An Extended Mutation Concept for the Local Selection Based Differential Evolution Algorithm

Jani Rönkkönen
Lappeenranta University of Technology
Department of Information Technology
P.O.Box 20, FI-53851, Lappeenranta, Finland
jani.ronkkonen@lut.fi

Jouni Lampinen
Lappeenranta University of Technology
Department of Information Technology
P.O.Box 20, FI-53851, Lappeenranta, Finland
jouni.lampinen@lut.fi

ABSTRACT

A new mutation concept is proposed to generalize local selection based Differential Evolution algorithm to work in general multi-modal problems. Three variations of the proposed method are compared with classic Differential Evolution algorithm using a set of five well known test functions and their variants. The general idea of the new mutation operation is to divide the mutation into two parts: the local and global mutation. The global mutation works as a migration operator allowing the algorithm perform global search efficiently, while the local mutation improves the efficiency of local search.

The results show that the concept of global mutation is able to generalize the good performance of local selection based Differential Evolution from convex uni-modal functions to general non-convex and multi-modal problems. Among the tested functions, the new method was able to outperform the classic Differential Evolution in all but one. A limited analysis of the effects of control parameters to the performance of the algorithm is also done.

Categories and Subject Descriptors

G.1.6 [**Numerical analysis**]: Optimization, Global optimization

General Terms

Algorithms

Keywords

Differential Evolution, Selection, Mutation

1. INTRODUCTION

In [7], a comparison between the original *Differential Evolution* algorithm, which uses *global selection* (DEGS) and a new, fundamentally different *Differential Evolution* algorithm using *local selection* (DELS), was performed. The results suggested DELS to clearly outperform DEGS in convex uni-modal functions as the dimensionality of the problem is increased. A strong dependence between the population size (*NP*) and the mutation scale factor (F) in DEGS was observed, which prevents the effective use of small F in conjunction with small *NP*. The results showed that values of F smaller than 0.5 will not increase the algorithms performance because of the required increase in *NP* to prevent premature convergence. DELS has no such dependence and allows the effective use of smaller F values which are effective in high dimensional problems. However, the results obtained for Rosenbrock's function verified that the simple DELS approach works only for convex uni-modal functions and an additional migration mechanism is needed to generalize DELS for more complicated problems.

In this paper a new mutation concept is introduced to expand the DELS algorithm to work in multi-modal problems. An either-or concept [8, p.117] is used to divide the mutation operation into two parts: the local- and global mutation. Three variations of the new mutation operation used with DELS, are compared with DEGS using five well known test problems and their variants.

GECCO'07, July 7-11, 2007 London, England, United Kingdom.

2. DIFFERENTIAL EVOLUTION

The *Differential Evolution* (DE) algorithm [14, 8, 4, 6] was introduced by Storn and Price in 1995 [13]. The design principles of DE were simplicity, efficiency, and use of floating-point encoding instead of binary numbers.

2.1 The Classic DE

In this paper the term classic DE refers to the DE/rand/1/bin scheme. DE starts from a random initial population. In each generation g, DE goes through each D dimensional target vector $\vec{x}_{i,g}$ of the population and creates a corresponding trial vector $\vec{u}_{i,g}$ using mutation (eq. 1) and crossover (eq. 2) operations. The difference between two randomly chosen population vectors $\left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right)$ defines the magnitude and direction of the mutation. This makes the mutation operation self adaptive because the average mutation step length decreases as the population converges.

$$v_{j,i,g} = x_{j,r_0,g} + F \cdot \left(x_{j,r_1,g} - x_{j,r_2,g}\right) \tag{1}$$

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } rand[0,1] \leq CR \vee j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{2}$$

The control parameters for classic DE are crossover rate *CR*, mutation factor F and the population size *NP*. $F \in]0,1+]$ is a scaling factor for the mutation step length and affects the population's convergence speed. $CR \in [0,1]$, controls the crossover by determining the average number of parameters the trial vector $\vec{u}_{i,g}$ inherits from the mutated vector $\vec{v}_{i,g}$. An important aspect to consider when using crossover is that the smaller the used value of *CR*, the less rotationally invariant the search becomes [8]. When the crossover

is disabled ($CR = 1$), the search becomes rotationally invariant. To prevent crossover from duplicating the objective vector, $\vec{u}_{i,g}$ always inherits the parameter with randomly chosen index j_{rand} from $\vec{v}_{i,g}$. Indices r_1, r_2, and r_0 are mutually different and drawn from the set of population indices. $rand[0,1]$ is a random number drawn anew for each round from uniform distribution in the range [0,1].

At the end of each generation the selection (eq. 3) operation is performed comparing each trial vector $\vec{u}_{i,g}$ to the corresponding target vector $\vec{x}_{i,g}$. If the trial has equal or lower cost, it replaces the target vector.

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g} & \text{if } f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g}) \\ \vec{x}_{i,g} & \text{otherwise} \end{cases} \quad (3)$$

The classic DE is described in algorithm 1.

Algorithm 1 DE/rand/1/bin

```
Initialize population, g = 1
while termination criterion not met do
  for i = 1; i ≤ NP; i = i + 1 do
    Randomly pick r0, r1, r2 ∈ {1, 2, ..., NP}, r0 ≠ r1 ≠ r2 ≠ i)
    Randomly pick j_rand ∈ {1, 2, ..., D}
    for j = 1; j ≤ D; j = j + 1 do
      Perform mutation using equation 1
      Perform crossover using equation 2
    end for
  end for
  for i = 1; i ≤ NP; i = i + 1 do
    Perform selection using equation 3
  end for
  g = g + 1
end while
```

2.2 Global and Local Selection

The *global selection* in classic DE refers to the fact that the base vector $\vec{x}_{r_0,g}$ is different than the target vector $\vec{x}_{i,g}$. This allows the under performing population members to be replaced by variants of the population's better solutions.

In *local selection* the equation 1 is modified so that the target and base vectors are the same ($r_0 = i$). Now each population member is always compared to it's own mutant in the selection phase. In effect local selection partitions the population into *NP* niches which evolve in isolation.

The important difference between local and global selection is that DELS is able to maintain the population diversity more efficiently compared to DEGS. Especially when F gets smaller in DEGS, the takeover time for the best solutions to take over the population will decrease. DELS, on the other hand, behaves quite the contrary: As the F decreases, the takeover time increases and approaches infinity as the F approaches zero. In practice this means that DEGS requires increase in population size to compensate the decreasing value for F to prevent premature convergence, which in turn decreases the convergence speed. For more information, refer to [7].

Experimental results for mutation only DE in [7] demonstrated that DEGS is unable to take advantage of the values of F below 0.5 efficiently. On the other hand the optimal value of F for DELS was perceived to scale roughly according to $F = 1.3/\sqrt{D}$ in convex uni-modal problems. The experiments confirmed that as the dimensionality of the problems were increased the performance of DEGS started to drop more rapidly compared to DELS. However, the good results did not generalize in Rosenbrock's function (9) where DELS performed poorly compared to DEGS. It seems that the isolated niches of DELS may form a problem in more complicated problems. Especially in multi-modal problems, the algorithm relies too much on the initial population, because good solutions cannot migrate efficiently. It would be possible to use the traditional uniform crossover operation also with DELS, but it would make the algorithm rotationally variant, which is not desirable. For this reason a new migration mechanic is needed to generalize DELS concept for multi-modal problems.

2.3 Modified Algorithm

The basic idea of the proposed algorithm is to divide the mutation operation into two parts: the local- and global mutation using either-or concept [8, p.117]. The local part is simply the normal DELS mutation with optimized F for convex uni-modal problems ($F = 1.3/\sqrt{D}$). This part is intended to import the algorithm a good behavior in uni-modal problems and increases the efficiency of local search in multi-modal cases. The local mutation is described by equation 4.

$$\vec{u}_{i,g} = \vec{x}_{i,g} + 1.3/\sqrt{D} \cdot \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) \quad (4)$$

For the global scale, a method is needed to allow more efficient migration of the solutions to new areas of the search space and escape local optima. In this paper we propose three variants of global mutation concept for this task. In all cases Gaussian probability distribution function is used to provide variance for the mutation factor, to increase the pool of potential trials, and avoiding the stagnation problem [5].

Because the scale of the mutation is now global, the expectation for the mutation step length is always the unscaled length of the differential to represent the actual scale of the population.

2.3.1 Dither

The simplest version of the global mutation uses the dither scheme [8, p.80]. The mutation differential is multiplied with a random number A, drawn anew for each mutation from Gaussian distribution, with expectation 1 and standard deviation σ. Dither randomizes the length of the differential, but does not affect its direction: the mutation can only reach points in a line along the differential vector, not potentially anywhere in the search space. This means that the process retains the rotation invariance of the mutation.

Equation 5 describes the dither version of the global mutation.

$$\vec{u}_{i,g} = \vec{x}_{i,g} + A \cdot \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) \quad (5)$$

2.3.2 Jitter

In jitter scheme [8, p.80] the mutation differential is multiplied component-wise with a D dimensional vector $\vec{B}$ of random numbers, drawn anew for each mutation from Gaussian distribution, with expectation 1 and standard deviation σ. The fundamental difference to jitter is that now each parameter of the differential vector is multiplied with a different number. In addition to scaling the length, the jitter mutation also changes the direction of the differential making it possible for the mutation to potentially reach any point of the search space with positive probability as long as the differential is not zero.

Equation 6 describes the jitter version of the global mutation.

$$\vec{u}_{i,g} = \vec{x}_{i,g} + \vec{B} \cdot \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) \quad (6)$$

The downside of jitter is that it is not a rotationally invariant process, but tends to turn the differential towards coordinate axes

which are closest the differential's direction as can be seen from figure 1. The problem becomes worse as larger values for σ are used.

2.3.3 Additive Jitter

To achieve rotational invariance, an additive version of jitter (ajitter) can be used. Now the jitter-like effect is acquired by adding a vector $\vec{C}$ of length D to the differential. The $\vec{C}$ is constructed by drawing random numbers from Gaussian distribution with expectation 0 and using the length of the differential scaled by $\sigma/\sqrt{D}$ as the standard deviation as described in equation 7. New $\vec{C}$ is constructed for each mutation operation.

$$C_j = N\left(0, |\vec{x}_{r_1,g} - \vec{x}_{r_2,g}| \cdot \sigma/\sqrt{D}\right), j = 1,2,\ldots,D \qquad (7)$$

The scaling by the length of unit vector keeps the length of $\vec{C}$ independent of the dimension of the problem and σ represents the proportional length of the differential. The scaling is necessary to allow ajitter to remain self-adaptive to the changes of population.

Like jitter, ajitter mutation may potentially reach any point of the search space with positive probability as long as the differential is not zero. Ajitter is also rotationally invariant process because the construction of $\vec{C}$ is not biased in any way. Equation 8 describes the additive jitter version of the global mutation.

$$\vec{u}_{i,g} = \vec{x}_{i,g} + \left(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}\right) + \vec{C} \qquad (8)$$

Figure 1 displays the distributions acquired by generating 400 random points for three differently aligned differentials of equal length using dither, jitter and ajitter versions of the global mutation with $\sigma = 0.1$. As can be seen dither and ajitter always produce similar distribution, but with jitter the shape of the distribution is affected by the rotation of the differential.

2.3.4 The Algorithm

The proposed version of DELS is described in algorithm 2. Compared to classic DE two new parameters are required, *PX* and σ. On the other hand parameters F and *CR* from classic DE are not required anymore and the total amount of parameters does not change. The frequency to use either local or global mutation is controlled by *PX* such that with $PX = 0$ global mutation is never used and with $PX = 1$ global mutation is always used. σ basically controls how large role the differential has in the global mutation. with $\sigma = 0$ the differential solely defines the mutation and increasing the value takes the global mutation towards a random search.

3. TEST FUNCTIONS

A five well known test problems and their variants were used to test the suggested three variants of the DELS scheme and for comparison similar tests were performed with classic DE. In the following the problems are introduced briefly. More detailed information of the functions is available from `http://www.it.lut.fi/ip/evo/functions/functions.html`

The first of the used functions, the Generalized Rosenbrock's function (9) is non-separable and uni-modal with $D < 4$. In [12] Shang and Qiu prove that there exists one local minimum in addition to the local minimum with $4 \leq D \leq 30$ and leave open the possibility of additional local minima with higher dimensions. The function is unconstrained, but we initialized the population in range $-30 \leq x_i \leq 30 \quad i = 1,\ldots,D$.

$$f_4(\vec{x}) = \sum_{i=1}^{D-1}\left(100({x_i}^2 - x_{i+1})^2 + (1-x_i)^2\right) \qquad (9)$$

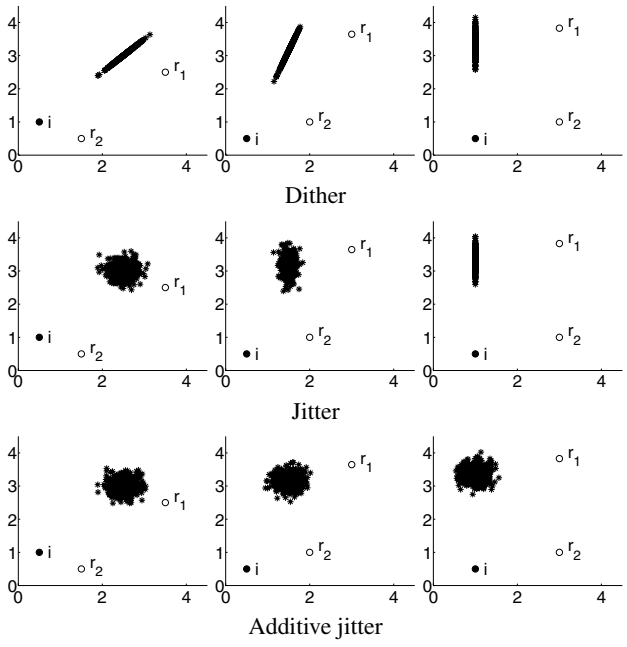

Figure 1: Graphical presentation of 400 random points produced by dither (top row), jitter (middle row) and additive jitter (bottom row) mutation operations with $\sigma = 0.1$. The points i,r_1 and r_2 represent the end points of vectors $\vec{x}_i$, $\vec{x}_{r_1}$ and $\vec{x}_{r_2}$ starting from point $0,0$ and the $r_1 - r_2$ is the differential which is added to i.

Second function was the generalized Rastrigin function (10), which is highly multi-modal but separable function with optimum in the middle. A randomly rotated (11) and randomly rotated, miss-scaled (12) versions of the Rastrigin were also used to change the function to non-separable form. Randomly rotated means that for each run the rotation angle changes randomly. All versions of the Rastrigin function are unconstrained. For initialization the range $-5.12 \leq x_i \leq 5.12 \quad i = 1,\ldots,D$ was used. In addition comparable runs were performed using $-0.12 \leq x_i \leq 10.12 \quad i = 1,\ldots,D$ as the initialization range for the f_5 to see if the placement of optimum has effect on the performance of the algorithms.

$$f_5(\vec{x}) = 10D + \sum_{i=1}^{D}\left({x_i}^2 - 10\cos(2\pi x_i)\right) \qquad (10)$$

$$f_{5.1}(\vec{y}) = 10D + \sum_{i=1}^{D}\left({y_i}^2 - 10\cos(2\pi y_i)\right) \qquad (11)$$

Where $\vec{y} = [\vec{o}_1,\ldots,\vec{o}_D]^T\vec{x}$ and the matrix $[\vec{o}_1,\ldots,\vec{o}_D]^T$ implements an angle preserving linear transformation of $\vec{x}$ [2].

$$f_{5.3}(\vec{y}) = 10D + \sum_{i=1}^{D}\left((10^{\frac{i-1}{D-1}}{y_i}^2 - 10\cos(2\pi \cdot 10^{\frac{i-1}{D-1}}y_i)\right) \qquad (12)$$

The third used function was the normalized Schwefel function (13). Schwefel function is a separable multi-modal function with optimum close to the border. The search area was constrained in range $-512 \leq x_i \leq 512, \quad i = 1,\ldots,D$. A non-separable version was generated similarly as in Rastrigin by randomly rotating the

Algorithm 2 Modified DELS Algorithm

```
Initialize population, g = 1
while termination criterion not met do
  for i = 1; i ≤ NP; i = i + 1 do
    Randomly pick r1, r2 ∈ {1, 2, ..., NP}, r1 ≠ r2)
    if rand[0, 1] < PX then
      Perform global mutation using equation 5, 6 or 8
    else
      Perform local mutation using equation 4
    end if
  end for
  for i = 1; i ≤ NP; i = i + 1 do
    Perform selection using equation 3
  end for
  g = g + 1
end while
```

function (14). Also the constraints were rotated to be in range $-512 \leq y_i \leq 512, \quad i = 1, \ldots, D$ in this version.

$$f_6(\vec{x}) = \frac{\sum_{i=1}^{D} -x_i \sin(\sqrt{|x_i|})}{D} \tag{13}$$

$$f_{6.1}(\vec{y}) = \frac{\sum_{i=1}^{D} -y_i \sin(\sqrt{|y_i|})}{D} \tag{14}$$

The fourth function was the Whitley's function (15) which is a unconstrained non-separable multi-modal function. The function combines a very steep overall slope with highly multi-modal area around the optimum, which is located at $x_i = 1, \quad i = 1, \ldots, D$. The initialization range of $-5.12 \leq x_i \leq 5.12 \quad i = 1, \ldots, D$ was used.

$$f_8(\vec{x}) = \sum_{i=1}^{D} \sum_{j=1}^{D} \Big(\frac{\big(100(x_i^2 - x_j)^2 + (1 - x_j)^2\big)^2}{4000} - \cos\big(100(x_i^2 - x_j)^2 + (1 - x_j)^2\big) + 1 \Big) \tag{15}$$

The last function was the normalized Rana's function (16) which is a very hard multi-modal non-separable function with optimum close to the border. The search area was constrained in range $-520 \leq x_i \leq 520 \quad i = 1, \ldots, D$

$$f_{12}(\vec{x}) = \Bigg(\sum_{i=1}^{D} \Big(x_i \sin\big(\sqrt{|x_j + 1 - x_i|}\big) \cos\big(\sqrt{|x_j + 1 + x_i|}\big) + (x_j + 1) \cos\big(\sqrt{|x_j + 1 - x_i|}\big) \sin\big(\sqrt{|x_j + 1 + x_i|}\big) \Big) \Bigg) / D \tag{16}$$

Where $j = (i + 1) \mod D$.

4. TEST SETUP

The algorithmic performance was measured by calculating the average number of function evaluations (NFE) required to find the optimum with accuracy ε from 100 independent runs. For f_4 a value of $\varepsilon = 10^{-6}$ and for other functions a value of $\varepsilon = 10^{-2}$ was used. A run was counted success if it found the optimum before reaching defined maximum number of function evaluations, NFE_{max}. The NFE_{max} was scaled in proportion to D and NP such that for f_4 a value of $NFE_{max} = 1000D \cdot NP$, for f_{12} a value of $NFE_{max} = 20000D \cdot NP$ and for the other functions a value of $NFE_{max} = 3000D \cdot NP$ was used. The fraction of successful runs (s) from the performed 100 independent runs was calculated and used to estimate the probability of success. Using this a success performance (Sp) was calculated by dividing the NFE by s. Because Sp combines the speed of algorithm and the estimated percentage of successes in one number, it is a useful performance measure. It would consider two methods to be equally effective if other solves the problem twice as often but uses also on average twice the time as the other. However, when s is very small the value of Sp is not very informative since a small change in s results in huge change in Sp and the variance in the s becomes an issue. For this reason the Sp values are calculated only for cases, when the problem is considered 'solved'. In this paper a value of $s > 5/100$ was required to categorize the problem 'solved'. A term 'occasional success' is used to refer the runs for which $s \leq 5/100$.

4.1 Dimensionality

Our goal was to measure how the algorithms perform with different problems as the dimensionality is increased. The function f_4 was sampled $D = 2, 6, \ldots, 50$, f_{12} was sampled $D = 2, 3, \ldots, 5$ and other problems were sampled $D = 2, 4, \ldots, 20$. However to save computation time, the higher dimensional runs were not performed, if the algorithm had already failed to solve a lower dimensional version of the problem with all tested parameter combinations.

4.2 Used Control Parameter Setup

Because the test setup was very time consuming, not all possible parameter combinations were tested. For classic DE the crossover was disabled using always $CR = 1$ to achieve rotation invariance. For F, values of $0.5, 0.6 \ldots 1$ were tested in all cases. The values $F < 0.5$ were left out, because of the findings in [7], which suggest that DEGS can not benefit from small values of F.

Based on earlier experience of DE (for example [10] and [3]), the increase in population size decreases the speed of algorithm linearly after reaching a minimum size which is required to solve the problem. Also the required population size is proportional to the problem dimension. A value of $NP = 9D$ was tested with all methods and functions because it was close the optimum NP value identified for Rosenbrock's function (f_4) in [7]. Additionally for classic DE a value of $NP = 2.4D^{1.5}$, and for DELS based methods, a value of $NP = 3D$ were tested on all problems. These values are close to the optimum NP values identified for convex uni-modal problems for DEGS and DELS in [7] and can be considered the lower bound for viable NP values for any harder problems. Because DEGS typically requires larger population sizes compared to DELS, due to larger selection pressure, a value of $NP = 20D$ was also tested with classic DE in all other problems except the f_4. Lastly with the f_{12} all methods were tested also with values $NP = 20D$ and $NP = 40D$.

For the DELS based methods, values of $\sigma = 0.1$ and $\sigma = 0.01$ were tested for all problems. $\sigma = 0$ was also tested to make sure the use of gaussian distribution based mutation really offers something to the algorithm. Additionally a value of $\sigma = 0.5$ was occasionally used, to get an impression how much larger σ values would perform and to make sure the used magnitude of σ is not too small altogether. These tests were not used, however, when determining the best Sp values, because a complete set of results were not generated using $\sigma = 0.5$. The selected values are somewhat based on the author's earlier experiences using jitter with DEGS [11], but can not be considered to be optimal values, rather educated queses, and more research is required in future to identify the optimal range for the values of σ. Lastly, the PX value was scaled from $0.2, 0.4, \ldots, 1$ for all DELS based methods and all problems. In addition a value $PX = 0$ was tested to make sure that the global mutation part really is needed and useful in tested problems.

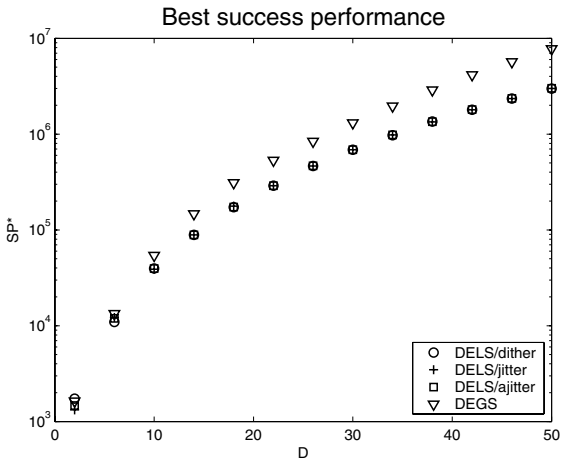

Figure 2: Best achieved Sp values for f_4 (Rosenbrock)

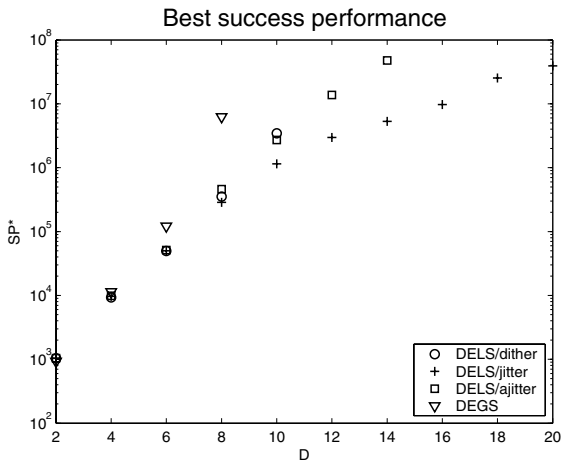

Figure 3: Best achieved Sp values for f_5 (Rastrigin) with centered initialization

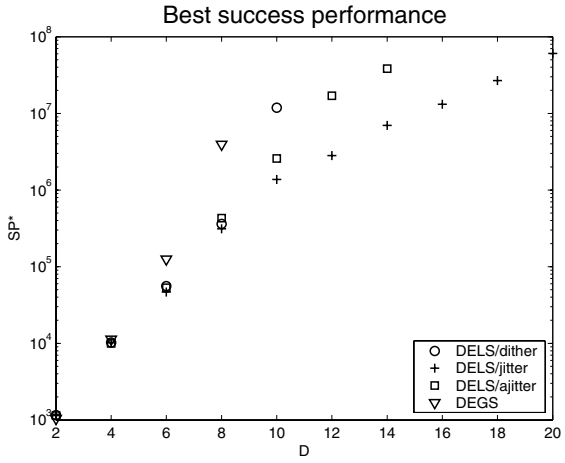

Figure 4: Best achieved Sp values for f_5 (Rastrigin) with non-centered initialization

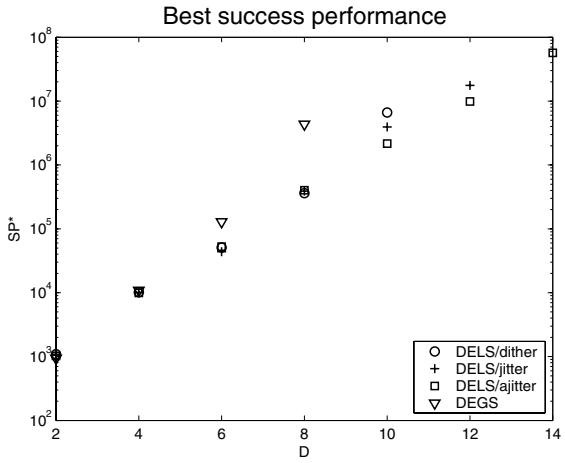

Figure 5: Best achieved Sp values for $f_{5.1}$ (Rotated Rastrigin)

5. RESULTS

Success performance was used as the main factor when deciding how well the algorithms performed and no other figures have been included into this paper due to lack of space. More comprehensive set of figures, can be found from: `http://www.it.lut.fi/ip/evo/`.

5.1 Rosenbrock

All tested algorithms were able solve the Rosenbrock's function regardless of the dimension. As can be seen from figure 2 all DELS based methods performed almost similarly and were clearly faster than DEGS. The relative difference between DELS and DEGS grows larger as the dimension of the problem increases.

5.2 Rastrigin

The results for unrotated Rastrigin are presented in figures 3 and 4. Changing the initialization range of population from centered to non-centered had no considerable effect on any of the methods. DEGS was able to solve (attain $s > 5$) when $D \leq 8$. For the DELS methods, dither was able to solve the problem when $D \leq 10$, ajitter when $D \leq 14$ and jitter with all tested dimensions. For the Sp, DEGS was again the slowest while jitter was able to outperform the dither and ajitter when $D > 8$.

The situation changes, when the Rastrigin function is rotated, which makes the function non-separable. As can be seen from figures 5 and 6, miss-scaling did not have significant effect on the performance of any of the algorithms. Also the rotation had negligible effect on the performance of DEGS, dither and ajitter. However, the results of jitter decrease notably and are now comparable to those of ajitter.

5.3 Schwefel

Also in Schwefel's function, DELS outperformed DEGS, as can be seen from figures 7 and 8. DEGS was able to solve the problem when $D \leq 14$ and the rotation again had only small effect on the performance. Ajitter had problems with the higher dimensional versions and was unable to solve (other than occasional successes) the unrotated version with $D = 20$ and the rotated version with $D > 16$. Dither and jitter were able to solve the problem with all tested dimensions. DEGS was the slowest, when comparing Sp. Again jitter outperformed the other DELS algorithms in the unrotated versions, but in the rotated version dither was the fastest.

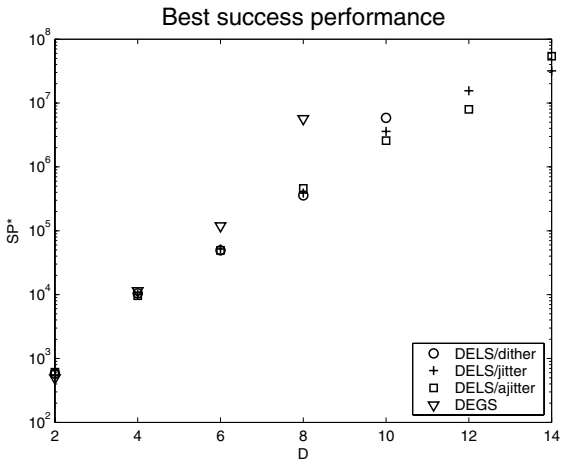

Figure 6: Best achieved Sp values for $f_{5.3}$ (Rotated miss-scaled Rastrigin)

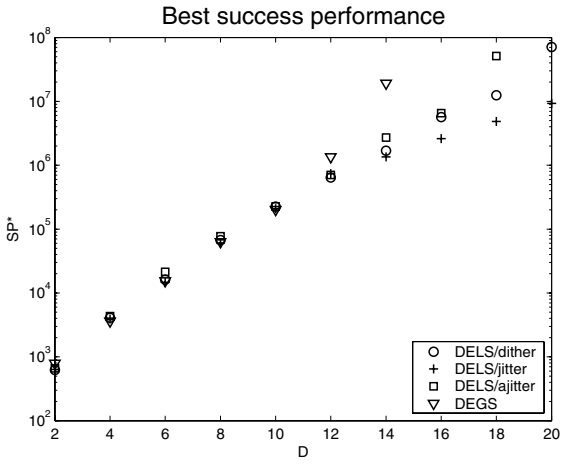

Figure 7: Best achieved Sp values for f_6 (Schwefel)

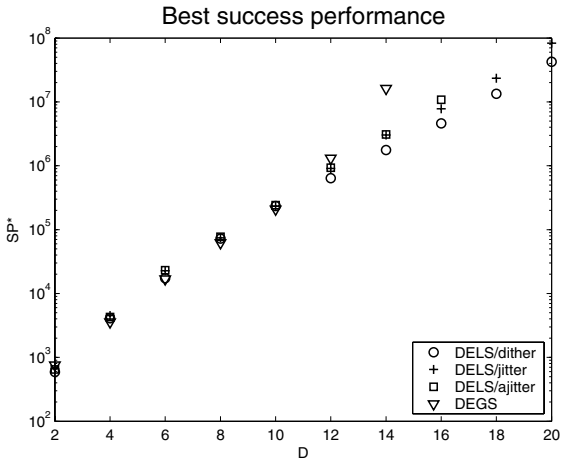

Figure 8: Best achieved Sp values for $f_{6.1}$ (Rotated Schwefel)

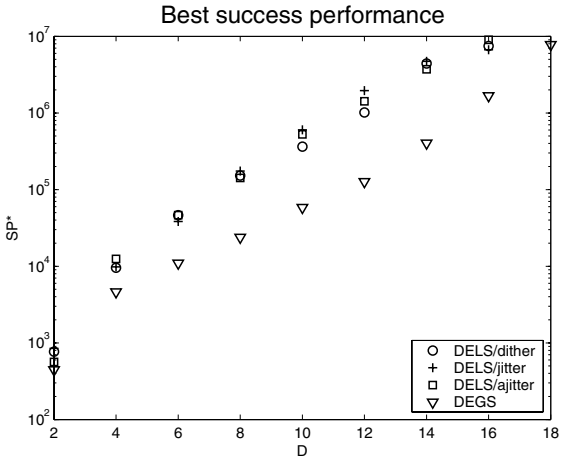

Figure 9: Best achieved Sp values for f_8 (Whitley)

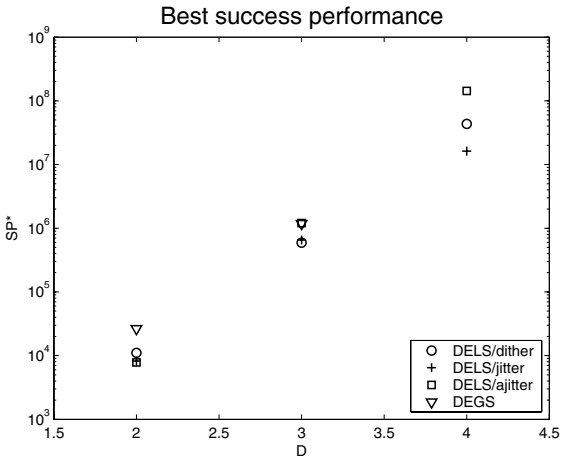

Figure 10: Best achieved Sp values for f_{12} (Rana)

5.4 Whitley

Whitley's function was the only one from the test set where DEGS was able to outperform DELS. DEGS managed to solve the problem when $D \leq 18$ when all DELS methods only managed to solve it when $D \leq 16$ (Still all methods scored occasional successes with all tested dimensions). When comparing the Sp values from figure 9, DEGS was clearly faster than DELS regardless of the dimension. The differences between performance of the different DELS methods were rather small.

5.5 Rana

Rana's function was clearly the hardest on the test setup. DEGS was able to solve it only when $D = 2,3$ while all DELS methods were able to solve it when $D = 2,3,4$. Dither was the only method to get occasional successes when $D = 5$ In terms of Sp (figure 10), DEGS was the slowest when $D = 2$ and shared the slowest performance with ajitter when $D = 3$. Among the DELS methods ajitter was the slowest and jitter the fastest.

5.6 Control Parameters

The value of $PX = 0$ (Global mutation disabled) performed very poorly in all tested cases throughout the test set. In Rosenbrock

all DELS methods behaved rather similarly and small values performed best. Increasing the value up to 0.6 did not have a significant effect in most cases, but using higher values started to decrease the performance of the algorithms. In the other functions *PX* did not have a clear optimum value. For dither, the optimum *PX* typically got smaller as the dimensionality of the problem increased. For jitter and ajitter the trend was not as clear.

In most cases smaller values for σ increased the speed, but decreased the percentage of successful runs. In the Rosenbrock's function small σ values performed generally well and it was the only tested function where the value of $\sigma = 0$ was able to compete in performance for the larger values. In Rastrigin and Schwefel value of $\sigma = 0.01$ performed well with small D, but with increasing dimensionality, $\sigma = 0.1$ gained the upper hand (except for ajitter in Schwefel) because of increased s. In Whitley's function dither performed well with $\sigma = 0.1$, but for jitter and ajitter the best value for σ was more often 0.01 than 0.1.

The best performing population sizes for all DELS methods on Rosenbrock were $NP = 3D$ for all dimensions and for DEGS they followed $NP = 2.4D^{1.5}$ as was expected. For Rastrigin all DELS methods performed best with $NP = 9D$, which was also the best population size for DEGS when $D \leq 4$. With increased dimension DEGS required $NP = 20D$. In Schwefel all DELS methods performed best with $NP = 3D$ until $D \leq 8$ and after that $NP = 9D$. DEGS used the $NP = 2.4D^{1.5}$ or $NP = 9D$ when $D \leq 12$ and $NP = 20D$ with $D = 14$. For Whitley, the best population size for DEGS was mostly $NP = 9D$, except for the $NP = 2.4D^{1.5}$ with $D = 6$ and $NP = 20D$ with $D = 18$. DELS methods used $NP = 3D$ for small dimensions and $NP = 9D$ for larger. Dither swapped at $D = 6$, jitter at $D = 8$ and ajitter at $D = 12$. In Rana, DEGS performed best with $NP = 40D$ while DELS methods used $NP = 3D$ for $D = 2$, $NP = 20D$, for $D = 3$ (except jitter, which performed best with $NP = 9D$) and $NP = 40D$ with higher dimensions.

6. DISCUSSION

When comparing the results, DELS was able to outperform DEGS in all but the Whitley's function. It seems that the Whitley's combination of very steep overall slope and highly multi-modal area around the optimum favor the greedier global selection.

In Rosenbrock's function, all DELS versions performed almost equally and were able to clearly outperform DEGS. This clearly demonstrates the usefulness of the global mutation operation in DELS, especially when the results are compared to the ones presented in [7] for the same function, where DELS without global mutation was clearly slower in terms of Sp than DEGS and was unable to solve the problem in higher dimensions.

It can be observed from the results of Rastrigin that none of the tested algorithms were sensitive to the miss-scaling or moving the optimum away from the center of the search space. Also the results for Rastrigin and Schwefel functions show that rotation had only a small effect on the performance of DEGS, dither and ajitter, which support the claim that these methods are rotationally invariant. Jitter, on the other hand, performs clearly better in the unrotated cases, which confirms that it exploits the separability of the functions. It is notable, however, that while being able to clearly outperform the dither and ajitter in unrotated versions, jitter is still able to achieve comparable performance in the rotated versions. Of course the used σ values in this paper were rather small and it is possible that optimum values for at least some problems are larger. Since the degree of rotational variance (how strongly the search is biased along the coordinate axis directions) of jitter increases as σ increases, it can be expected that the comparative performance of jitter, will be worse in non-separable problems, if larger σ is used.

From the DELS versions, jitter performed the best overall in terms of Sp. When the separable cases are left out, the differences in performance between the algorithms are quite small. Ajitter had some problems with Schwefel and Rana, but performed well in Rastrigin. Dither on the other hand had problems with Rastrigin, but worked well in Schwefel.

6.1 Analysis of Control Parameters

The poor performance, when using $PX = 0$, was to be expected, because the local mutation alone cannot escape local optima efficiently enough. Disabling the local mutation (using $PX = 1$), performed well in some functions, especially in Rana. Still, in most tested cases smaller *PX* value seemed to find the optimum more reliably, especially when the dimensionality was increased. The local mutation was able to most clearly improve the performance of DELS in Rosenbrock's function. This is well in line with the supposed role of local mutation operation: to improve the algorithm's performance in uni-modal cases and speed up the local search of the niches. It is logical that the benefits were most clearly visible in Rosenbrock, which has only one local optimum in higher dimensions. Because the local mutation seemed to be useful in most problems, while not clearly harmful in any of them, using $PX = 1$ is not recommended for initial guess when using the proposed algorithms. When the extremes were left out, the value of *PX* parameter did not seem to be of critical importance to the performance of the algorithm. For this reason, a safe initial guess for the value would likely be around 0.5.

The results clearly demonstrated that using $\sigma > 0$ was beneficial in multi-modal problems. In most cases the value $\sigma = 0.1$ offered better robustness than smaller values and thus it would probably make a decent initial guess. Still, additional research is required to learn more about the effect of the σ parameter. Generally dither seemed to prefer a bit larger σ values than jitter and ajitter. A possible explanation for this is the fact that dither is able to produce more limited variety of different trials, and needs a bit larger σ value to compensate the loss. The few performed experiments with $\sigma = 0.5$, suggest that especially with dither, larger values than $\sigma = 0.1$ could be useful.

The used population sizes were likely close to optimum for the Rosenbrock, since the *NP* values for DEGS were the optimum values identified in [7]. Also for DELS the used $NP = 3D$ seemed to work nicely and none of the runs performed better with larger population sizes. However, the harder multi-modal problems did not scale similarly, and as dimensionality was increased, often the best performance was achieved with the highest tested population size. For DELS often at the beginning values of $NP = 3D$ performed well, but as the dimensionality increased, $NP = 9D$ was required. This suggests that the optimum population size does not increase linearly with dimension anymore, like in uni-modal problems, but also for DELS the optimum population size will grow faster than that in multi-modal problems. For this reason it is possible that the usage of larger population sizes for the higher dimensional cases would increase the percentage of successes and the success performance. The optimum population size seems to be rather problem dependent and more information is required to give accurate recommendations of the best values for DELS methods.

7. CONCLUSIONS

Since only a limited amount of different parameter combinations were studied for each method, it can not be exclusively claimed that some of the tested methods outperform others even inside the selected function testbed, because with different parameter setup the results may change. Especially, enabling the crossover for the un-

rotated Rastrigin and Schwefel, would definitely increase the performance of DEGS, because it could exploit the separability of the functions. The goal of this paper was, however, to demonstrate that with a suitable migration operator, the concept of local selection based Differential Evolution [7] can be efficiently applied to general multi-modal problems, and that goal was achieved. The performance obtained with DELS using global mutation compared favorably with the performance of DEGS.

The performance differences between different DELS versions were generally smaller than differences between DELS and DEGS. Even though jitter performed overally best in the chosen test setup, the lack of rotation invariance remains a risk in non-separable functions. Since the performance differences between different global mutation methods seem to be small, it will probably not make a significant difference in practice which one is used. Still, using any of the proposed methods seems to increase the performance compared to case without any gaussian mutation.

From the theoretical point of view ajitter is the most interesting, because it is a rotationally invariant process and, along with jitter, able to potentially reach any point on the search space with positive probability as long as the differential is not zero. This means that an algorithm using jitter or ajitter satisfy the conditions for provably convergent algorithm in [9] as long as the whole population has not converged in one spot causing all possible differentials to become zero.

8. FUTURE WORK

The global mutation was shown to be a promising way of generalizing the DELS concept to multi-modal problems. We believe that DELS can be a potential optimization tool especially in problems where the ability to maintain the diversity of population is important. Especially problems with multiple optima which should all be found simultaneously or problems where the optima is changing with time could benefit from DELS. DELS might also be useful in multi-objective problems, since it could prevent the population from converging into a one part of the Pareto frontier giving a better distribution of solutions. On the other hand it is possible that the algorithm is not greedy enough to find a dense distribution of points along the Pareto frontier fast enough, because some of the dominated solutions will live longer, and because the typically larger population sizes used in multi-objective optimization, may favor greedier methods. Also more experimental data is needed both in the performance of DELS in different problems and in the effect of the control parameter's values on the performance of the algorithm. It would be interesting to compare the performance of DELS to other evolutionary methods which rely on Gaussian distribution in generating solutions, like covariance matrix adaptation evolution strategy (CMA-ES) [2] or real coded genetic algorithm using a parent-centric recombination (PCX) [1].

The reason for the inferior performance of DELS in Whitley's function should be examined more closely to determine which of the function's features made it hard for DELS. One possibility to improve the performance of DELS and ease the parameter selection would be to make the parameters σ and *PX* adaptive. It seems rational to assume that larger values for both parameters would be better at the start of the search, while smaller would probably speed up the final convergence. Also a hybrid between DELS and DEGS could be possibly beneficial, where DELS would provide the exploration capabilities at the beginning and DEGS would be used later to speed up the convergence, for example in multi-objective problems.

9. ACKNOWLEDGMENTS

The authors wish to thank Kenneth Price for sharing his vast knowledge and ideas on DELS, which have been of great help. The authors also wish to thank Ville Kyrki for his ideas on the implementation of global mutation.

10. REFERENCES

[1] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395, 2002.

[2] N. Hansen and A. Ostermeier. Completely derandomized self adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[3] S. Kukkonen and J. Lampinen. Constrained real-parameter optimization with generalized differential evolution. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, pages 911–918, Vancouver, BC, Canada, July 2006.

[4] J. Lampinen and R. Storn. *New Optimization Techniques in Engineering*, chapter Differential Evolution, pages 123–166. Springer-Verlag, 2004. ISBN: 3-540-20167.

[5] J. Lampinen and I. Zelinka. On stagnation of the differential evolution algorithm. In *Proceedings of MENDEL 2000, 6th International Conference on Soft Computing*, pages 76–83, Brno, Czech Republic, 7-9 June. ISBN 80-214-1609-2.

[6] K. Price. An introduction to differential evolution. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 79–108. McGrawHill. ISBN 007-709506-5.

[7] K. V. Price and J. I. Rönkkönen. Comparing the uni-modal scaling performance of global and local selection in mutation-only differential evolution algorithm. In *Proceedings of 2006 IEEE World Congress on Computational Intelligence*, pages 7387–7394, Vancouver, Canada, 16-21 July 2006. ISBN 0-7803-9489-5.

[8] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005. ISBN 3-540-20950-6.

[9] G. Rudolph. Convergence of evolutionary algorithms in general search spaces. In *Proceedings of the third IEEE Conference of Evolutionary Computation*, pages 50–54. IEEE Press.

[10] J. Rönkkönen, S. Kukkonen, and J. Lampinen. A comparison of differential evolution and generalized generation gap algorithms. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 9(5), September 2005. ISSN 1343-0130.

[11] J. Rönkkönen and J. Lampinen. On using normally distributed mutation step lenght for the differential evolution algorithm. In *Proceedings of MENDEL 2003, 9th International Conference on Soft Computing*, pages 11–18, Brno, Czech Republic, 4-6 June 2003. ISBN 80-214-2411-7.

[12] Y.-W. Shang and Y.-H. Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.

[13] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuos spaces. Technical Report TR-95-012, ICSI, 1995.

[14] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, pages 341–359, 1997.