Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization

Karin Zielinski and Rainer Laur

Institute for Electromagnetic Theory and Microelectronics, University of Bremen,
P.O. Box 330440, 28334 Bremen, Germany
{zielinski,rlaur}@item.uni-bremen.de

Summary. Because real-world problems generally include computationally expensive objective and constraint functions, an optimization run should be terminated as soon as convergence to the optimum has been obtained. However, detection of this condition is not a trivial task. Because the global optimum is usually unknown, distance measures cannot be applied for this purpose. Stopping after a predefined number of function evaluations has not only the disadvantage that trial-and-error methods have to be applied for determining a suitable number of function evaluations, but the number of function evaluations at which convergence occurs may also be subject to large fluctuations due to the randomness involved in evolutionary algorithms. Therefore, stopping criteria should be applied which react adaptively to the state of the optimization run. In this work several stopping criteria are introduced that consider the improvement, movement or distribution of population members to derive a suitable time for terminating the Differential Evolution algorithm. Their application for other evolutionary algorithms is also discussed. Based on an extensive test set the criteria are evaluated using Differential Evolution, and it is shown that a distribution-based criterion considering objective space yields the best results concerning the convergence rate as well as the additional computational effort.

1 Introduction

Since the development of Differential Evolution (DE) in 1995 [1], considerable effort has been spend to improve its convergence characteristics e.g. by varying operators [2, 3] or changing the handling of constraints [4, 5]. As a consequence, several enhancements have been found during the last years that have led to successful applications in many different fields [6]. However, even the performance of a very good algorithm may be bad for practical purposes when it is not stopped at a proper time. For theoretical work about convergence properties or a comparison of different implementations of DE this aspect is generally not important because for this purpose usually test functions are employed for which the optimum is known. In that case, the execution of the algorithm can be terminated if the optimum is found with a given accuracy, and the involved computational effort can be used to analyze the performance of different DE implementations. An alternative is to terminate after a defined number of function evaluations (FEs) and to evaluate the distance of the best individual to the

U.K. Chakraborty (Ed.): Advances in Differential Evolution, SCI 143, pp. 111–138, 2008.
springerlink.com

optimum. This approach works well for theoretical work when algorithm variants are tested against each other but for real-world problems the situation is different because the optimum is usually unknown.

A stopping rule for problems with unknown optimum that is widely used in the literature is to terminate the execution of an algorithm after a given maximum number of function evaluations FE_{max} (this stopping criterion will be called *LimFuncEval* in the following). This approach is associated with two problems: Suitable settings for FE_{max} vary considerably for different optimization problems, so usually FE_{max} has to be figured out by trial-and-error methods. A further problem is that the number of objective function evaluations FE_{conv} that is needed for convergence for one specific optimization problem may also be subject to large variations due to the stochastic nature of DE. This statement holds for many different implementations of DE as can be seen in [3, 4, 5, 7, 8, 9, 10]. Because real-world problems usually contain computationally expensive objective and constraint functions it is imperative that unnecessary function evaluations are avoided. Therefore, it is important to examine other alternatives for stopping the execution of the DE algorithm besides termination after a fixed number of function evaluations. In order to deal with the problem that is caused by fluctuations of FE_{conv}, the stopping criteria have to be able to detect when convergence is reached. Thus, they have to react adaptively to the current state of an optimization run. The stopping criteria have to ensure that the algorithm is executed long enough to obtain convergence to the global optimum but without wasting of computational resources.

Different mechanisms can be used for deriving conclusions about the current state of an optimization run. In principle any phenomenon can be used that exhibits a definite trend from the beginning to the end of an optimization run. For instance both the improvement as well as the movement of individuals are typically large in the beginning of an optimization run and both become small when approaching convergence. Another example is the distribution of population members as they are scattered throughout the search space initially but usually converge to one point towards the end of an optimization run. Consequently, each of these properties is basically usable for detecting convergence.

Any of the before-mentioned population characteristics like improvement, movement and distribution can be used in various implementations for the creation of stopping conditions. Because the performance of different implementations is not necessarily similar, an extensive study analyzing their abilities will be presented in this chapter. Conclusions will be derived about which mechanisms are best suited to meet the demands of reliable stopping after convergence to the optimum has been obtained without wasting of computational resources.

Besides the problem with fluctuations of FE_{conv}, terminating after a fixed number of function evaluations is also connected with the problem that a suitable setting for parameter FE_{max} has to be found. Apparently, this kind of problem is inherent to all stopping criteria because up to now no parameter-free stopping criterion is known. The adaptive stopping criteria which will be presented in this chapter are also associated with parameters which have to be chosen by the

user. Interestingly, it will be shown in this work that standard settings which can be used for a large range of optimization problems exist for some of them. As a consequence, the application of these stopping criteria is easy for the user.

One problem in the field of optimization is that authors often use different sets of test functions or different accuracies (for defining convergence to the optimum as well as for the allowed constraint violation of equality constraints). Hence, it is generally difficult to compare results. In contrast, a subset of a standardized well-defined test set that was specified in [11] for the Special Session on Constrained Real Parameter Optimization at the Congress on Evolutionary Computation 2006 (CEC06) is used in this work. The reason for using only a subset of the mentioned test set is that for the examination of stopping criteria it is reasonable to use optimization problems for which the employed algorithm is able to converge reliably, meaning that convergence is obtained in every optimization run. In that case, the evaluation of stopping criteria is simplified because a convergence rate of less than 100% can be considered to be a result of unsuitable stopping conditions. The performance of the DE variant that is used in the present examination has already been analyzed adhering to the demands of [11] in a former study [10]. Based on this study, 16 out of 24 test functions have been selected for which a reliable convergence behavior has been found in [10].

Based on the previous considerations, the remainder of this chapter is organized as follows: In Sect. 2 the specification of the Differential Evolution variant that is used for the present examination is given. In Sect. 3 an overview about stopping criteria is provided, including a discussion if they can also possibly be used for other evolutionary algorithms besides DE. The description of experimental settings in Sect. 4 specifies parameter settings of DE, parameter settings of the stopping criteria and the performance measures that are applied in this work. Results are discussed in Sect. 5, and Sect. 6 ends with conclusions about the suitability of the presented stopping criteria for Differential Evolution.

2 Differential Evolution

In this section first the general process of Differential Evolution is described before giving details about the variant that is used here. For DE the positions of individuals are represented as real-coded vectors which are randomly initialized inside the limits of the given search space in the beginning of an optimization run (see Fig. 1). The individuals are evolved during the optimization run by applying mutation, recombination and selection to each individual in every generation. A stopping criterion determines after the building of every new generation if the optimization run should be terminated.

In this work the Differential Evolution algorithm is used in the variant DE/rand/1/bin [12]. This notation means that in the mutation process a randomly chosen population member $\mathbf{x}_{r_1}$ is added to one vector difference (also built from two randomly chosen members $\mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$ of the current population) where $\mathbf{x}_{r_1}$, $\mathbf{x}_{r_2}$, $\mathbf{x}_{r_3}$ and the so-called target vector $\mathbf{x}_i$ are mutually different:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{1}$$

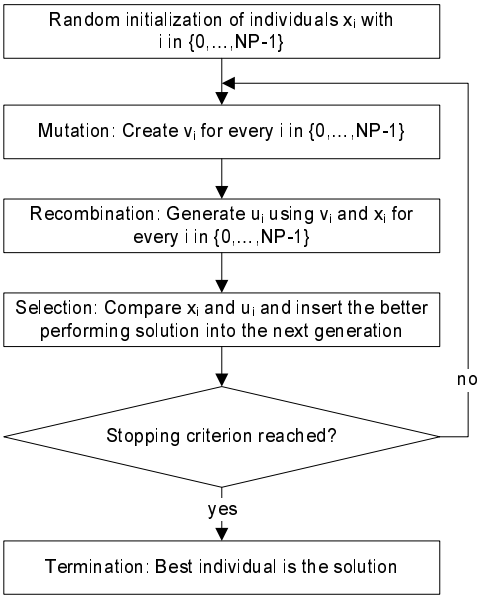


Fig. 1. Flowchart of Differential Evolution

F is a control parameter of DE that is usually chosen from the interval $[0, 1]$. Best values are usually in the range $[0.5, 0.9]$ as indicated in [13, 14, 15, 16].

Furthermore, the notation of the variant DE/rand/1/bin specifies that a binomial recombination process is used that can be written as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand_j \leq CR \text{ or } j = k \\ x_{i,j} & \text{otherwise} \end{cases} \tag{2}$$

Equation 2 generates the so-called trial vector $\mathbf{u}_i$ by copying components from the mutated vector $\mathbf{v}_i$ and the target vector $\mathbf{x}_i$ in dependence on a random number $rand_j \in [0, 1]$ that is compared to the control parameter CR (where $rand_j$ is chosen anew for each parameter in every generation). Good settings for CR are typically close to one but for some functions also small values have been reported to yield good results [13, 14, 15, 16]. Because during the selection process the target vector and the corresponding trial vector will compete for a place in the subsequent generation, it is ensured that $\mathbf{u}_i \neq \mathbf{x}_i$ by selecting at least one component from the mutated vector $\mathbf{v}_i$. For this purpose the variable $k \in \{0, \ldots, D-1\}$ (where D is the dimension of the optimization problem that is equal to the number of objective function parameters) is randomly chosen for every trial vector in each generation, and the k-th component of the trial vector is copied from the mutated vector.

Hence, four population members are involved in the creation of each trial vector which leads to an adaptive scaling of step sizes because the magnitude of the vector differences varies during the different stages of an optimization run.

Selection is a deterministic process in DE: The target vector and the trial vector are compared to each other, and the one with the lower objective function value (for minimization problems like in this work) is inserted into the next generation. Because this selection scheme allows only improvement but not deterioration of the objective function value, it is called greedy [14]. A further characteristic of the DE selection process is that the best objective function value cannot get lost when moving from one generation to the next. This property is called elitist, and it is usually associated with fast convergence behavior [6].

Mutation, recombination and selection is applied to every population member $\mathbf{x}_i$ with $i \in \{0, \ldots, NP-1\}$ in each generation where NP specifies the population size that has to be adjusted by the user. The fact that the evolutionary operators are applied to every population member is a property that distinguishes DE from several other evolutionary algorithms which often select only a subset of the population for mating [17]. In that case, individuals with better characteristics generally have better chances to reproduce, resulting in a possible increase of the convergence speed but also in loss of diversity. Because DE already generates enough convergence pressure by using an elitist selection procedure, diversity is emphasized by allowing each individual to generate offspring.

Differential Evolution has originally been developed for unconstrained single-objective optimization. Hence, a method for constraint-handling has to be added if constrained optimization problems should be solved like in this work. Several different constraint-handling approaches have been suggested in the literature [4, 5, 6]. In this work a method is employed that is widely used because it is simple but effective. It does not change the mutation and recombination processes of DE but only modifies selection in the following way:

- Feasible individuals (meaning individuals that fulfill all constraints) are favored over infeasible individuals.
- In the comparison of two infeasible individuals the one with the lower sum of constraint violation wins.
- The original selection method is used in the comparison of two feasible individuals.

Using this approach the search is guided to feasible regions of the search space by preferring individuals with lower or no constraint violation. This technique is easy to use because no additional parameters have to be set. With a small modification it can also be used for multi-objective optimization [18].

Boundary constraints are treated as a special case of constraint functions here because especially for real-world problems it may be crucial that individuals stay inside certain boundaries. A position that exceeds a limit is reset to the middle between old position and boundary, so the boundary is approached asymptotically [19]:

$$u_{i,j,G+1} = \begin{cases} \frac{1}{2}\left(x_{i,j,G} + X_{max,j}\right) & \text{if } u_{i,j,G+1} > X_{max,j} \\ \frac{1}{2}\left(x_{i,j,G} + X_{min,j}\right) & \text{if } u_{i,j,G+1} < X_{min,j} \\ u_{i,j,G+1} & \text{otherwise} \end{cases} \quad (3)$$

where $X_{max,j}$ is the upper limit and $X_{min,j}$ is the lower limit for the j-th parameter, and Eq. 3 is given for the i-th individual in generation G.

3 Stopping Criteria

The stopping criteria which are used in this work are grouped into three classes:

- Improvement-based criteria,
- movement-based criteria and
- distribution-based criteria.

In the following several implementations of stopping criteria based on monitoring improvement, movement and distribution are summarized. Most of the stopping criteria that are presented here have already been used for DE [20, 21]. Additionally, for one of them a generalization is newly introduced here because it has been indicated elsewhere [22] that the generalized criterion may exhibit improved behavior over the special case that was formerly used. Many of the stopping criteria that are examined in this work can also be employed for other evolutionary algorithms but it was shown that the performance is not necessarily equal [20, 22]. This conclusion is also reached in [23] where it is stated that the effectiveness of a stopping criterion is closely related to the procedure of a certain optimization strategy and not automatically transferable to other algorithms. Therefore, in the following description of stopping criteria references are added, if available, in which other context or for which other optimization algorithm the stopping criteria can be used also.

Every criterion that is presented here includes one or two specific parameters which have to be set by the user. This property seems to be inherent to all stopping criteria because even if a problem with known optimum is used and termination is done when the optimum is found, the accuracy has to be set by the user. Similarly, parameter FE_{max} has to be set when criterion *LimFuncEval* is employed. Usually, no general guidelines can be given for the setting of FE_{max} but the adaptive stopping criteria do not necessarily have this property as will be shown in this work.

3.1 Improvement-Based Criteria

If the improvement of the objective function value decreases to a small value for some time, it can be assumed that convergence has been obtained. Because improvement can be measured in different ways, three conditions are examined here:

- *ImpBest*: The improvement of the best objective function value of each generation is monitored. If it falls below a user-defined threshold t for a number of generations g, the optimization run will be terminated.

A similar approach is also discussed in [24] for Particle Swarm Optimization (PSO) and furthermore in [25] to determine a suitable switch-over point from a Genetic Algorithm to a local optimization technique.

- *ImpAv*: Because the best objective function value might not correctly reflect the state of the whole population, the average improvement computed from the whole population is examined for this criterion. Similar to *ImpBest*, an optimization run is terminated if the average improvement is below a given threshold t for g generations.

  This criterion is also used in [26] to stop a local search procedure that is embedded in a Genetic Algorithm. For the same purpose similar criteria as *ImpBest* and *ImpAv* are also employed in [27].
- *NoAcc*: Because DE incorporates a greedy selection scheme, the acceptance of trial vectors means that there is improvement in the population. Based on this fact, it is monitored if still trial vectors have been accepted in a specified number of generations g, and the optimization run is terminated if this condition is violated.

  NoAcc has the advantage that only one parameter has to be set whereas all other stopping conditions that are presented here require the setting of two parameters. However, in contrast to the other improvement-based criteria, it is specific to the functionality of DE and may not be assignable to other evolutionary algorithms. For PSO *NoAcc* can be adapted by observing if new personal best positions have been found in a predefined number of generations but in [22] the performance of this criterion was poor.

  NoAcc is also described for DE in [6], and it is recommended to set g not too low because long periods without improvement may occur during optimization runs.

3.2 Movement-Based Criteria

In the beginning of an optimization run the individuals are randomly scattered in the search space, and large step sizes are generated in mutation and recombination. Towards the end of an optimization run the population generally converges to one point in the search space. Thus, step sizes become small because of the adaptive scaling of DE. As a result, the movement of individuals in parameter space can also be used to derive a stopping criterion (parameter space means that the positions of the individuals are regarded while objective space refers to the objective function values of the individuals):

- *MovPar*: If the average movement of the population members is below a threshold t for a given number of generations g, the optimization run is terminated.

  MovPar is also usable for other evolutionary algorithms with real-coded variables. It might be possible to adapt it also for binary-coded individuals if a suitable distance measure can be found. Moreover, stopping criteria like this are used in classical optimization algorithms like hill climbing techniques [23].

Movement can also be measured in objective space but because of the greedy selection scheme of DE, the objective function value can only improve but not deteriorate. Therefore, a stopping criterion based on movement in objective space would be equal to an improvement-based criterion. In contrast, a criterion *MovObj* could be used for other evolutionary algorithms which permit deterioration of objective function values:

- *MovObj*: If the average movement of the population members in objective space is below a threshold t for g generations, the optimization run is stopped.

3.3 Distribution-Based Criteria

In single-objective optimization DE individuals usually converge to one point in the search space towards the end of an optimization run. As a result, the distribution of individuals can be used to derive conclusions about the state of an optimization run. Several possibilities exist to measure the distribution of individuals. One of the easiest alternatives is the following:

- *MaxDist*: The maximum distance of any population member to the individual with the best objective function value is monitored in parameter space. If it falls below a threshold m, the optimization run will be terminated.
  A similar criterion is also discussed in [24] for PSO.

If the positions of all population members should be regarded instead of observing only the maximum distance to the best individual, the following stopping criterion can be used:

- *StdDev*: The standard deviation of positions of all population members is examined. The optimization run is stopped if it drops below a given threshold m.
  In [28] a similar criterion is also used for DE.

Especially for Particle Swarm Optimization it has been shown that a generalization of *MaxDist* has advantages [20, 22]:

- *MaxDistQuick*: Instead of examining the maximum distance of all population members to the current best individual, only a subset of the current population is used. For this purpose, the population members are sorted due to their objective function value using a Quicksort algorithm, and only for the best p% of the population it is checked if their distance is below a threshold m. Because a feasible solution is wanted, it is also checked if the best p% of the individuals are feasible.
  MaxDist can be derived from *MaxDistQuick* by setting p to 100%.

In [22] it was concluded for PSO that it might be beneficial if a generalization of *StdDev* is also examined because it was shown that the generalized criterion *MaxDistQuick* has advantages over the special case *MaxDist*. *StdDev* and *MaxDist* rely on similar mechanisms, so it can be expected that the performance of a generalized criterion might also be better for *StdDev*. Consequently, the following criterion is newly introduced here:

- *StdDevQuick*: Similar to *MaxDistQuick*, the population is first sorted due to their objective function value using a Quicksort algorithm. The standard deviation of positions is then calculated for the best $p\%$ of the population and compared to the user-defined threshold m. Again, it is also examined if the best $p\%$ of the individuals are feasible.

  Similar to the relationship between *MaxDist* and *MaxDistQuick*, *StdDev* is a special case of *StdDevQuick* with $p = 100\%$.

Because *MaxDist* and *StdDev* are special cases of *MaxDistQuick* and *StdDevQuick*, respectively, only the generalizations *MaxDistQuick* and *StdDevQuick* are regarded in the following.

All distribution-based criteria that have been mentioned so far are calculated in parameter space. Another possibility to evaluate the distribution of the population members is to regard objective space:

- *Diff*: The difference between best and worst objective function value in a generation is checked if it is below a given threshold d. Furthermore, it is demanded that at least $p\%$ of the individuals are feasible because otherwise *Diff* could lead to early termination of an optimization run if e.g. only two individuals are feasible and they are close to each other by chance but the population has not converged yet.

  A similar implementation of this criterion without parameter p is described in [6, 23] and also used in [29] (interestingly, it will be shown in the following that the results of the present examination indicate that the performance of *Diff* is independent from p so it may be omitted). It is recommended in [6] to set d to a value that is several orders of magnitude lower than the desired accuracy of the optimum.

No DE-specific information is used for the distribution-based criteria so in principle they can be used for other algorithms also. However, if another representation than real-coded vectors is used for the positions of the individuals, the distribution-based criteria in parameter space will have to be adapted.

3.4 Combined Criteria

Because functions have different features it can be concluded that a combination of different stopping criteria may result in good performance. For example an criterion like *Diff* that is easy to check can be tested first. Because the first criterion might fail for certain characteristics of the objective function (e.g. it was shown in former work [20] that *Diff* fails for functions with a flat surface), a second criterion that is based on another mechanism might be evaluated after the stopping condition of the first criterion has been fulfilled. In former work the following combined criteria were tested:

- *ComCrit*: First, the improvement-based criterion *ImpAv* is evaluated. If *ImpAv* indicates that the optimization run should be stopped, the distribution-based criterion *MaxDist* is regarded additionally. *ComCrit* was examined in [20, 21] for DE and also in [22] for PSO.

- *Diff_MaxDistQuick*: In this case, distribution-based criteria in objective and parameter space were joined (*Diff* and *MaxDistQuick*) so that *MaxDistQuick* is only checked if the stopping condition of *Diff* has been fulfilled. Up to now this criterion has only been applied for PSO in [22].

In all former examinations the combined criteria always needed more function evaluations for detecting convergence than the individual criteria. Furthermore, selecting appropriate parameter settings was complicated because three parameters have to be set for each stopping criterion in contrast to one or two parameters for the individual criteria, respectively. Moreover, the connection between parameter settings and problem features like desired accuracy was obliterated. Because of these disadvantages, combined criteria are not considered further in this work.

4 Experimental Settings

To be able to derive general conclusions about the suitability of stopping criteria for a broad range of optimization problems, 16 test functions are used here. They are chosen from the standardized test set that was used in the Special Session on Constrained Real Parameter Optimization at the Congress on Evolutionary Computation 2006. The test set originally consists of 24 constrained single-objective test functions (g01–g24) but this work concentrates on the functions for which a convergence rate of 100% has been found in former work [10] for the same algorithm with the same parameter settings ($F = 0.7$, $CR = 0.9$, $NP = 50$). In this case, the analysis of stopping criteria is simplified because performance variations concerning the convergence rate can be accredited to the unsuitability of stopping criteria. Because of these considerations, functions g02, g03, g13, g17, g20, g21, g22 and g23 are omitted here. Nevertheless, the remaining functions permits extensive testing of stopping criteria because a broad spectrum of different features is represented by them:

- Dimensionality,
- type of function,
- ratio of feasible space to the whole search space,
- linear and nonlinear inequality and equality constraints,
- active constraints at the optimum (meaning that the optimum is located at the boundary of one or more constraints) and
- disconnected feasible regions.

Because the exact definition of the test functions has been shown in several works, it is not repeated here as it takes a lot of space. Instead, the interested reader should refer to [11, 30, 31, 32], while some general information is also given in Table 1. In Table 1 $\rho = \frac{|\mathcal{F}|}{\mathcal{S}}$ specifies the estimated ratio of feasible space to the whole search space. The following four columns of Table 1 give information about the number and type of constraints: LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear

Table 1. Details of the test functions (from [10] and [11])

Problem	D	Type of function	ρ	LI	NI	LE	NE	a	Median FEs in [10]
g01	13	quadratic	0.0111%	9	0	0	0	6	32996
g04	5	quadratic	52.1230%	0	6	0	0	2	16166
g05	4	cubic	0.0000%	2	0	0	3	3	105780
g06	2	cubic	0.0066%	0	2	0	0	2	7198
g07	10	quadratic	0.0003%	3	5	0	0	6	93752
g08	2	nonlinear	0.8560%	0	2	0	0	0	1091
g09	7	polynomial	0.5121%	0	4	0	0	2	25602
g10	8	linear	0.0010%	3	3	0	0	6	120624
g11	2	quadratic	0.0000%	0	0	0	1	1	14993
g12	3	quadratic	4.7713%	0	1	0	0	0	5398
g14	10	nonlinear	0.0000%	0	0	3	0	3	68147
g15	3	quadratic	0.0000%	0	0	1	1	2	51619
g16	5	nonlinear	0.0204%	4	34	0	0	4	11522
g18	9	quadratic	0.0000%	0	13	0	0	6	80322
g19	15	nonlinear	33.4761%	0	5	0	0	0	176127
g24	2	linear	79.6556%	0	2	0	0	2	3067

equality constraints. Besides, the number of active constraints at the optimum is given by a. Table 1 also gives information about the median number of function evaluations that have been needed for convergence for the same algorithm in former work [10] because it will be shown that the behavior of some stopping criteria is dependent on it.

Each stopping criterion includes one or two parameters. The parameter settings that are examined in this work are given in Table 2. For every parameter combination and each test function 100 independent runs are conducted. In the CEC06 special session 500,000 FEs were allowed for solving each optimization problem [11], so a maximum number of $FE_{max} = 500,000$ is used in connection with each stopping criterion to terminate the optimization run if the stopping criterion is not able to do it. However, an optimization run that is stopped at 500,000 FEs is considered as unsuccessful. In successful runs the execution of the algorithm must be stopped before reaching 500,000 FEs, and the optimum must be located with an accuracy of 10^{-4} as it was required in [11] for the CEC06 special session (naturally, the solution must also be feasible where the allowed remaining constraint violation for equality constraints is 10^{-4} as in [11]).

Two aspects are important for the assessment of the performance of stopping criteria:

- Has convergence been achieved i.e. has the optimum been reached with the desired accuracy before the algorithm was terminated?
- How fast was the termination i.e. how many function evaluations were done after convergence has been reached?

Table 2. Parameter settings for the stopping criteria

Criterion	Parameter	Start value	Stop value	Modifier
ImpBest	t	1e-2	1e-6	$\cdot$ 1e-1
	g	5	20	+ 5
ImpAv	t	1e-2	1e-6	$\cdot$ 1e-1
	g	5	20	+ 5
NoAcc	g	1	5	+ 1
MovPar	t	1e-2	1e-6	$\cdot$ 1e-1
	g	5	20	+ 5
MaxDistQuick	m	1e-2	1e-5	$\cdot$ 1e-1
	p	0.1	1.0	+ 0.1
StdDevQuick	m	1e-2	1e-5	$\cdot$ 1e-1
	p	0.1	1.0	+ 0.1
Diff	d	1e-1	1e-6	$\cdot$ 1e-1
	p	0.1	1.0	+ 0.1

The first performance measure is evaluated by computing the percentage of successful runs out of 100 independent optimization runs. Clearly, the first performance measure is more important than the second here because fastness is irrelevant if convergence is not obtained. Given that a sufficient convergence rate has been achieved, the second performance measure also provides important information about the abilities of stopping criteria. It is examined by calculating the additional computational effort $\frac{FE_{stop}-FE_{conv}}{FE_{conv}}$: The difference between the number of function evaluations at which the execution of the algorithm is terminated (FE_{stop}) and the number of function evaluations at which convergence is achieved for the first time (FE_{conv}) is computed and the result is divided by FE_{conv}. Thereby, the additional computational effort is normalized because the test problems require very different amounts of FEs for convergence. For both performance measures the median is calculated for each function (the median is preferred instead of the average that is often used in the literature because it is more robust to outliers).

Due to the high amount of data that has been collected for this examination, only general results can be visualized instead of going into detail. Therefore, box plots of the number of successful runs and the additional computational effort will be shown that provide a concise overview over the performance for all 16 optimization problems. Hence, performance over a large range of functions can be easily evaluated for each combination of parameter settings of the stopping criteria from Table 2.

Naturally, different parameter settings of stopping criteria are required if the demanded accuracy of the result is varied. As a consequence, all examinations are repeated with an accuracy of $\epsilon = 10^{-2}$ in order to make comparisons with the formerly used accuracy of $\epsilon = 10^{-4}$ that was specified in [11]. Because the

visualization of results takes a lot of space, the results for $\epsilon = 10^{-2}$ will only be summarized qualitatively in the text.

5 Results

In this section the results based on the experimental settings discussed in the previous section are shown for each stopping criterion.

5.1 Criterion *ImpBest*

Criterion *ImpBest* has a very bad performance (see Fig. 2). For many functions the DE algorithm is stopped too early so the convergence rate is low. Only for g08, g16 and g24 a rather high convergence rate has been achieved for certain parameter settings (visible in Fig. 2 as outliers in the box plots). When searching for commonalities of functions with a good performance, it is noticeable that function g08 for which the best results have been achieved needs the least amount of function evaluations for convergence of all functions (see Table 1). g16 and g24 also belong to the functions which can be optimized with a comparably low amount of function evaluations. With $\epsilon = 10^{-2}$ the convergence rate improves for several functions but again only for functions which need a low amount of function evaluations for convergence (e.g. g01, g06, g11, g12). Therefore, the conclusion can be derived that *ImpBest* can only be used for functions which can be optimized with low computational effort. However, obviously this property is not sufficient because a poor performance concerning convergence rate is yielded for several functions which can be optimized using few function evaluations.

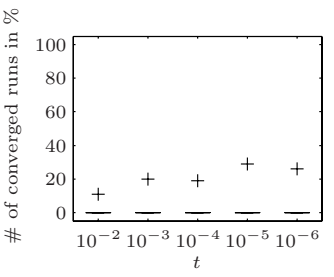
Concerning the additional computational effort, a moderate result has been achieved when compared to the performance of other stopping criteria. Naturally, the additional computational effort is always higher if $\epsilon = 10^{-2}$ is used with the same parameter settings of stopping criteria, respectively, because convergence is obtained earlier.

When analyzing the dependence of the convergence rate and the additional computational effort on parameter settings, it can be noticed that both slightly increase with decreasing improvement threshold t and increasing number of generations g. In general, results are very different for different functions, so it is not obvious how parameter settings should be chosen.

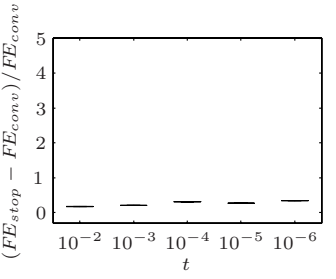
Mainly because of its bad results concerning convergence rate, *ImpBest* cannot be regarded as a reliable stopping criterion. Moreover, the applicability of *ImpBest* is complicated because of the difficulty that is connected with choosing parameter settings. Former work on stopping criteria for DE also supports this conclusion [20, 21].

5.2 Criterion *ImpAv*

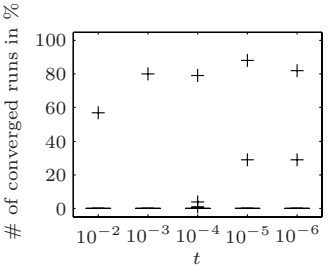
Criterion *ImpAv* shows a better performance than *ImpBest* but again the algorithm is constantly terminated too early for many functions (see Fig. 3). For



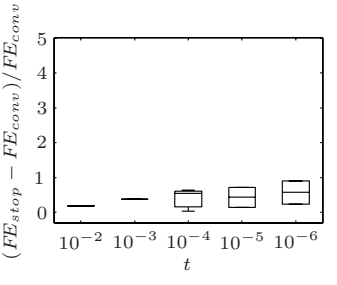
(a) Percentage of successful runs for $g = 5$



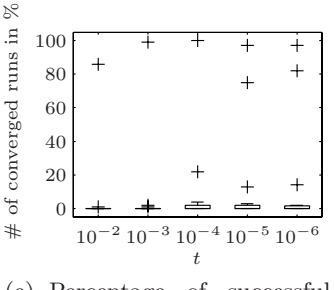
(b) Additional computational effort for $g = 5$



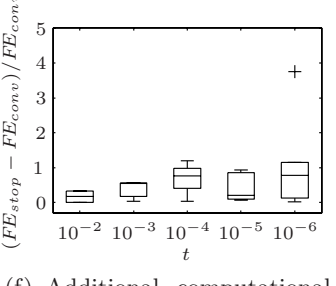
(c) Percentage of successful runs for $g = 10$



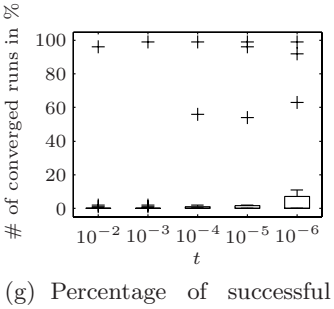
(d) Additional computational effort for $g = 10$



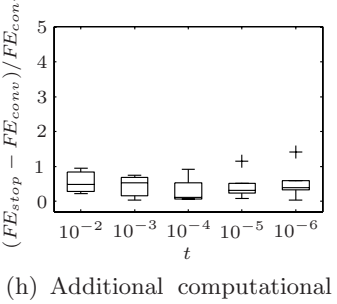
(e) Percentage of successful runs for $g = 15$



(f) Additional computational effort for $g = 15$

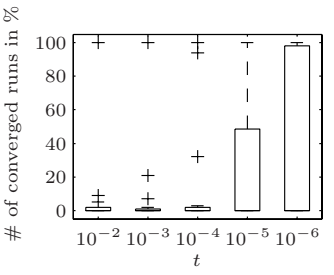


(g) Percentage of successful runs for $g = 20$

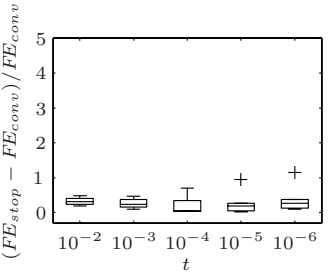


(h) Additional computational effort for $g = 20$

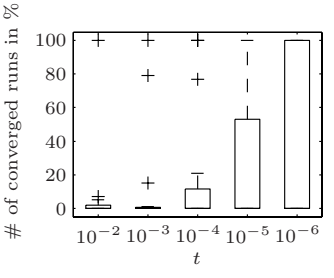
Fig. 2. Results for criterion *ImpBest*



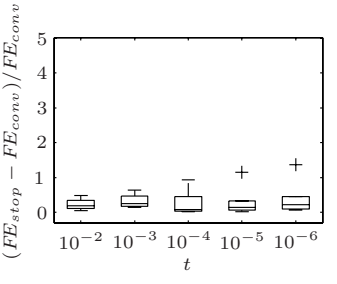
(a) Percentage of successful runs for $g = 5$



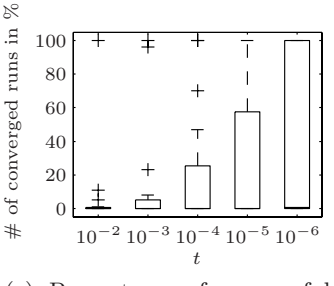
(b) Additional computational effort for $g = 5$



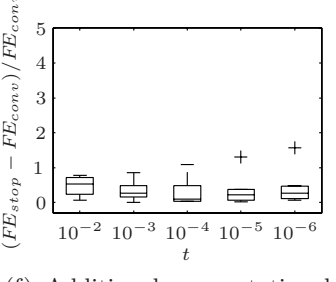
(c) Percentage of successful runs for $g = 10$



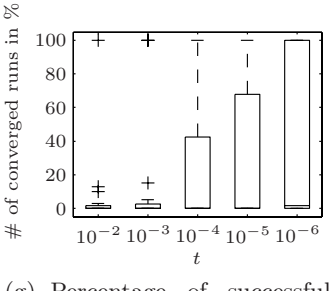
(d) Additional computational effort for $g = 10$



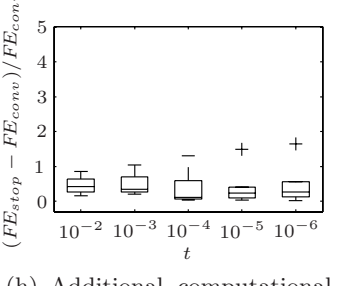
(e) Percentage of successful runs for $g = 15$



(f) Additional computational effort for $g = 15$



(g) Percentage of successful runs for $g = 20$



(h) Additional computational effort for $g = 20$

Fig. 3. Results for criterion *ImpAv*

$\epsilon = 10^{-4}$ the functions with better performance share the same property as for *ImpBest* which is that a relatively small number of function evaluations is needed for convergence. For $\epsilon = 10^{-2}$ this property is less pronounced, and there are only two functions left for which convergence is never reached before termination of the algorithm (g04 and g10).

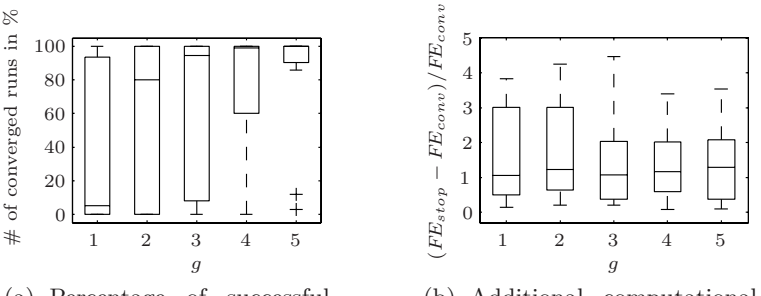
Again, it is difficult to choose parameter settings because results differ for the functions, and no commonalities are visible. Generally, a definitive dependence of the performance on the threshold of improvement t can be seen where the convergence rate improves and the additional computational effort degrades for decreasing values of t. There is also a small difference in performance for varying numbers of generations g as the convergence rate and the additional computation effort slightly increase for higher settings of g. A similar result is also shown in [21] where the performance varies considerably for different settings of the parameters.

The additional computational effort is in moderate range for $\epsilon = 10^{-4}$, similar as for *ImpBest*, but it increases dramatically for $\epsilon = 10^{-2}$ with the same parameter settings, respectively.

Summing up, *ImpAv* cannot be regarded as a reliable stopping criterion, neither: No general guidelines for parameter settings can be given, and furthermore there are other criteria that result in faster detection of convergence. As a result, the use of *ImpAv* cannot be recommended.

5.3 Criterion *NoAcc*

For *NoAcc* mostly high convergence rates have been achieved, especially for high settings of the number of generations without improvement g (see Fig. 4). Unfortunately, especially for functions that can be optimized with a low amount of function evaluations like g08, g16 and g24 the additional computational effort is very high (see Fig. 4). There are several functions for which high convergence rates have already been reached for $g = 1$ (g06, g08, g24) but there are also functions for which even with parameter setting $g = 5$ no reliable detection of



(a) Percentage of successful runs

(b) Additional computational effort

Fig. 4. Results for criterion *NoAcc*

convergence can be achieved. For $\epsilon = 10^{-2}$ the convergence rates become higher but there are still two functions for which the convergence rate is only moderate (g18 and g19).

Although *NoAcc* has the advantage that it only incorporates one parameter, its reliability is limited because it may not be easy to set parameter g as the results concerning convergence rate vary for different functions. The additional computational complexity also reaches considerable magnitude for *NoAcc*, especially for $\epsilon = 10^{-2}$. In former work *NoAcc* has shown a good performance [21] but failed for a function with a flat surface [20].

In summary, *NoAcc* has the advantages that it incorporates only one parameter, and moreover it is easy to check as only the number of accepted trial vectors has to be counted. Nevertheless, it cannot be recommended without hesitation because suitable settings of parameter g vary for different functions, and the additional computational effort is very high for several functions. Additionally, parameter g cannot take values smaller than 1 but even this setting leads to a high additional computational effort for several functions. The missing possibility of scaling to lower values which would lead to earlier termination of the algorithm may be unfavorable.

5.4 Criterion *MovPar*

The convergence rate of criterion *MovPar* is dependent on both the threshold of improvement t and the number of generations g (see Fig. 5). For small settings of t and large settings of g convergence rates of 100% have been found for most functions. Exceptions are g01, g04 and g06 but it is not clear which property is the determining factor because the characteristics of these functions are quite dissimilar. In contrast, for $\epsilon = 10^{-2}$ convergence rates of 100% are found for all functions if parameter settings $g = 20$ and $t \leq 10^{-5}$ are used.

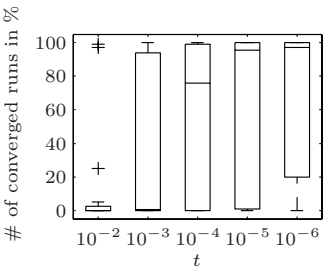
Concerning the additional computational effort, mostly moderate performance is shown with $\epsilon = 10^{-4}$, but it reaches a considerable size for few outliers (g08 and g18). Again, it increases strongly for $\epsilon = 10^{-2}$ with the same parameter settings.

Similar as for *ImpBest* and *ImpAv*, determination of suitable settings for the parameters is not easy which can also be seen in former work [21]. In the present examination mostly settings of $t \leq 10^{-5}$ and $g \geq 10$ yielded good results.

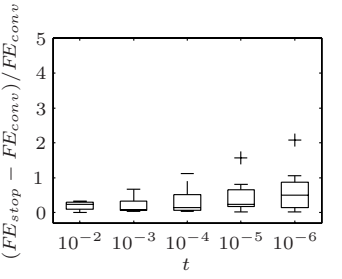
To sum up, it can be stated that *MovPar* yields better results than *ImpBest* and *ImpAv* here, but there are still a few functions where no reliable convergence behavior was observed for $\epsilon = 10^{-4}$. It can be argued that different parameter settings may result in better convergence rates but it must also be considered that the additional computational effort becomes large for some functions. As a consequence, *MovPar* cannot be recommended as a stopping criterion for DE.

5.5 Criterion *MaxDistQuick*

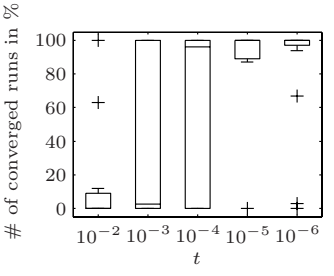
MaxDistQuick achieved very good results in former work on stopping criteria for DE [20, 21] but with the broad set of functions that is used in this work the



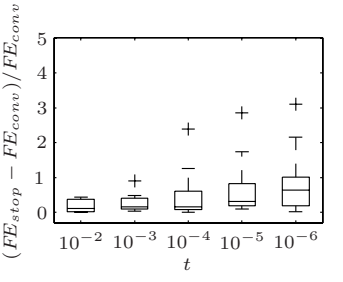
(a) Percentage of successful runs for $g = 5$



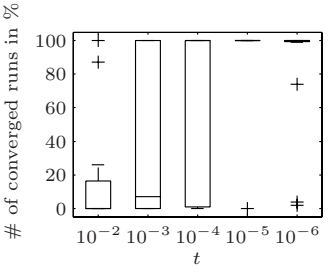
(b) Additional computational effort for $g = 5$



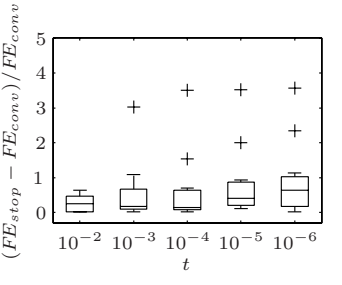
(c) Percentage of successful runs for $g = 10$



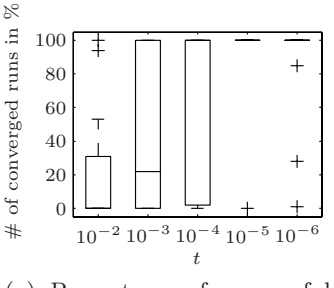
(d) Additional computational effort for $g = 10$



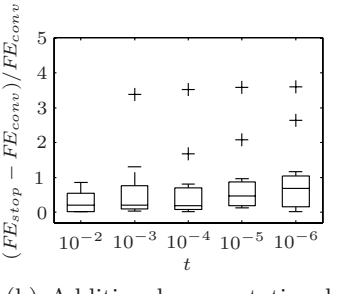
(e) Percentage of successful runs for $g = 15$



(f) Additional computational effort for $g = 15$

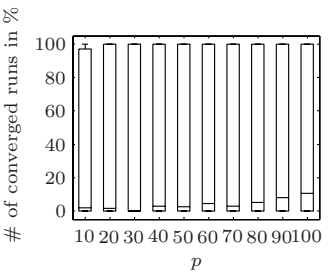


(g) Percentage of successful runs for $g = 20$

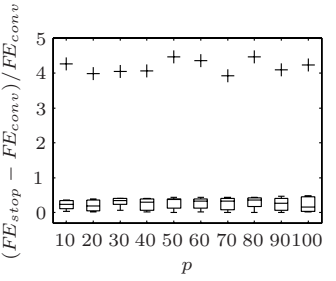


(h) Additional computational effort for $g = 20$

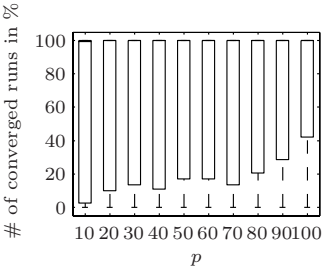
Fig. 5. Results for criterion *MovPar*



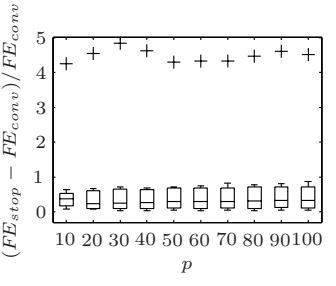
(a) Percentage of successful runs for $m = 10^{-2}$



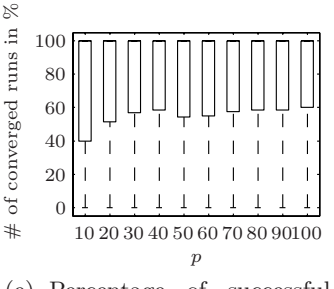
(b) Additional computational effort for $m = 10^{-2}$



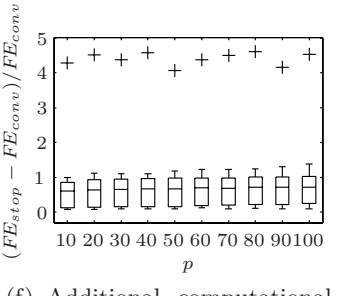
(c) Percentage of successful runs for $m = 10^{-3}$



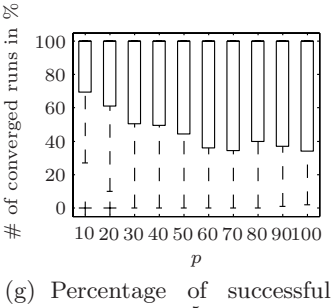
(d) Additional computational effort for $m = 10^{-3}$



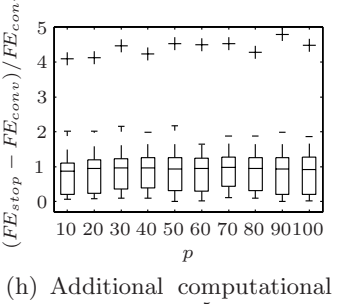
(e) Percentage of successful runs for $m = 10^{-4}$



(f) Additional computational effort for $m = 10^{-4}$



(g) Percentage of successful runs for $m = 10^{-5}$



(h) Additional computational effort for $m = 10^{-5}$

Fig. 6. Results for criterion *MaxDistQuick*

evaluation of its performance is not that clear (see Fig. 6). For several functions convergence rates of 100% have been reached but for other functions no parameter settings have resulted in good convergence behavior (g04, g06, g18). For g18 this is still the case for $\epsilon = 10^{-2}$. In contrast, convergence rates of 100% could be obtained for all other functions with $\epsilon = 10^{-2}$ if proper parameter settings are used.

For most functions the convergence rate slightly increases for growing p if m is large, but for small m the opposite effect can be seen as the convergence rate decreases for increasing p in this case. This is due to the fact that with a small setting of m convergence is often not detected before reaching the maximum number of function evaluations which leads to a decrease of convergence rate. Hence, for several functions convergence rates of 100% are already reached for $m = 10^{-2}$ (g05, g08, g10, g12, g16) but e.g. for g05 and g10 it is decreased for $m = 10^{-5}$.

The additional computation effort mostly shows a good or at least moderate performance (see Fig. 6). There is only one outlier that is caused by g18 because the global optimum is always found a long time before all population members have converged to the required distance from the optimum, so in that case a larger setting of m would be required. For other functions generally an increase of the additional computational effort can be seen for decreasing m.

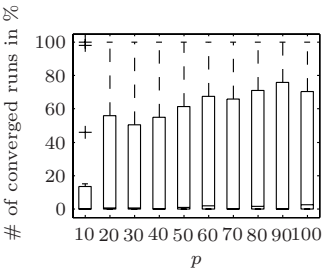
In summary, *MaxDistQuick* is an interesting criterion that leads to reliable termination of the DE algorithm when proper parameter settings have been found. Thus, some test runs will be necessary when applying *MaxDistQuick*, similar as for *LimFuncEval*.

5.6 Criterion *StdDevQuick*

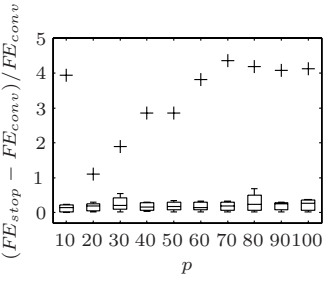
For *StdDevQuick* similar results are obtained as for *MaxDistQuick* (see Fig. 7). This outcome was expected because both stopping criteria rely on similar mechanisms. Nevertheless, there are some differences: As it was also noticed for *MaxDist* and *StdDev* in former work [21], generally the setting of m has to be lower for *StdDevQuick* to yield the same convergence rate as for *MaxDistQuick*, so the results are often shifted. There are two functions (g06, g18) for which even with $\epsilon = 10^{-2}$ no satisfactory convergence rate has been achieved. For $\epsilon = 10^{-4}$ three other functions also yielded bad performance (g01, g04 and g15).

For the development of the convergence rate as well as for the additional computational effort, the same general dependence on parameter settings was seen as for *MaxDistQuick*. The outlier that can be seen in the additional computational effort in Fig. 7 is again caused by g18.

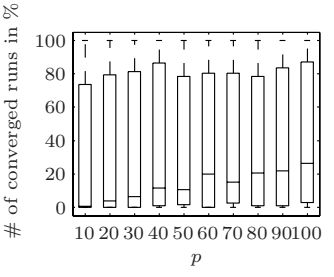
Recapitulating, it can be stated that although the results are shifted in contrast to *MaxDistQuick*, the same conclusions can be derived which are that reliable detection of convergence is obtained if suitable parameter settings are used, but the proper settings may be different for varying functions.



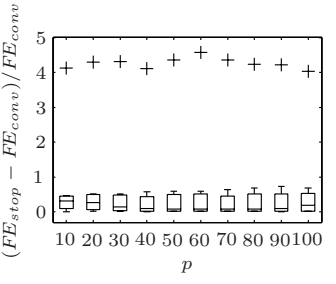
(a) Percentage of successful runs for $m = 10^{-2}$



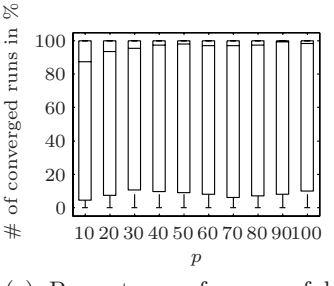
(b) Additional computational effort for $m = 10^{-2}$



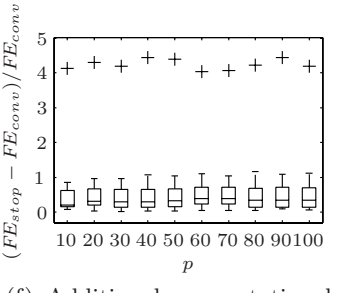
(c) Percentage of successful runs for $m = 10^{-3}$



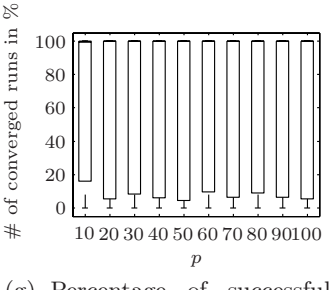
(d) Additional computational effort for $m = 10^{-3}$



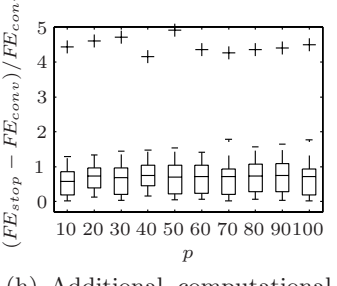
(e) Percentage of successful runs for $m = 10^{-4}$



(f) Additional computational effort for $m = 10^{-4}$

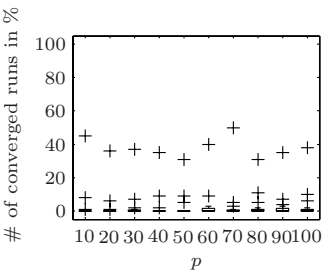


(g) Percentage of successful runs for $m = 10^{-5}$

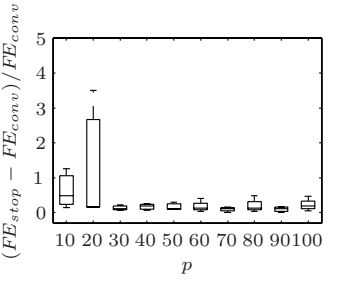


(h) Additional computational effort for $m = 10^{-5}$

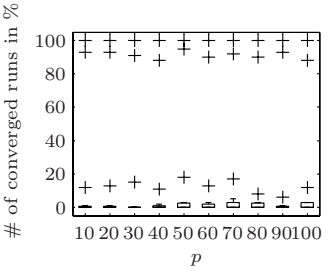
Fig. 7. Results for criterion *StdDevQuick*



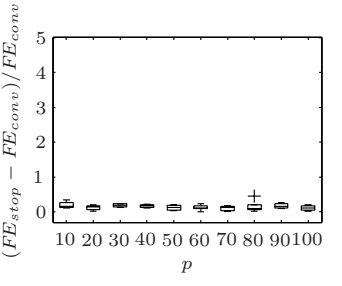
(a) Percentage of successful runs for $d = 10^{-1}$



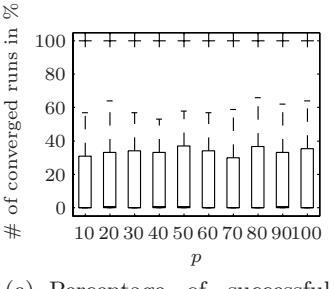
(b) Additional computational effort for $d = 10^{-1}$



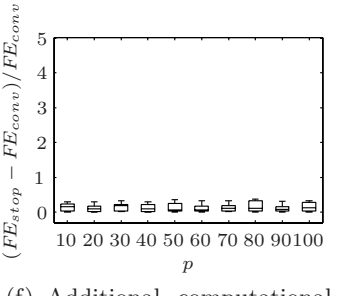
(c) Percentage of successful runs for $d = 10^{-2}$



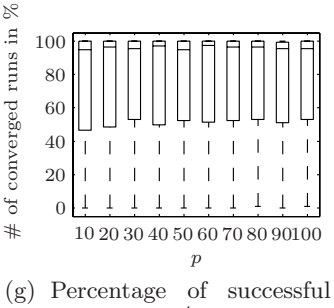
(d) Additional computational effort for $d = 10^{-2}$



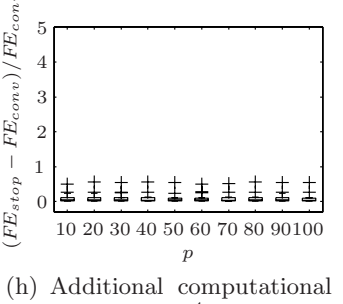
(e) Percentage of successful runs for $d = 10^{-3}$



(f) Additional computational effort for $d = 10^{-3}$

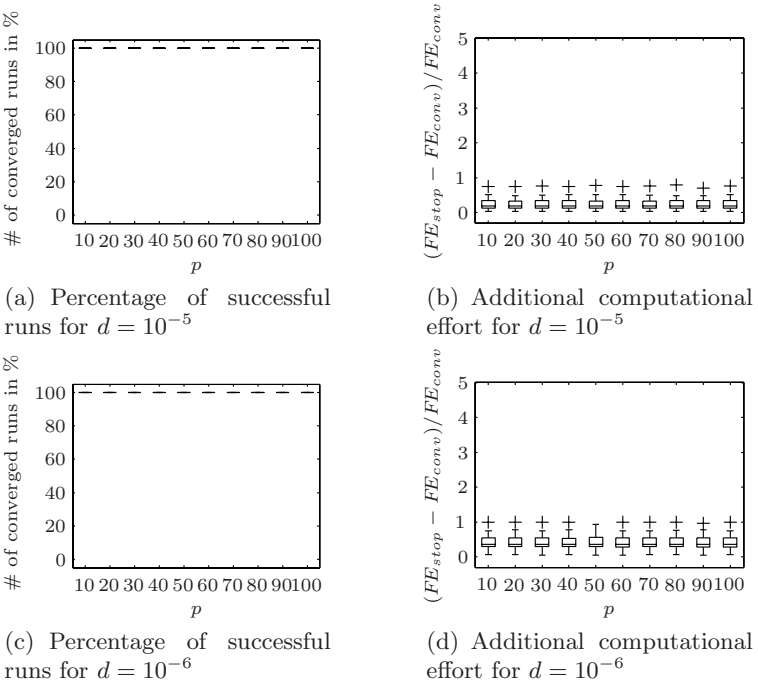


(g) Percentage of successful runs for $d = 10^{-4}$



(h) Additional computational effort for $d = 10^{-4}$

Fig. 8. Results for criterion *Diff*



(a) Percentage of successful runs for $d = 10^{-5}$

(b) Additional computational effort for $d = 10^{-5}$

(c) Percentage of successful runs for $d = 10^{-6}$

(d) Additional computational effort for $d = 10^{-6}$

Fig. 9. Results for criterion *Diff* (continued)

5.7 Criterion *Diff*

The results of *Diff* are the most promising ones of this examination (see Fig. 8 and 9). Convergence rates of 100% have been achieved for all functions when the difference threshold is set to $d \leq 10^{-5}$ which is exactly one order of magnitude smaller than the demanded accuracy of $\epsilon = 10^{-4}$. For $\epsilon = 10^{-2}$ a similar result is obtained as convergence rates of 100% have been achieved for all functions with $d \leq 10^{-3}$ which is again one order of magnitude smaller than the desired accuracy. It can be concluded that choosing suitable settings of parameter d is an easy task because a connection to the demanded accuracy can be made.

For parameter p that denotes the percentage of the population that is demanded to be feasible, no dependence can be noticed, neither regarding convergence rate nor regarding the additional computational effort. Thus, parameter p can be omitted for criterion *Diff*. In this case, the number of parameters for *Diff* is reduced from two to one, contributing to its simplicity.

The additional computation effort is relatively low when compared to the results of other stopping criteria (see Fig. 8 and 9) which is also an advantage of *Diff*.

One limitation of *Diff* is that it yields bad results when an optimization problem contains an objective function with a flat surface, meaning that the same objective function value is yielded for a large subset of the search space

[20]. Fortunately, it can be argued that this is a special case that rarely occurs. Moreover, it can be discovered easily when the optimization run is monitored, so in that case another stopping criterion could be employed. Otherwise, the performance of *Diff* was also good in former work [21].

In summary, *Diff* has led to the best results of the stopping criteria that were examined here because all functions could be successfully terminated. Additionally, parameter p can be omitted completely as no dependence on it could be found. Choosing of parameter d is also simple because the results of this work indicate that it is sufficient to set it to one order of magnitude lower than the desired accuracy. Moreover, the results of *Diff* concerning the additional computational effort are good in contrast to other stopping criteria.

6 Conclusions

In this chapter stopping criteria have been presented that react adaptively to the state of an optimization run by considering the improvement, movement or distribution of population members. The use of adaptive stopping criteria was motivated by the fact that they could help to avoid the unnecessary high computational effort that is usually associated with stopping after a preassigned fixed number of function evaluations because of variations in the number of function evaluations that are needed for convergence. This approach is mainly intended for real-world problems because in this case normally the optimum is unknown.

The best results were yielded by criterion *Diff* that can be classified as a distribution-based criterion in objective space as it terminates an optimization run if the difference between best and worst objective function value in the current generation has fallen below a given threshold d. This work has shown that the setting of d is linked with the desired accuracy of the result, so choosing a suitable parameter setting for d is simple. The second parameter p that corresponds to the demanded feasible percentage of the population can be omitted completely as no dependence on it could be seen. The only limitation of *Diff* was revealed in former work where *Diff* failed for a function with a flat surface [20]. However, this property can be detected easily when observing an optimization run so this limitation is not grave.

The results for other stopping criteria were not as distinct as for *Diff*. Two distribution-based criteria in parameter space were examined that terminate an optimization run if the maximum distance of a specified subset of the population to the best population member (*MaxDistQuick*) or the standard deviation of a subset of population members (*StdDevQuick*) is below a user-defined threshold m, respectively. Most functions could be successfully terminated in reasonable time but the parameter settings that yielded these results varied for different functions. There are few functions for which none of the examined parameter settings were able to induce convergence rates of 100% (even with a decreased demanded accuracy of $\epsilon = 10^{-2}$) but it is assumed that higher settings of m might result in better performance for these functions.

Apart from distribution-based stopping criteria, also a movement-based criterion in parameter space *MovPar* was examined that induces termination of an optimization run if the movement of the individuals in parameter space falls below a given percentage t in a predefined number of generations g. For three functions bad results concerning the convergence rate were found regardless of parameter settings for an accuracy of $\epsilon = 10^{-4}$ while for the decreased accuracy of $\epsilon = 10^{-2}$ all functions could be successfully terminated after convergence has been obtained. It can be concluded that the criterion is basically able to stop optimization runs reliably if suitable parameter settings have been found.

Two implementations of improvement-based criteria yielded the worst results of this examination, where criterion *ImpBest* that observes the improvement of the best objective function value is yet worse than criterion *ImpAv* that monitors the improvement averaged over all individuals. For many functions no convergence rates of 100% could be found so these criteria cannot be considered as reliable, and furthermore choosing of parameter settings is not easy.

A third improvement-based criterion *NoAcc* was based on the number of generations g in which no trial vector has been accepted. Problems occurred because parameter g has to be different for varying functions in order to give good performance. Particularly, the missing scalability to values smaller than 1 may lead to high additional computational effort for certain optimization problems. If suitable settings for g can be found, *NoAcc* shows reliable performance.

It should be noted that although a large test set that contains a broad range of functions was used here, still optimization problems may exist for which the obtained conclusions do not hold. However, it is expected that at least similar behavior will be found in these cases.

Most of the stopping criteria that are described in this work can also be used for other evolutionary algorithms besides DE but it has to be noted that the performance may be different as it was shown for Particle Swarm Optimization in former work [20, 22]. It would be interesting to try the stopping criteria also for other optimization algorithms in future work.

The stopping criteria presented in this chapter are mainly designated for real-world problems with unknown optimum because for optimization problems with known optimum other good alternatives exist for terminating an optimization run. Only single-objective optimization was addressed yet but real-world problems often contain multiple objectives, thus future work must include the development of reliable stopping criteria for multi-objective optimization. Unfortunately, the situation is more difficult in multi-objective optimization because usually optimization goals are contradicting. Thus, not one single optimal point exists but several trade-off solutions which are usually called the Pareto-optimal front [17].

As a consequence, it is not easy to detect convergence even if the Pareto-optimal front is known. It was shown here that distribution-based criteria provide the best results for single-objective optimization, but for multi-objective problems with unknown Pareto-optimal front this concept will be generally not transferable because usually multiple Pareto-optimal solutions exist. Monitoring

the movement of individuals may also lead to false conclusions because the individuals may still move along the Pareto-optimal front after the population has converged to it. If the improvement of individuals should be taken as basis for stopping criteria, the problem arises how to define improvement in the presence of several objectives.

Apart from the mechanisms presented in this work, there are some concepts inherent in multi-objective optimization which may also possibly be exploited for the definition of stopping conditions. For instance, a stopping criterion based on the observation of crowding distance was tested in [33]. A further possibility is to monitor the improvement of performance measures like hypervolume [17]. Because multi-objective optimization is a research topic that is currently discussed intensively in the evolutionary algorithms community and its importance can still be expected to grow in the following years, this is an interesting field for future work.

References

1. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA (1995)
2. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A Comparative Study of Differential Evolution Variants for Global Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (2006)
3. Huang, V., Qin, A., Suganthan, P.: Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (2006)
4. Takahama, T., Sakai, S.: Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 308–315 (2006)
5. Tasgetiren, M.F., Suganthan, P.: A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 340–347 (2006)
6. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
7. Brest, J., Žumer, V., Maučec, M.S.: Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 919–926 (2006)
8. Kukkonen, S., Lampinen, J.: Constrained Real-Parameter Optimization with Generalized Differential Evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 911–918 (2006)
9. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: Modified Differential Evolution for Constrained Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 332–339 (2006)
10. Zielinski, K., Laur, R.: Constrained Single-Objective Optimization Using Differential Evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 927–934 (2006)

11. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello Coello, C.A., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore (2006)
12. Onwubolu, G.C.: Optimizing CNC Drilling Machine Operations: Traveling Salesman Problem-Differential Evolution Approach. In: Onwubolu, G.C., Babu, B. (eds.) New Optimization Techniques in Engineering, pp. 537–566. Springer, Heidelberg (2004)
13. Lampinen, J., Storn, R.: Differential Evolution. In: Onwubolu, G.C., Babu, B. (eds.) New Optimization Techniques in Engineering, pp. 123–166. Springer, Heidelberg (2004)
14. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
15. Gämperle, R., Müller, S.D., Koumoutsakos, P.: A Parameter Study for Differential Evolution. In: Grmela, A., Mastorakis, N. (eds.) Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, pp. 293–298. WSEAS Press (2002)
16. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.D.: Parameter Study for Differential Evolution Using a Power Allocation Problem Including Interference Cancellation. In: Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 6748–6755 (2006)
17. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
18. Deb, K., Pratap, A., Agrawal, S., Meyarian, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
19. Onwubolu, G.C.: Differential Evolution for the Flow Shop Scheduling Problem. In: Onwubolu, G.C., Babu, B. (eds.) New Optimization Techniques in Engineering, pp. 585–611. Springer, Heidelberg (2004)
20. Zielinski, K., Peters, D., Laur, R.: Stopping Criteria for Single-Objective Optimization. In: Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore (2005)
21. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.D.: Examination of Stopping Criteria for Differential Evolution based on a Power Allocation Problem. In: Proceedings of the 10th International Conference on Optimization of Electrical and Electronic Equipment, Braşov, Romania, vol. 3, pp. 149–156 (2006)
22. Zielinski, K., Laur, R.: Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimization Algorithm. Informatica 31(1), 51–59 (2007)
23. Schwefel, H.P.: Evolution and Optimum Seeking. John Wiley and Sons, Chichester (1995)
24. van den Bergh, F.: An Analysis of Particle Swarm Optimizers. PhD thesis, University of Pretoria (2001)
25. Syrjakow, M., Szczerbicka, H.: Combination of Direct Global and Local Optimization Methods. In: Proceedings of the IEEE International Conference on Evolutionary Computing (ICEC 1995), Perth, WA, Australia, pp. 326–333 (1995)
26. Espinoza, F.P.: A Self-Adaptive Hybrid Genetic Algorithm for Optimal Groundwater Remediation Design. PhD thesis, University of Illinois (2003)
27. Vasconcelos, J., Saldanha, R., Krähenbühl, L., Nicolas, A.: Genetic Algorithm Coupled with a Deterministic Method for Optimization in Electromagnetics. IEEE Transactions on Magnetics 33(2), 1860–1863 (1997)

28. Zaharie, D., Petcu, D.: Parallel Implementation of Multi-Population Differential Evolution. In: Proceedings of the 2nd Workshop on Concurrent Information Processing and Computing (2003)
29. Babu, B.V., Angira, R.: New Strategies of Differential Evolution for Optimization of Extraction Process. In: Proceedings of International Symposium & 56th Annual Session of IIChE (CHEMCON 2003), Bhubaneswar, India (2003)
30. Mezura-Montes, E., Coello Coello, C.A.: A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems. IEEE Transactions on Evolutionary Computation 9(1), 1–17 (2005)
31. Mezura-Montes, E., Coello Coello, C.A.: What Makes a Constrained Problem Difficult to Solve by an Evolutionary Algorithm. Technical report, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Mexico (2004)
32. Michalewicz, Z., Schoenauer, M.: Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation 4(1), 1–32 (1996)
33. Rudenko, O., Schoenauer, M.: A Steady Performance Stopping Criterion for Pareto-based Evolutionary Algorithms. In: Proceedings of the 6th International Multi-Objective Programming and Goal Programming Conference, Hammamet, Tunisia (2004)