar setup, i.e. using the same population size, control parameters, and trial point generation strategy using test problems from the literature.

We have selected six test problems with varying levels of complexity from the literature, and used their implementations given in [22]. Although our aim is to provide an algorithm suitable for solving real-life engineering problems, for performance testing we used only analytical test problems, as results of these are easier to compare, and they possess certain features that test the different abilities of optimization algorithms.

The Rosenbrock function has its global minimum in a deep, curved valley, the Michalewicz function contains constant valued grooves with differing orientation, and one rather deep global minimum, and the Rastrigin and Griewangk functions contain both numerous local minima in the otherwise bowl-shaped surface. The Ackley function is essentially flat, but with a wrinkled surface, and the global optimum is located in a very deep valley in the middle, whereas the Levy function contains grooves and barriers around a rather flat area around the global minimum. Formulations, ranges of search spaces and optimal objective function and design variable values for the test functions are given in Table 1.

In all algorithms, we used two different trial point generation strategies, namely *DE/rand/1* (denoted by ODE1, DE1, FDE1, FDE1-4, and strategy=1 in figures) and *DE/local-to-best/1* (denoted by ODE2, DE2, FDE2, FDE2-4 , and strategy=2 in figures). See [31, 36, 37] for details and comparisons of different point generation strategies. In this study the scale factor F was set to 0.8, crossover probability CR to 0.1, and the population size to $11n$, n standing for the number of variables. No attempt was made to tune these parameters for a better performance. For each of the six test functions, versions with 2, 5 and 10 design variables were tested, using a maximum budget of 500, 1000, and 2000 evaluations, respectively. The best results gained at 10 different evaluation levels (e.g. in 2D case results after 50, 100, 150, etc. evaluations) were logged for each of the algorithms for 100 test runs using the same initial populations.

Using this test setup it is possible to compare the differences of various implementations of the same algorithm, as well as the differences of the same implementation using different trial point generation strategies. Further, comparison of all different algorithms is possible head to head. The averaged results for each of the test functions are shown in Figures 2, 3, 4, 5, 6 and 7. In each of the figures in the left column are the results of all algorithms using the strategy *DE/rand/1*, and at the right column using the strategy *DE/local-to-best/1*. In both columns, results are given for 2, 5, and 10 design variable versions of the test functions.

In Tables 2 and 3 we show statistics of the best results for all the test runs after the evaluation budget has exhausted. This data includes mean, minimum and maximum objective values, and also standard deviation (Std) and variance (Var) of the values.

Table 1: Test problems used in this study.

Function	Formulation
Rosenbrock	$f(x) = \sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$
Range	$-5.12 \leq x_i \leq 5.12$
Optimum	$x^* = (1, ..., 1), f(x^*) = 0$
Michalewicz	$f(x) = -\sum_{i=1}^{n} sin(x_i)(sin(ix_i^2/\pi))^{2m}; m = 10$
Range	$0 \leq x_i \leq \pi$
Optimum	$n = 2, f(x^*) = -1.8013; n = 5, f(x^*) = -4.687; n = 2, f(x^*) = -9.66$
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
Range	$-5.12 \leq x_i \leq 5.12$
Optimum	$x^* = (0, ..., 0), f(x^*) = 0$
Griewangk	$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(x_i/\sqrt{i}) + 1$
Range	$-600 \leq x_i \leq 600$
Optimum	$x^* = (0, ..., 0), f(x^*) = 0$
Ackley	$f(x) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{-\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)}$
Range	$-32.768 \leq x_i \leq 32.768$
Optimum	$x^* = (0, ..., 0), f(x^*) = 0$
Levy	$f(x) = sin^2(\pi y_1) + \sum_{i=1}^{n}[(y_i - 1)^2(1 + 10sin^2(\pi y_i + 1))]$ $+(y_n - 1)^2(1 + 10sin^2(2\pi y_n)), y_i = 1 + \frac{x_i - 1}{4}, i = 1, ...n.$
Range	$-10 \leq x_i \leq 10$
Optimum	$x^* = (1, ..., 1), f(x^*) = 0$

Figure 2: Evaluation histories of Rosenbrock function with 2, 5 and 10 dimensions.

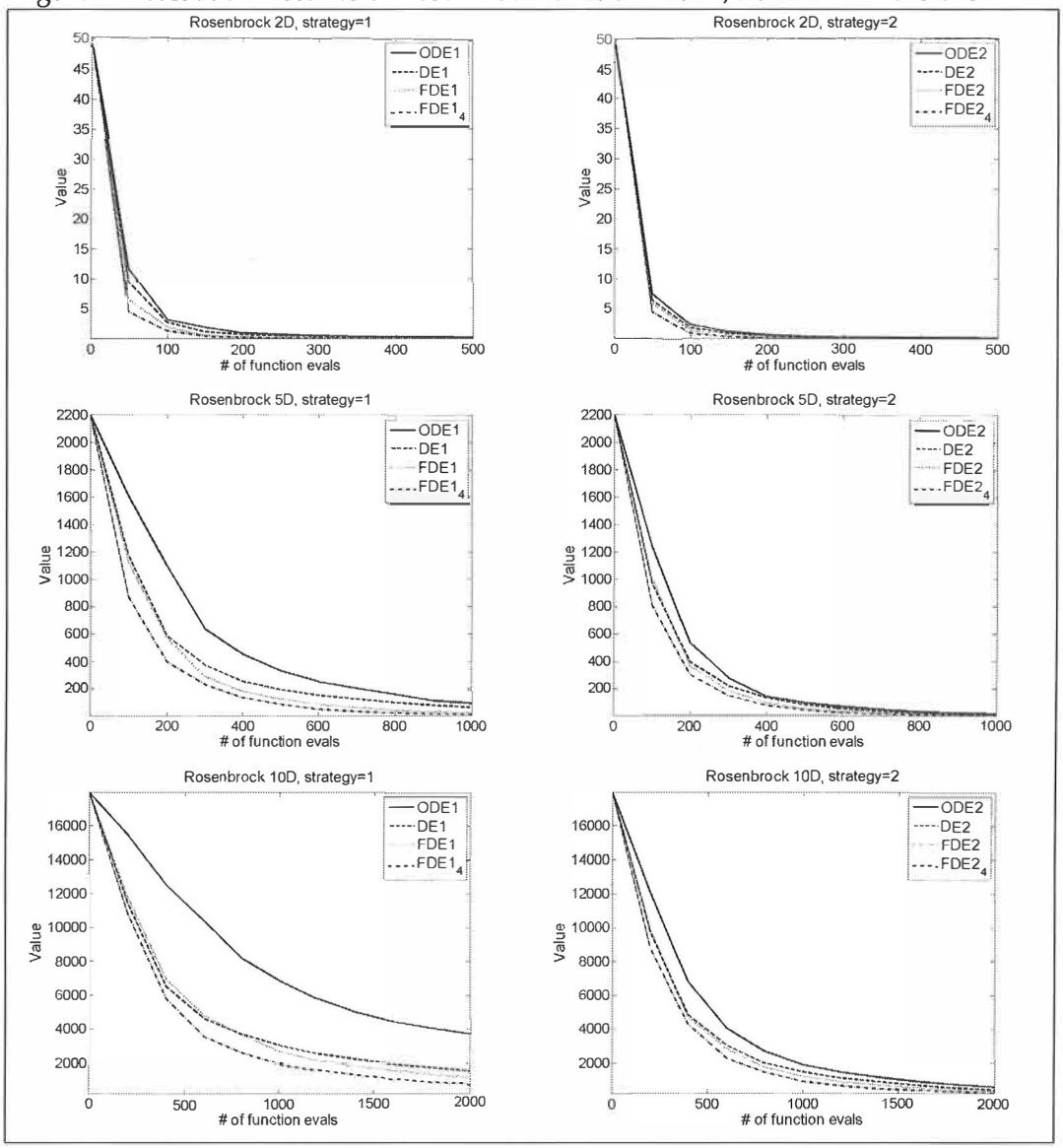


Figure 3: Evaluation histories of Michalewicz function with 2, 5 and 10 dimensions.

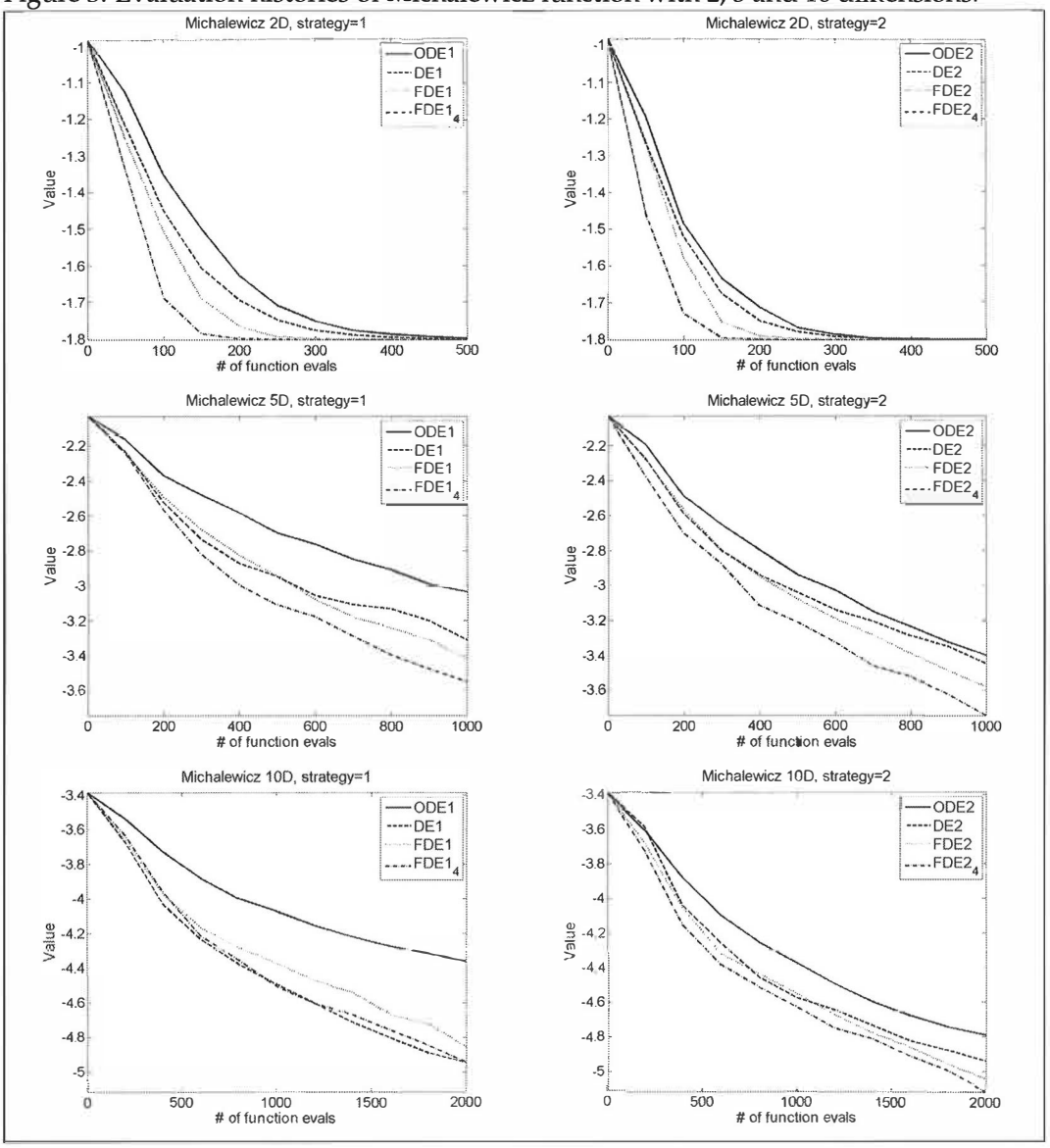


Figure 4: Evaluation histories of Rastrigin function with 2, 5 and 10 dimensions.

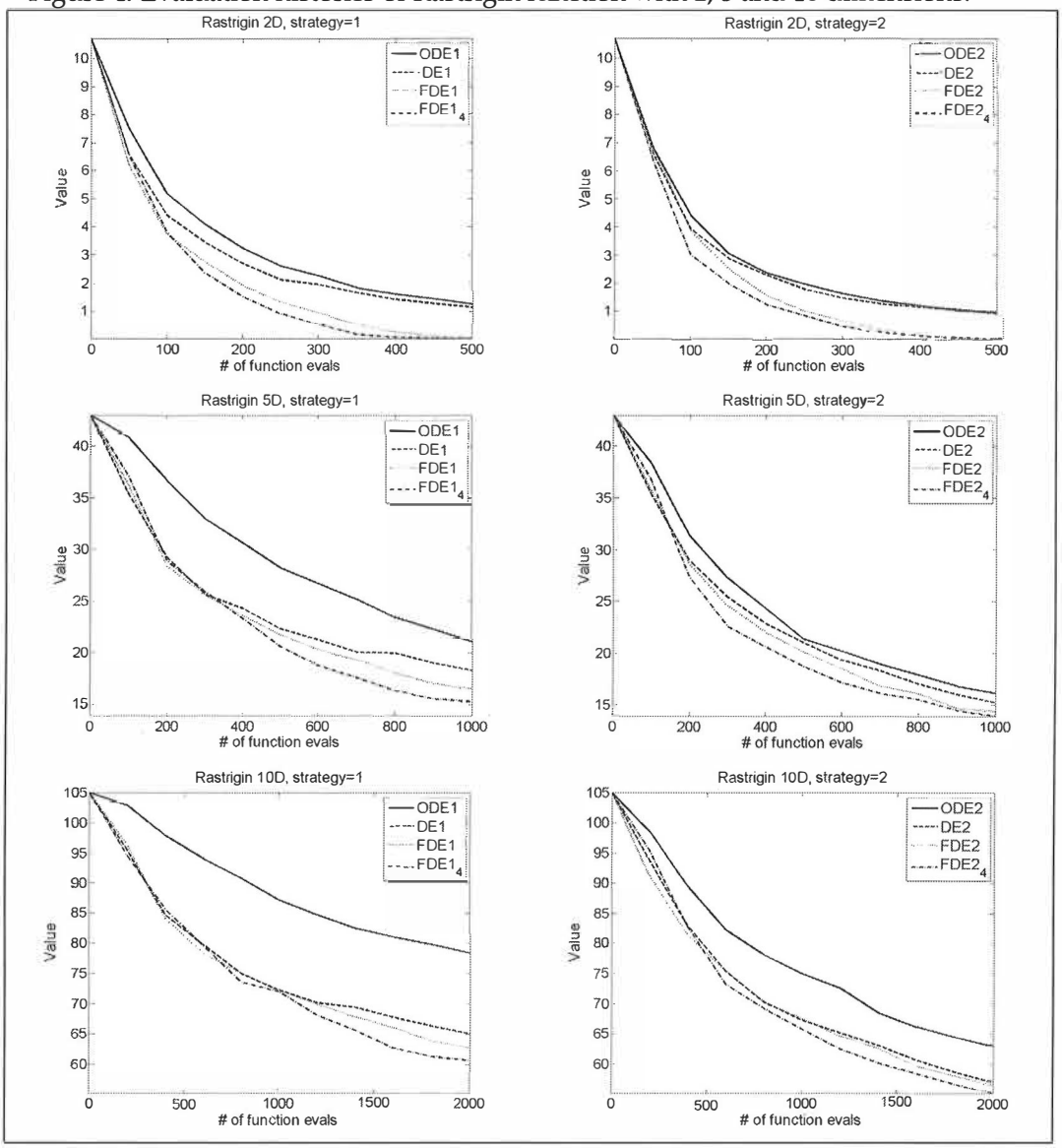


Figure 5: Evaluation histories of Griewangk function with 2, 5 and 10 dimensions.

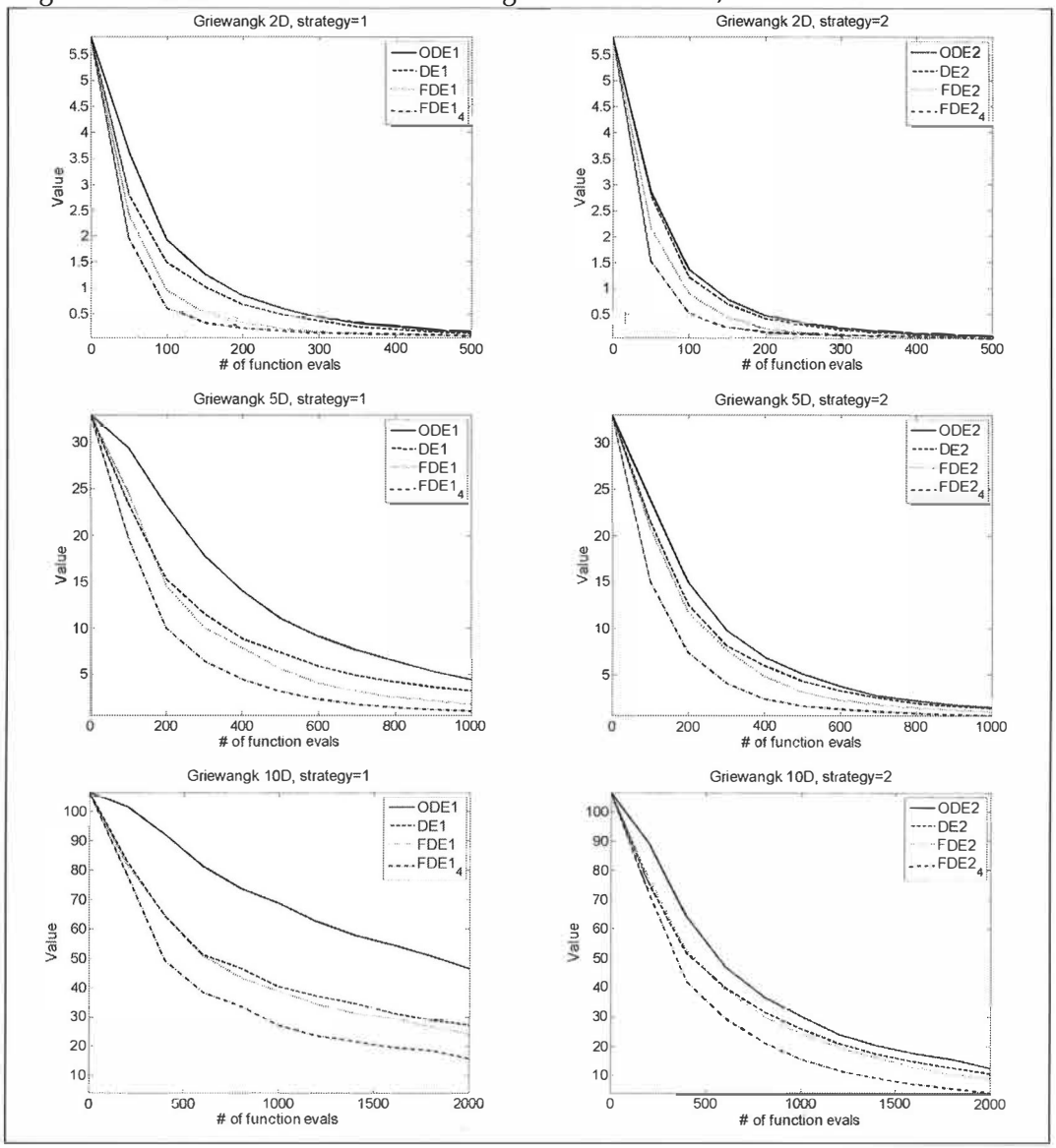


Figure 6: Evaluation histories of Ackley function with 2, 5 and 10 dimensions.

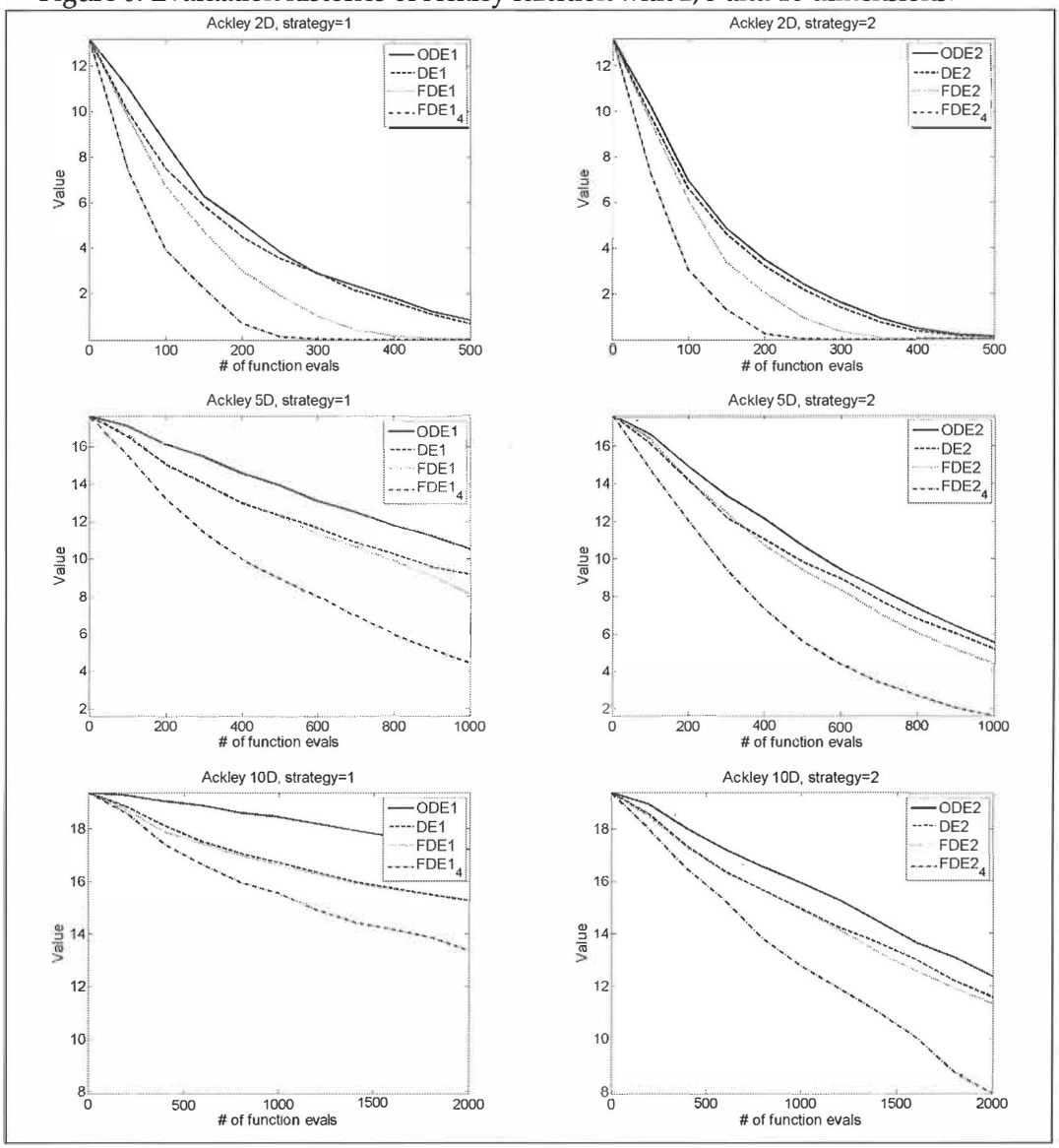


Figure 7: Evaluation histories of Levy function with 2, 5 and 10 dimensions.

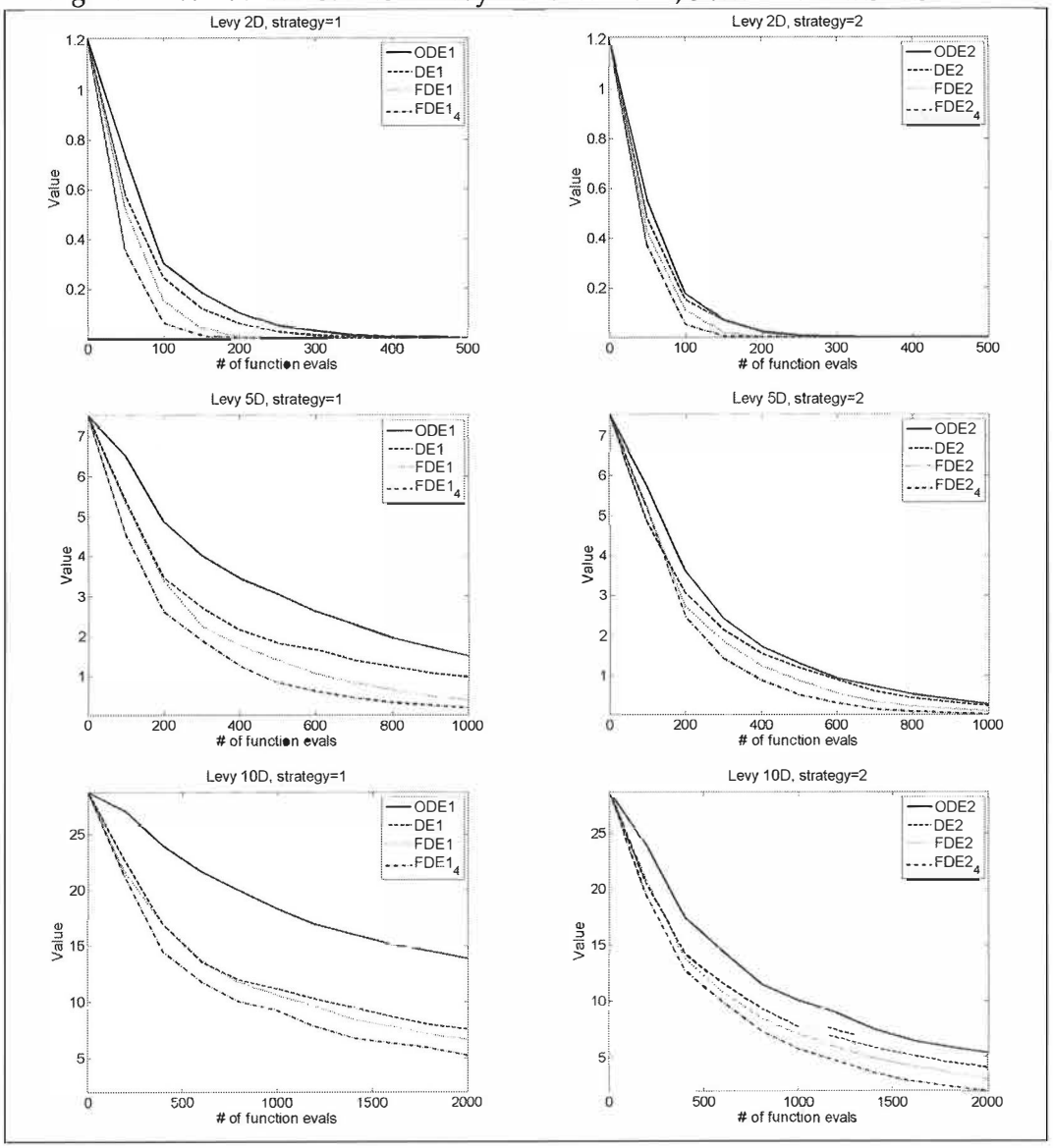


Table 2: Statistics of objective function values of Rosenbrock, Michalewicz and Rastrigin test functions.

Alg.	Dim.	Mean	Min	Max	Std	Var
Rosenbrock						
ODE1	2D	0.13933	0.0011829	1.2031	0.2064	0.042599
DE1		0.10267	0.0002514	0.71448	0.12992	0.016879
FDE1		0.0051592	6.67e-07	0.30827	0.031333	0.00098175
FDE1-4		0.00022297	2.86e-07	0.0050053	0.00070582	4.98e-07
ODE2		0.049814	4.40e-06	0.81035	0.11481	0.013181
DE2		0.020344	7.79e-06	0.43632	0.050369	0.002537
FDE2		0.00010014	1.25e-08	0.0086671	0.00086564	7.49e-07
FDE2-4		9.37e-06	1.68e-08	9.50e-05	1.55e-05	2.40e-10
ODE1	5D	95.152	9.817	265.83	50.361	2536.3
DE1		65.052	5.8807	200.51	37.508	1406.9
FDE1		24.379	6.8257	69.179	12.075	145.8
FDE1-4		12.459	3.7832	31.294	5.9154	34.992
ODE2		16.916	1.8072	49.011	8.3409	69.571
DE2		12.952	3.0554	31.746	5.6007	31.368
FDE2		7.5358	2.3445	18.301	2.9544	8.7285
FDE2-4		4.5609	2.003	8.2543	1.4095	1.9866
ODE1	10D	3688.3	928.24	8431.2	1631	2.66e+06
DE1		1537.7	318.97	3009.6	633.35	4.01e+05
FDE1		1121.2	214.47	2704.7	498.67	2.49e+05
FDE1-4		749.75	194.3	1537.4	328.8	1.08e+05
ODE2		571.97	88.942	1627.2	262.4	68853
DE2		395.54	105.03	1082.5	164.23	26973
FDE2		300.59	90.228	638.16	116.94	13674
FDE2-4		209.26	95.84	382.31	70.654	4992
Michalewicz						
ODE1	2D	-1.7975	-1.8013	-1.7714	0.0050179	2.52e-05
DE1		-1.8001	-1.8013	-1.7932	0.0014807	2.19e-06
FDE1		-1.8013	-1.8013	-1.8012	1.15e-05	1.32e-10
FDE1-4		-1.8013	-1.8013	-1.8013	4.02e-09	1.62e-17
ODE2		-1.8009	-1.8013	-1.7968	0.00081231	6.60e-07
DE2		-1.8011	-1.8013	-1.7982	0.00041874	1.75e-07
FDE2		-1.8013	-1.8013	-1.8013	4.76e-07	2.26e-13
FDE2-4		-1.8013	-1.8013	-1.8013	3.07e-08	9.41e-16
ODE1	5D	-3.0357	-4.0281	-2.5076	0.29944	0.089663
DE1		-3.3112	-4.3251	-2.6711	0.33934	0.11515
FDE1		-3.4151	-4.3432	-2.7847	0.26525	0.070358
FDE1-4		-3.5505	-4.1775	-2.9042	0.26236	0.068835
ODE2		-3.399	-4.1911	-2.7683	0.34017	0.11572
DE2		-3.4459	-4.2158	-2.7565	0.28644	0.082046
FDE2		-3.579	-4.2791	-2.6737	0.32099	0.10303
FDE2-4		-3.7426	-4.597	-3.0953	0.28832	0.08313
ODE1	10D	-4.3632	-5.8381	-3.643	0.39037	0.15239
DE1		-4.945	-6.4729	-4.1635	0.44341	0.19661
FDE1		-4.8562	-6.1639	-4.175	0.38355	0.14711
FDE1-4		-4.9425	-6.2233	-4.1579	0.47112	0.22195
ODE2		-4.7897	-5.8447	-4.0173	0.36469	0.133
DE2		-4.9409	-5.9533	-4.3711	0.35175	0.12373
FDE2		-5.0466	-6.3121	-4.2761	0.39339	0.15476
FDE2-4		-5.1154	-6.3622	-4.3308	0.39613	0.15692
Rastrigin						
ODE1	2D	1.2652	0.039037	3.6711	0.69544	0.48363
DE1		1.1558	0.025507	3.7253	0.78223	0.61188
FDE1		0.04337	8.88e-06	1.0009	0.17217	0.029643
FDE1-4		0.017031	2.95e-06	0.99533	0.12844	0.016497
ODE2		0.96201	0.015212	2.3392	0.6412	0.41114
DE2		0.92531	0.0048362	3.3515	0.6955	0.48372
FDE2		0.044176	4.00e-06	0.99567	0.19528	0.038133
FDE2-4		0.036352	4.99e-07	0.99508	0.17964	0.032269
ODE1	5D	21.085	5.5753	32.581	5.0979	25.989
DE1		18.308	8.2619	27.726	4.4469	19.775
FDE1		16.501	7.4347	23.906	3.61	13.032
FDE1-4		15.274	6.9825	25.754	4.4782	20.054
ODE2		16.144	4.9601	25.534	4.5482	20.686
DE2		15.242	4.8189	25.32	4.0509	16.41
FDE2		14.379	4.6423	23.716	3.9838	15.871
FDE2-4		13.924	4.3374	20.29	4.3467	18.894
ODE1	10D	78.442	55.564	102.09	8.5632	73.328
DE1		65.048	37.947	78.627	7.3014	53.311
FDE1		62.731	42.552	78.608	7.9112	62.587
FDE1-4		60.68	42.776	75.687	7.5012	56.267
ODE2		63.066	36.655	77.442	9.1629	83.959
DE2		57.051	36.888	73.259	7.9219	62.757
FDE2		56.529	38.047	69.551	7.8482	61.595
FDE2-4		55.194	39.882	69.159	6.5264	42.594

Table 3: Statistics of objective function values of Griewangk, Ackley and Levy test functions.

Alg.	Dim.	Mean	Min	Max	Std	Var
Griewangk						
ODE1	2D	0.15828	0.015902	0.62475	0.098673	0.0097363
DE1		0.12045	0.010989	0.37088	0.070397	0.0049558
FDE1		0.077168	0.00051745	0.21885	0.048762	0.0023777
FDE1-4		0.078194	0.0079527	0.24222	0.051369	0.0026388
ODE2		0.089289	0.015299	0.36469	0.060016	0.0036019
DE2		0.086146	0.0027124	0.26114	0.053509	0.0028632
FDE2		0.047309	0.0027585	0.18224	0.032163	0.0010345
FDE2-4		0.044792	0.0074184	0.14827	0.028018	0.00078499
ODE1	5D	4.4451	1.2716	11.375	1.8272	3.3387
DE1		3.2407	1.1328	7.8422	1.1855	1.4054
FDE1		1.7922	0.89638	2.8796	0.39925	0.1594
FDE1-4		1.053	0.47794	1.3756	0.19257	0.037082
ODE2		1.4537	0.70289	2.2839	0.32296	0.1043
DE2		1.3769	0.77265	2.069	0.29379	0.086315
FDE2		0.99204	0.44286	1.5092	0.2015	0.040603
FDE2-4		0.60009	0.24967	1.0213	0.15221	0.023167
ODE1	10D	46.486	15.592	73.724	9.7731	95.513
DE1		27.202	11.394	43.745	6.4303	41.349
FDE1		24.071	10.687	43.423	5.8795	34.568
FDE1-4		15.86	8.0157	24.205	4.0717	16.579
ODE2		12.493	4.7809	22.074	3.2822	10.773
DE2		10.634	5.7312	21.011	3.0984	9.6003
FDE2		8.9361	3.9286	16.064	2.4632	6.0674
FDE2-4		4.0786	2.3326	7.8508	1.1273	1.2709
Ackley						
ODE1	2D	0.87119	0.029443	2.6731	0.63714	0.40594
DE1		0.71839	0.028378	2.503	0.5597	0.31327
FDE1		0.022522	0.0006528	0.25714	0.029161	0.00085039
FDE1-4		0.00018045	1.68e-05	0.00052317	0.00013568	1.84e-08
ODE2		0.11096	0.0058637	1.3049	0.14731	0.0217
DE2		0.082308	0.0020054	0.47114	0.078768	0.0062045
FDE2		0.0038188	0.00039338	0.046296	0.0053189	2.83e-05
FDE2-4		1.57e-05	1.55e-06	7.58e-05	1.48e-05	2.19e-10
ODE1	5D	10.534	5.2078	14.467	1.9167	3.6739
DE1		9.1868	4.2638	12.735	1.5805	2.498
FDE1		8.1576	3.5055	12.932	1.6107	2.5944
FDE1-4		4.428	2.8381	6.6601	0.80722	0.6516
ODE2		5.5387	3.0095	7.9144	1.0396	1.0807
DE2		5.1995	3.3695	7.7681	0.9162	0.83942
FDE2		4.3941	2.3573	5.7381	0.70847	0.50194
FDE2-4		1.6077	0.65515	2.5346	0.48062	0.23099
ODE1	10D	17.201	13.78	18.77	0.88176	0.7775
DE1		15.269	11.965	17.082	1.0199	1.0403
FDE1		15.292	12.276	17.165	0.98665	0.97347
FDE1-4		13.365	11.587	15.19	0.82879	0.6869
ODE2		12.37	7.8045	14.73	1.1944	1.4266
DE2		11.581	8.2956	13.901	1.173	1.376
FDE2		11.327	8.583	14.052	1.0974	1.2043
FDE2-4		7.9283	5.3189	9.816	1.0149	1.03
Levy						
ODE1	2D	0.0010884	1.91e-06	0.010404	0.00159	2.53e-06
DE1		0.00058036	1.86e-06	0.012404	0.0013374	1.79e-06
FDE1		9.20e-07	2.26e-09	6.35e-06	1.22e-06	1.50e-12
FDE1-4		1.41e-08	5.31e-11	1.50e-07	2.95e-08	8.70e-16
ODE2		2.14e-05	2.60e-08	0.00024394	3.24e-05	1.05e-09
DE2		1.13e-05	5.18e-08	7.04e-05	1.61e-05	2.58e-10
FDE2		1.00e-07	6.46e-10	9.09e-07	1.52e-07	2.30e-14
FDE2-4		6.48e-10	6.63e-12	1.27e-08	1.67e-09	2.79e-18
ODE1	5D	1.5163	0.42048	3.4471	0.68766	0.47288
DE1		0.99509	0.2359	2.2107	0.42471	0.18038
FDE1		0.39701	0.091409	0.98924	0.19043	0.036263
FDE1-4		0.20058	0.057588	0.50282	0.076989	0.0059274
ODE2		0.28163	0.022184	0.86838	0.14988	0.022463
DE2		0.24506	0.018783	0.67668	0.13881	0.019268
FDE2		0.11329	0.01355	0.26834	0.050271	0.0025272
FDE2-4		0.030253	0.0051786	0.096573	0.018185	0.00033069
ODE1	10D	13.866	4.5205	21.458	3.5869	12.866
DE1		7.5849	3.3281	12.802	2.1542	4.6406
FDE1		6.6402	3.0326	10.36	1.7888	3.1999
FDE1-4		5.264	2.4893	8.3384	1.3582	1.8448
ODE2		5.3854	1.6434	12.87	1.8516	3.4285
DE2		4.0206	1.8946	8.3759	1.1549	1.3339
FDE2		3.0407	1.1679	6.0894	0.85864	0.73727
FDE2-4		2.0024	0.73066	3.7209	0.6377	0.40666

From the results we can first notice that in several cases there is a big difference in performances of ODE and DE implementations, although they represent basically the same algorithm. There are at least two possible factors explaining the performance difference. First is how the box constrained search space is handled. In ODE, trial points which are not inside the box are bounced i.e. reflected back, whereas in DE these points are neglected and new points are generated until one is found which is inside the given box. Another difference is that in ODE, the random selection of vectors in trial point generation is performed by shuffling the population array (to reduce computational overhead of the algorithm), and hence a certain vector cannot be chosen twice in the same term of the perturbation expression. This obviously cuts down the number of possible combinations, and it may be detrimental to the performance (with regard to objective function evaluations needed) of the algorithm.

Another observation is that in some cases the performance difference of FDE over DE is only a minor one, suggesting that the filtered approach with only a single trial point does not offer remarkable advantage. In a sense this is very natural, because the only decision to make is whether to evaluate the trial point or not, and as our approach to predict the objective function value is rather crude, no remarkable advantage is gained.

On the other hand, FDE-4 with only four trial points (which must be considered as a very low number if we are dealing with computationally expensive objective function) for each parent produced in most cases clearly better performance than DE and FDE approaches. Obviously this is due to the fact that when compared to its single point counterpart, in some sense, FDE-4 gathers four times more information about the search space via the surrogate, and this approach seems to be more effective.

From the results we can make the general remark that the comparable efficiency of the FDE variants seems to be inversely proportional to the efficiency of the chosen point generation strategy. This is probably due to the fact that the point generation strategy *DE/local-to-best/1* seems to be far more efficient in itself, and there are not so many points which can be readily judged inefficient by the proposed filtering method. However, in all the cases (except the Michalewicz, 10D, with strategy=1) the proposed algorithm is more efficient than the respective DE version without filtering.

With respect to the problem dimensions, it seems that the proposed approach, especially with the FDE-4 variant maintains it efficiency with increasing dimensions. Anyhow, with a higher number of dimensions than used in this study, it is possible that at some stage the proposed algorithm starts to lose its efficiency. This is due to the fact that the concept of nearest neighbor may not be meaningful in some cases in high dimensions [6], and thus also the nearest neighbor interpolation may not work efficiently.

In every example case, the difference between the least (ODE) and the most (FDE-4) efficient algorithm is notable. In some cases, the difference between the non-filtered and the filtered approach is rather small, but in most of the cases the use

of more trial points in the filtered approach produces remarkable performance advantage. A strange anomaly is the Michalewicz function in 10D, with strategy=1, where the non-filtered approach seems to be more efficient than either of the filtered approaches.

5 Discussion and conclusions

In this work, we have discussed some drawbacks of the widely used global optimization approaches. As our emphasis is to provide tools to solve some real engineering problems where the number of affordable objective function evaluations is somewhat restricted, we have proposed a new algorithm which seems to be computationally rather efficient and complements previous approaches by overcoming some of the drawbacks discussed. Namely, the proposed approach follows the sublevel set framework by generating set of trial points, and filtering presumably inefficient ones away using some ideas borrowed from meta-modelling approaches. Yet, our approach does not require any model management procedure, and there is no explicit surrogate fitting, which may be a difficult or time consuming task in itself. Also, computational cost required to predict the objective function value of the trial point is negligible (compared to that of the expensive objective function evaluation), as it requires computation of a point-to-point distance matrix of the population once in each generation, and computation of a distance vector from the trial point to all evaluated points for every trial point. Further, the proposed algorithm is very straightforward to implement, and the proposed filtering mechanism could probably be adapted to enhance the performance of several different population based algorithms (for example, GA or PSO).

The computational efficiency of the new algorithm is explained by the fact that some of the trial points are excluded from the expensive objective function evaluation, based on the information contained in the previously sampled points. In this way, based on the numerical tests conducted in this study, savings up to 80% in the number of objective function evaluations, compared to that of commonly available implementation of DE (ODE), could be realized.

The performance comparisons presented in this study encourage us to believe that our approach has some real potential, but there is still need for further study. For example, a more thorough testing is needed to find out how the performance of the proposed algorithm is related to the number of trial points that are generated for each parent. Further, it would be useful to be able to provide guideline or even self-adaptivity for the algorithm itself to balance between the cost of the objective function, and a reasonable number of trial points.

In our approach, there is also room for other types of improvement. For example, a more accurate interpolation could be implemented, accompanied by different means to estimate the prediction error, and the results of both these could be verified by some sort of a cross validation scheme, i.e., against the objective function value of some other neighboring point. Conversely to the current approach, more intelligent trial point generation strategies should be developed (to follow the sublevel set

framework), as this would decrease the need for filtering the trial points in general.

The proposed algorithm has been designed to suit best for the problems where the expense of the objective function evaluations is mediocre, i.e., one objective function evaluation takes from a few seconds to some minutes. Inexpensive problems can be easily solved using known EA's, for example, Differential Evolution. On the other hand, very expensive problems, where one evaluation may take several hours, benefit most of more sophisticated (and also more complex) utility function based surrogate approaches, such as EGO, where all sampled information is exploited to its full extent.

It is reasonable to assume, but open to further research, that with the problems of mediocre computational cost the efficiency of our approach can be adjusted by using suitable number of trial points for each parent. In this manner, the computational overhead of our algorithm can be balanced with the cost of objective function evaluation, thus minimizing the wall clock time needed to solve a certain problem.

6 Acknowledgements

The author would like to thank Professor Kaisa Miettinen for support, insights, perspectives and proof reading during the research and preparation of this article. Further, the author would like to thank Mr. Sauli Ruuska, Dr. Jussi Hakanen, Dr. Sami Äyrämö and Dr. Ferrante Neri for some advice and valuable discussions.

References

[1] T. Aittokoski (2007): On Optimization of Simulation Based Design. Licentiate Thesis. Jyväskylä Licentiate Thesis in Computing 8. University of Jyväskylä.

[2] T. Aittokoski and K. Miettinen (2008): Cost Effective Simulation-Based Multiobjective Optimization in Performance of Internal Combustion Engine. Engineering Optimization 40(7), 593-612.

[3] P. J. Angeline (1998): Evolutionary Optimization versus Particle Swarm Optimization. In V. W. Porto, N. Saravanan, D. Waagen, A. E. Eiben (eds.) Evolutionary programming VII, 601-610. Springer, Berlin.

[4] B. V. Babu and R. Angira, Optimization of Water Pumping System Using Differential Evolution Strategies. In Proceedings of The Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS-2003), Singapore, 2003.

[5] J.- F. M. Barthelemy and R. T. Haftka (1993): Approximation concepts for optimum structural design - a review. Structural Optimization 5, 129-144.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft (1999): When Is "Nearest Neighbor" Meaningful? In Proceedings of International Conference on Database Theory, 217-235.

[7] M. D. Buhmann (2003): Radial Basis Functions. Cambridge University Press, Cambridge.

[8] L. Bull (1999): On model-based evolutionary computation. Soft Computing 3, 76-82.

[9] C. Cortes and V. Vapnik (1995): Support vector networks. Machine Learning 20, 273-297.

[10] N. Cressie (1990): The origins of kriging. Mathematical Geology 22, 197-202.

[11] D. Dasgupta and Z. Michalewicz (Eds.) (1997): Evolutionary Algorithms in Engineering Applications. Springer, Berlin.

[12] J. E. Dennis and V. Torczon (1997): Managing approximation models in optimization. In N. Alexandrov and M. Y. Hussaini, eds., Multidisciplinary Design Optimization: State of the Art, 330-347. Society for Industrial & Applied Mathematics.

[13] A. E. Eiben and C. A. Schippers (1998): On Evolutionary Exploration and Exploitation. Fundamenta Informaticae 35(1-4), 35-50.

[14] M. A. El-Beltagy, P. B. Nair and A. J. Keane (1999): Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In Proceedings of Genetic and Evolutionary Conference, 196-203, Orlando, 1999.

[15] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck and K. Giannakoglou (2002): Metamodel-assisted evolution strategies. In Parallel Problem Solving from Nature, number 2439 in Lecture Notes in Computer Science, 371-380. Springer, Berlin.

[16] V. Feoktistov (2006): Differential Evolution - In Search of Solutions. Springer, Berlin.

[17] K. C. Giannakoglou, M. K. Karakasis and I. C. Kampolis (2006): Evolutionary Algorithms with Surrogate Modeling for Computationally Expensive Optimization Problems. ERCOFTAC 2006 Design Optimization International Conference, April 5-7 2006, Gran Canaria, Spain.

[18] A. P. Giotis, M. Emmerich, B. Naujoks, K. C. Giannakoglou and T. Bäck (2002): Low-cost stochastic optimization for engineering applications. Evolutionary Methods for Design, Optimisation and Control, K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou and T. Fogarty (Eds.).

[19] H. -M. Gutmann (2001): A Radial Basis Function Method for Global Optimization. Journal of Global Optimization 19, 201-227.

[20] P. Hansen, B. Jaumard and S. H. Lu (1992): On Using Estimates of Lipschitz Constants in Global Optimization. Journal of Optimization Theory and Applications 75(1), 195-200.

[21] Simon Haykin (1998): Neural Networks: A Comprehensive Foundation, 2nd Ed. Prentice Hall, New Jersey.

[22] A. Hedar (2008): Test Functions for Unconstrained Global Optimization. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

[23] Y. Jin (2005): A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. Soft Computing 9(1), 3-12.

[24] Y. Jin, M. Olhofer and B. Sendhoff (2001): Managing approximate models in evolutionary aerodynamic design optimization. In Proceedings of IEEE Congress on Evolutionary Computation 1, 592-599, May 2001.

[25] D. R. Jones, M.Schonlau and W. J. Welch (1998): Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13(4), 455-492.

[26] H. J. Kushner (1964): A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering 86, 97-106.

[27] N. Muttil and S.-Y. Liong (2004): Superior Exploration-Exploitation Balance in Shuffled Complex Evolution. Journal of Hydraulic Engineering 130(12), 1202-1205.

[28] H. Nakayama, M. Arakawa and R. Sasaki (2002): Simulation-Based Optimization Using Computational Intelligence. Optimization and Engineering 3, 201-214.

[29] B. Naudts and A. Schippers (1999): A motivated definition of exploitation and exploration. Technical report 02-99, University of Antwerp, Belgium.

[30] Y. S. Ong, P. B. Nair, A. J. Keane and K. W. Wong (2004): Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin (ed.), Knowledge Incorporation in Evolutionary Computation, 307-332. Springer, Berlin.

[31] K. V. Price, R. M. Storn and J. A. Lampinen (2005): Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin.

[32] A. Ratle (1998): Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer and H.-P. Schwefel, eds., Parallel Problem Solving from Nature, volume V, 87-96. Springer, Berlin.

[33] A. Ratle (1999): Optimal sampling strategies for learning a fitness model. In Proceedings of 1999 Congress on Evolutionary Computation, 3, 2078-2085, Washington D.C., July 1999.

[34] S. Ruuska and T. Aittokoski (2008): The Effect of Trial Point Generation Schemes on the Efficiency of Population-Based Global Optimization Algorithms. In Proceedings of International Conference on Engineering Optimization, Rio de Janeiro, Brazil, 2008.

[35] M. J. Sasena (2002): Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Doctoral Thesis. University of Michigan.

[36] R. Storn (2008): Differential Evolution Homepage.
`http://www.icsi.berkeley.edu/~storn/code.html`

[37] R. Storn and K. Price (1995): Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. ICSI Technical Report tr-95-012.

[38] R. Storn and K. Price (1997): Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341-359.

[39] Z. Tu and Y. Lu (2004): A robust stochastic genetic algorithm for global numerical optimization. IEEE Transactions on Evolutionary Computation 8(5), 456-470.

[40] V. N. Vapnik (1998): Statistical Learning Theory. John Wiley & Sons, New York.

[41] J. Vesterstrom and R. Thomsen (2004): A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proceedings of the 2004 Congress on Evolutionary Computing 2, 1980-1987.

[42] G. R. Wood and B. P. Zhang (1996): Estimation of the Lipschitz Constant of a Function. Journal of Global Optimization 8, 91-103.

[43] X. Yao and Y. Liu (1997): Fast Evolution Strategies. In P. J. Angeline, R. J. Reynolds, J. R. McDonnell, R. Eberhart (eds.), Evolutionary programming VI, 151-161. Springer, Berlin.

[44] X. Yao, Y. Liu and G. M. Lin (1999): Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3, 82-102.

[45] A. Zilinskas (1985): Axiomatic Characterization of a Global Optimization Algorithm and Investigation of its Search Strategy. Operations Research Letters 4(1), 35-39.