ENHANCING DIFFERENTIAL EVOLUTION ALGORITHM FOR SOLVING CONTINUOUS OPTIMIZATION PROBLEMS

Miguel Leon

2016

School of Innovation, Design and Engineering
Abstract

Differential Evolution (DE) has become one of the most important metaheuristics during the recent years, obtaining attractive results in solving many engineering optimization problems. However, the performance of DE is not always strong when seeking optimal solutions. It has two major problems in real world applications. First, it can easily get stuck in a local optimum or fail to generate better solutions before the population has converged. Secondly, its performance is significantly influenced by the control parameters, which are problem dependent and which vary in different regions of space under exploration. It usually entails a time consuming trial-and-error procedure to set suitable parameters for DE in a specific problem, particularly for those practitioners with limited knowledge and experience of using this technique.

This thesis aims to develop new DE algorithms to address the two aforementioned problems. To mitigate the first problem, we studied the hybridization of DE with local search techniques to enhance the efficiency of search. The main idea is to apply a local search mechanism to the best individual in each generation of DE to exploit the most promising regions during the evolutionary process so as to speed up the convergence or increase the chance to escape from local optima. Four local search strategies have been integrated and tested in the global DE framework, leading to variants of the memetic DE algorithms with different properties concerning diversification and intensification. For tackling the second problem, we propose a greedy adaptation method for dynamic adjustment of the control parameters in DE. It is implemented by conducting greedy search repeatedly during the run of DE to reach better parameter assignments in the neighborhood of a current candidate. The candidates are assessed by considering both, the success rate and also fitness improvement of trial solutions against the target ones. The incorporation of this greedy parameter adaptation method into standard DE has led to a new adaptive DE algorithm, referred to as Greedy Adaptive Differential Evolution (GADE).

The methods proposed in this thesis have been tested in different benchmark problems and compared with the state of the art algorithms, obtaining
competitive results. Furthermore, the proposed GADDE algorithm has been applied in an industrial scenario achieving more accurate results than those obtained by a standard DE algorithm.
Sammanfattning


Den föreslagna metoden i denna licentiatavhandling har testats i olika pre-
standamätningar och jämförts med state-of-the-art-algoritmer, med goda resultat. Dessutom har den föreslagna GADE-algoritmen använts i ett industriellt scenario och uppnådde då mer exakta resultat än den med en standard DE-algoritm.
To my family
Acknowledgements

First and foremost, I will like to express my sincere gratitude to my main supervisor Ning Xiong (Mälardalen University) and my two co-supervisors Peter Funk (Mälardalen University) and Yigen Zenlander (ABB FACTS). I have learned a lot from their advices, feedback and guidance, without their support this thesis would not have been possible. I also would like to express the deepest appreciation to my co-authors Magnus Evestedt (Prevas), Francisco Herrera (Granada University) and Daniel Molina (Cadiz University) for their cooperation in the papers. I would like to thank Baran Çürüklü for helping me with the lectures and nice discussions.

I would like to thank all my current and past roommates, Fredrik Ekstrand, Nikola Petrovic, Carl Ahlberg, Lennie Carlén Eriksson, Jonatan Tidare, Martin Ekström, Gregory Koshmak, Jimmie Hagblad and Henrik Johansson. It is a pleasure to share office with you.

I thank the lecturers and professors whom I learned a lot from during meetings, lectures, seminars and PhD courses including Thomas Nolte, Gordana Dodig-Crnkovic, Jan Gustafsson and Mikael Sjödin. You all have been a source of inspiration for me. I thank the administrative staff in particular, Malin Åshuvud, Sofia Jäderén, Carola Ryttersson and Jenny Hagglund for making my live easier.

I would like to thank my friends and colleagues at the division of IFT: Ankica, Branko, Elaine, Farid, Fredrik, Giacomo, Håkan, Hamid, Hamidur, Ivan, Jiaying, Jonas, José-Fernán, Lars, Magnus, Maria Ehn, Maria Lindén, Masoud, Mia, Miguel Angel, Mikael, Mirgita, Mobyen, Per, Per Olov, Sara, Shahina, Shaibal, Torbjörn, Youssef and others. Also I would like to thank Radu, Inmaculada, Filip and others from other divisions.

Last but not least, a huge thank to my family for their support even from the long distance. Also I would like to thank my girlfriend and best friend Laura, without your support this would be impossible.

Miguel Leon, Västerås, 2016
List of Publications

Papers included in the licentiate thesis 1:

Miguel Leon, Ning Xiong, Evolutionary Computation (CEC), 2015 IEEE Congress on, p. 385-392.

Miguel Leon, Magnus Evestedt, Ning Xiong, accepted in Studies in Computational Intelligence, Springer, 2016.


Paper D: A new Differential Evolution Algorithm with Alopex-Based Local Search.
List of Publications

Papers included in the licentiate thesis¹:


¹The included articles have been reformatted to comply with the licentiate layout
Additional papers, not included in the licentiate thesis:


- **Principles and state-of-the-art of engineering optimization techniques.** Ning Xiong, Miguel Leon, Proc. The 7th International Conference on Advanced Engineering Computing and Applications in Sciences, 2013, p. 36-42.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Domain-based evaluation</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Differential Evolution process</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Random mutation with one difference vector</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Mutation on the best individual with one difference vector</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>current to best mutation with one difference vector</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Example of binomial crossover</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of exponential crossover (k=3,L=3)</td>
<td>14</td>
</tr>
<tr>
<td>7.1</td>
<td>DE, parameterized model, and temperature estimation</td>
<td>61</td>
</tr>
<tr>
<td>7.2</td>
<td>Random mutation with one difference vector</td>
<td>62</td>
</tr>
<tr>
<td>7.3</td>
<td>The evolution of the scaling factor F</td>
<td>70</td>
</tr>
<tr>
<td>7.4</td>
<td>The evolution of the distribution center for the crossover probability</td>
<td>71</td>
</tr>
<tr>
<td>8.1</td>
<td>Random mutation strategy</td>
<td>80</td>
</tr>
<tr>
<td>8.2</td>
<td>Current to best mutation strategy</td>
<td>81</td>
</tr>
<tr>
<td>8.3</td>
<td>Current to rand mutation strategy</td>
<td>82</td>
</tr>
<tr>
<td>8.4</td>
<td>Distribution probability</td>
<td>86</td>
</tr>
<tr>
<td>8.5</td>
<td>Function 8 with two dimensions</td>
<td>87</td>
</tr>
<tr>
<td>8.6</td>
<td>Function 9 with two dimensions</td>
<td>88</td>
</tr>
<tr>
<td>8.7</td>
<td>Function 10 with two dimensions</td>
<td>88</td>
</tr>
<tr>
<td>9.1</td>
<td>Random mutation with one difference vector</td>
<td>104</td>
</tr>
</tbody>
</table>
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>Alopex-based Local Search</td>
</tr>
<tr>
<td>CR</td>
<td>Crossover Rate</td>
</tr>
<tr>
<td>DE</td>
<td>Differential Evolution</td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
</tr>
<tr>
<td>ERS</td>
<td>Eager Random Search</td>
</tr>
<tr>
<td>F</td>
<td>mutation Factor</td>
</tr>
<tr>
<td>GADE</td>
<td>Greedy Adaptive Differential Evolution</td>
</tr>
<tr>
<td>LIP</td>
<td>Local Improvement Process</td>
</tr>
<tr>
<td>LLS</td>
<td>Local Improvement process oriented Local Search</td>
</tr>
<tr>
<td>LS</td>
<td>Local Search</td>
</tr>
<tr>
<td>MA</td>
<td>Memetic Algorithm</td>
</tr>
<tr>
<td>RC</td>
<td>Research Contribution</td>
</tr>
<tr>
<td>RQ</td>
<td>Research Question</td>
</tr>
<tr>
<td>XLS</td>
<td>Crossover-based Local Search</td>
</tr>
</tbody>
</table>
# Contents

## I Thesis

1 **Introduction**

- 1.1 Problem Formulation ........................................ 4
- 1.2 Research Questions ......................................... 6
- 1.3 Scientific Contribution ..................................... 6
- 1.4 Research Method ............................................ 8

2 **Background**

- 2.1 Alopex Search ............................................... 9
- 2.2 Differential Evolution ...................................... 10

3 **Related work**

- 3.1 Parameter adaptation in Differential Evolution ............. 15
- 3.2 Local Search in Differential Evolution ....................... 17
  - 3.2.1 LLS in DE ............................................. 17
  - 3.2.2 XLS in DE ........................................... 18
  - 3.2.3 Strategies of Applying LS in DE ......................... 18

4 **Overview of the Papers** ........................................ 21

5 **Conclusion and Future Work** ................................... 25

**Bibliography** .................................................... 27

## II Included Papers

6 **Paper A:**

  Greedy Adaptation of Control Parameters in Differential Evolution
for Global Optimization Problems

6.1 Introduction .................................. 37
6.2 Basic DE .................................. 38
6.3 Adaptation of Control Parameters in DE ........ 40
   6.3.1 Parameter Adaptation in SaDE ............. 40
   6.3.2 Parameter adaptation in JADE .............. 41
   6.3.3 Parameter adaptation in MDE_pBX .......... 42
   6.3.4 Parameter adaptation in SHADE ............ 42
6.4 Greedy Adaptation of DE Parameters ............... 43
   6.4.1 Greedy Search for Parameter Adaptation .... 44
   6.4.2 Evaluation of Candidates for Parameter Assignment . 44
   6.4.3 Greedy Adaptive DE Algorithm ............. 45
6.5 Experiments .................................. 46
   6.5.1 Experimental Settings ..................... 46
   6.5.2 Comparison of GADE with other algorithms ... 49
6.6 Conclusion .................................. 52

7 Paper B:
Adaptive Differential Evolution Supports Automatic Model Calibration in Furnace Optimized Control System

7.1 Introduction .................................. 59
7.2 Problem Formulation .......................... 60
7.3 Differential Evolution ........................ 61
7.4 A New DE Algorithm with Parameter Adaptation .... 63
   7.4.1 Greedy Adjustment Scheme ................. 64
   7.4.2 Adaptive DE Algorithm with Greedy Search ... 66
7.5 Experiments and Results ........................ 66
   7.5.1 Experimental Settings ..................... 68
   7.5.2 Results and Comparison in Problem 1 ....... 68
   7.5.3 Results and Comparison in Problem 2 ....... 69
   7.5.4 Evolution of the DE Parameters .......... 70
7.6 Relevant Work ................................ 71
7.7 Conclusion .................................. 72

8 Paper C:
Differential Evolution Enhanced with Eager Random Search for Solving Real-Parameter Optimization Problems

8.1 Introduction .................................. 77
8.2 Related Work ................................ 78
8.3 Basic DE .................................. 79
8.4 DE Integrated with ERS ........................................... 83
  8.4.1 Eager Random Local Search (ERS) ....................... 83
  8.4.2 The Proposed Memetic DE Algorithm ...................... 86
8.5 Experiments and Results .......................................... 86
  8.5.1 Experimental Settings ........................................ 89
  8.5.2 Performance of the Memetic DE with random mutation
        strategy ...................................................... 90
  8.5.3 Performance of the Memetic DE with Current to Rand
        Mutation Strategy ............................................ 90
  8.5.4 Performance of the Memetic DE with Current to Best
        Mutation Strategy ............................................ 92
8.6 Conclusions ....................................................... 94

9 Paper D:
A new Differential Evolution Algorithm with Alopex-Based Local
Search ................................................................. 99
  9.1 Introduction ..................................................... 101
  9.2 Related Work .................................................... 102
  9.3 Differential Evolution ........................................... 103
  9.4 Differential Evolution with Alopex Local Search .......... 104
    9.4.1 Alopex .................................................... 105
    9.4.2 DEALS .................................................... 106
  9.5 Experiments and Results ........................................ 107
    9.5.1 Experimental Settings ..................................... 107
    9.5.2 Comparing DEALS with Basic DE and DENLS ........... 108
    9.5.3 Comparing DEALS with others algorithms ............... 109
  9.6 Conclusion ....................................................... 110
I

Thesis
Chapter 1

Introduction

Nowadays, optimization is one of the most important issues in industrial systems design and product development [1, 2, 3, 4]. Optimization algorithms enhance the performance of industrial products while they help reducing the cost to be more competitive. In the industrial world, companies often need to optimize one or more factors (less cost, better performance, etc.), where optimization can play an important role, adjusting or fine tuning system designs in terms of the factors mentioned before. This is not a trivial task, especially when the problem space is complex and has high dimensionality.

Metaheuristics [5, 6, 7] are promising techniques that are beneficially used when classic optimization algorithms cannot solve complex problems effectively. Metaheuristics can solve a wide range of problems with different characteristics and they are able to find good solutions, acceptable solutions, in a feasible time, compared with traditional algorithms. Metaheuristics for optimization can be divided into two categories: trajectory-based approaches and population-based approaches. Trajectory-based algorithms make transitions from a possible solution to a different one (sometimes even worse solutions are also considered for a transition). Some examples of trajectory-based approaches are hill-climbing [5], simulated annealing [8], tabu search [9], gradient descent [10], or simplex method [11]. Multi-start strategy has also been used here to obtain better global solutions, leading to multi-trajectory based algorithms such as iterative local search [6], variable neighborhood search [12], greedy randomized adaptive search procedure [13] and iterative greedy search [14]. In population-based approaches such as evolutionary algorithms (EAs), a group of solutions (also called population) are handled at the same time, with a new population being generated from an old one. During the last decades, EAs have been proved as a powerful tool to solve many real-world optimization problems. They can find a good solution (probably not the theoretically optimal one).
Chapter 1

Introduction

Nowadays, optimization is one of the most important issues in industrial systems design and product development [1, 2, 3, 4]. Optimization algorithms enhance the performance of industrial products while they help reducing the cost to be more competitive. In the industrial world, companies often need to optimize one or more factors (less cost, better performance, etc.), where optimization can play an important role, adjusting or fine tuning system designs in terms of the factors mentioned before. This is not a trivial task, especially when the problem space is complex and has high dimensionality.

Metaheuristics [5, 6, 7] are promising techniques that are beneficially used when classic optimization algorithms cannot solve complex a problems effectively. Metaheuristics can solve a wide range of problems with different characteristics and they are able to find good solutions, acceptable solutions, in a feasible time, compared with traditional algorithms. Metaheuristics for optimization can be divided into two categories: trajectory-based approaches and population-based approaches. Trajectory-based algorithms make transitions from a possible solution to a different one (sometimes even worse solutions are also considered for a transition). Some examples of trajectory-based approaches are hill-climbing [5], simulated annealing [8], tabu search [9], gradient descent [10] or simplex method [11]. Multi-start strategy has also been used here to obtain better global solutions, leading to multi-trajectory based algorithms such as iterative local search [6], variable neighborhood search [12], greedy randomized adaptive search procedure [13] and iterative greedy search [14]. In population-based approaches such as evolutionary algorithms (EAs), a group of solutions (also called population) are handled at the same time, with a new population being generated from an old one. During the last decades, EAs have been proved as a powerful tool to solve many real-world optimization problems. They can find a good solution (probably not the theoretically opti-
mal one) of a NP-hard problem in a feasible time. There is a big family of EAs, but the most famous are Differential Evolution (DE) [7], Genetic Algorithm (GA) [15] and Particle Swarm Optimization (PSO) [16]. Other algorithms can be found in [17].

Hybridization of EAs with local search (LS) algorithms [18, 19] has been a hot research topic. Applying local search techniques can bring two benefits to the whole search procedure. First, they can drive the search into a better area further from local optima. Second, but not less important, they can enhance the exploitation of some promising areas of the search space so as to speed up the convergence of the search. These hybridized algorithms are also called Memetic Algorithms (MAs) [20, 21] in the literature.

Differential Evolution (DE) has been a competitive class of EAs. It is characterized by the way in which new solutions are created based on differences between pair(s) of individuals randomly selected from the population. DE has been popular in many applications due to its high ability, simple implementation and easy use with a relatively low number of control parameters [4, 22]. However, the performance of DE is not always excellent. On one hand, DE can occasionally get stuck in local optima causing unsatisfactory final solutions. On the other hand, DE has some operations and parameters that are problem dependent. This means that, depending on the characteristics of distinct problems, different parameter values will be required. Improper setting of control parameters for DE will lead to poor optimization performance.

This thesis aims to develop new methods and techniques to mitigate the two aforementioned problems of DE in solving complex optimization problems. First, suitable local search schemes have been investigated and incorporated into DE to reduce the risk of local stagnation. Second, adaptive DE algorithm has been developed allowing for automatic adjustment of control parameters of DE without requiring prior knowledge of the problem.

1.1 Problem Formulation

The problem of optimization, without loss of generality, can be formulated as the task of finding a vector of variables \( X = (x_1, x_2, \ldots, x_n) \) inside the search space \( \Omega \), to minimize the objective function \( f \). Therefore, no any other point in the search space will have an objective value larger than \( f(X^*) \).

In mathematical principle, an optimal solution to a nonlinear objective function must have all its partial derivatives equal to zero, as shown in Eq. (1.1),

\[
\frac{\partial f}{\partial x_i} = 0, \quad i = 1, 2, \ldots, n
\]  

(1.1)
All points that satisfy the condition in Eq. (1.1) are termed as stationary points. Further, in order to be a minimum solution, the Hessian matrix of the second derivatives, as defined in Eq. (1.2), has to be positive definite at the stationary point.

\[
H = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}
\end{bmatrix}.
\] (1.2)

However, when it comes to an engineering problem, normally an analytic formulation of the objective function is not available, not to mention the second order partial derivatives. Very frequently the objective value for a possible solution is derived from a specific domain-based evaluation such as simulation, as shown in Fig. 1.1. Hence we need to seek numerically approximate solutions instead of mathematically exact solutions in such practical engineering scenarios.

This thesis focuses on Differential Evolution as a class of metaheuristic techniques to search for approximate solutions to real-parameter optimization tasks. The goal is to develop new methods and algorithms to enhance standard DE algorithm in dealing with a wide variety of complex problems.

Figure 1.1: Domain-based evaluation
1.2 Research Questions

Given the weakness of DE as stated before, it is important to address the following two research questions to further augment the performance of DE in wide engineering applications.

**RQ1:** How an online adaptation of the main parameters of DE can be made to make search more efficient in different regions of the space with different properties?

Mutation factor and crossover rate are the main control parameters that affect the search process of differential evolution. Depending on the mutation factor value, the movement step size inside DE can be larger or smaller, therefore this value should be determined based on the state in which the evolutionary process is situated. The crossover rate affects the number of variables that will be altered in every iteration of the evolutionary process, hence its setting will be dependent on the characteristics of the problem at hand.

**RQ2:** How local search can be conducted inside the global search process of DE to enhance its performance whereas requiring no derivative information?

It is well known that gradient descent can be used as a local search method to locate new positions to quickly reduce objective values. However, approximating derivative information requires evaluations of a number of adjacent points, which would consume too much computing resource. An interesting issue for investigation is how DE could be combined with derivative-free local search for improving its exploration and exploitation capability, leading to higher quality solutions found in shorter time.

1.3 Scientific Contribution

In this section, we present the contributions of the thesis that address the previously formulated research questions.

**SC1:** The first research contribution of this thesis deals with RQ1, providing a method to adapt mutation factor and crossover rate inside DE. An adaptive differential evolution algorithm called “Greedy Adaptive Differential Evolution” (GADE) has been developed in paper A. The adaptation approach proposed for GADE can automatically adjust mutation factor and
crossover rate during the course of search while solving a specific problem. It relies on experience to update both parameters online, adopting a new method to calculate the progress rate that a candidate (for mutation factor and crossover rate) has accomplished. In paper B, GADE was further tested and verified with the application to a practical industrial scenario.

**SC2:** The second scientific contribution tackles RQ2, proposing the possibilities of incorporating four derivative-free local search schemes into a standard DE algorithm. Three of these approaches were handled in Paper C and they belong to a group called “Eager Local Search”. Each method in that group uses a different probability distribution, in order to create random perturbation to the best individual in the population, as is summarized below:

- Random Local Search uses a uniform distribution in order to create evenly distributed perturbations when testing new positions in the search space.
- Cauchy Local Search uses a Cauchy distribution to perform a small perturbation with high probability, but still it gives low chance to perform a large perturbation.
- Normal Local Search uses a normal distribution to offer the highest chance of small perturbation in comparison with uniform and Cauchy distributions.

The forth method is called “Alopex-based Local Search” (ALS) and it was investigated in paper D. This approach, using a temperature parameter that changes with state of the evolutionary process, makes large moves at the early stage of the search and small moves at the late phase. By means of this, the balance between exploration and exploitation is adapted automatically.

Papers A, B, C, and D are included in the thesis. The relation between the included papers and the research questions is given in Table 1.
Table 1.1: Relation between the included papers and research questions

<table>
<thead>
<tr>
<th>paper</th>
<th>RQ1</th>
<th>RQ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPER A</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PAPER B</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PAPER C</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PAPER D</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

1.4 Research Method

In this thesis, the basic research methodology was to identify the research problems by studying the state-of-art and related works and then attempt to propose new methods and algorithms to solve the problems. For evaluation, we employed an empirical method to evaluate the proposed solutions mainly on a set of benchmark problems that are widely used by the research community. Essentially the research has been conducted iteratively in the following steps:

1. Perform literature study to identify research problems

2. Propose a solution (algorithm or method) to solve the identified research problems

3. Implement the proposed solution in a software system

4. Evaluate the proposed solution in different benchmark problems given by the research community. It is also possible to do evaluation in practical industrial problems.

5. Compare the results of the proposed solution with the current state-of-art algorithms
Chapter 2

Background

Numerical methods for optimization aim to create progressively improved approximation to the problem in a number of iterations [17]. Generally, they can be divided into two large categories: trajectory-based approaches and population-based approaches. In trajectory-based approaches only one new solution is generated in each iteration, while population-based approaches produce a new population of solutions from the preceding population. In relevance to the research of the thesis, this chapter will introduce Alopex [23] search that belongs to the first category (trajectory-based) and standard Differential Evolution (DE) algorithm that belongs to the second category (population-based).

2.1 Alopex Search

Harth and Tzanakou proposed Alopex (Algorithm of pattern extraction) for optimization and pattern matching in visual receptive fields [23]. Alopex aims to estimate the gradient of the objective function by measuring the changes between two independent solutions. In the first step, the correlation between two solutions, \( A = (a_1, a_2, \ldots, a_n) \) and \( B = (b_1, b_2, \ldots, b_n) \), is calculated following Eq. (2.1).

\[
C_j = (a_j - b_j) \cdot [f(A) - f(B)] \quad \text{for} \quad j = 1, 2, \ldots, n \tag{2.1}
\]

where \( f(A) \) and \( f(B) \) are the objective values of A and B, respectively.

Considering a minimization problem as an example, the probability of a negative move on the \( j \)th variable is calculated using the Bolzmann distribution as
\[ P_j = \frac{1}{1 + e^{-\frac{C_j}{T}}} \]  \hspace{1cm} (2.2)

where T is obtained following Eq. (2.3), in which the average of the absolute value from all the correlations of all variables is calculated.

\[ T = \frac{1}{n} \cdot \sum_{j=1}^{n} |C_j| \]  \hspace{1cm} (2.3)

The direction of the move is represented by \( \delta_j \) and it is determined according Eq. (2.4).

\[ \delta_j = \begin{cases} 
-1 & \text{if } P_j \geq \text{rand}(0, 1) \\
1 & \text{otherwise} \end{cases} \]  \hspace{1cm} (2.4)

In the final step, the new trial solution, \( Q = (q_1, q_2, \ldots, q_n) \), is calculated as

\[ q_j = a_j + \delta_j \cdot |a_j - b_j| \cdot N(0, \lambda) \]  \hspace{1cm} (2.5)

where \( N(0, \lambda) \) is a random value between \(-\infty\) and \(\infty\) following a normal distribution with mean equal to 0 and standard deviation equal to \(\lambda\). If \( f(Q) \) is better or equal than \( f(A) \), then A is replaced by T. This process can continue until a stop criteria is satisfied, i.e., the maximum number of evaluations have been reached or a sufficiently good solution has been found.

### 2.2 Differential Evolution

DE is a stochastic and population-based algorithm. The population is composed of \( N_p \) individuals and every individual in the population represents a possible solution to the problem. DE operates in three consecutive steps in every iteration: mutation, crossover and selection, as shown in the diagram in Fig. 2.1. The explanation of these steps is given below:

**MUTATION.** \( N_p \) mutated individuals are generated using some individuals of the population. A vector for the mutated solution is called mutant vector. There are different strategies to create a mutant vector. Usually they are notated as \( \text{DE/x/y} \), where x stands for the vector to be mutated and y represents the number of difference vectors used in the mutation. Here only the three most common mutation methods are explained. Other mutation strategies and their performance have been discussed in [24].
The first mutation strategy is random mutation strategy, in which three randomly selected individuals from the population are used to generate the mutant vector. More precisely, this mutant vector is created according to Eq. (2.6).

\[ V_{i,g} = X_{r_1,g} + F \cdot (X_{r_2,g} - X_{r_3,g}) \]  

(2.6)

where \( V_{i,g} \) represents the mutant vector, \( i \) stands for the index of the vector, \( g \) stands for the generation, and \( r_1, r_2, r_3 \in \{1, 2, \ldots, N_p\} \) are random integers. \( F \) is the scaling factor in the interval \([0, 2]\). Fig. 2.2 shows how this mutation strategy works in a two dimensional problem. All the variables in the figure appear in Eq. (2.6) with the same meaning, and \( d \) stands for the difference vector between the individuals \( X_{r_2,g} \) and \( X_{r_3,g} \). This mutation method is often represented by \( \text{DE/rand/1} \), because it only uses one difference vector.

Figure 2.1: Differential Evolution process

Figure 2.2: Random mutation with one difference vector
The second mutation strategy uses the best individual from the population to create the mutant vectors. This strategy is represented by DE/best/1 and the mutant vectors are created in terms of Eq. (2.7).

\[ V_{i,g} = X_{\text{best},g} + F \cdot (X_{r1,g} - X_{r2,g}) \]  

(2.7)

where \( X_{\text{best},g} \) represents the best individual in the population at generation \( g \). Fig. 2.3 shows how DE/best/1 works in a two dimensional problem. All the variables in the figure appear in Eq. (2.7) with the same meaning, and \( d \) stands for the vector of difference between the individuals \( X_{r1,g} \) and \( X_{r2,g} \).

![Figure 2.3: Mutation on the best individual with one difference vector](image)

The third mutation strategy moves the current individual towards the best individual in the population before being disturbed with a scaled difference of two randomly selected individuals. This mutation strategy is notated as DE/current-to-best/1. Thus, the mutant individual is created according to Eq. (2.8).

\[ V_{i,g} = X_{i,g} + F1 \cdot (X_{\text{best},g} - X_{i,g}) + F2 \cdot (X_{r1,g} - X_{r2,g}) \]  

(2.8)

where \( X_{i,g} \) represents the current individual at generation \( g \). Fig. 2.4 shows how DE/current-to-best/1 works in a two dimensional problem. All the variables in the figure appear in Eq. (2.8) with the same meaning, and \( d_1 \) stands for the difference vector between the current individual \( X_{i,g} \) and the best individual \( X_{\text{best},g} \), and \( d_2 \) denotes the difference vector between \( X_{r1,g} \) and \( X_{r2,g} \).

**CROSSOVER.** In the second step, we recombine the set of mutant vectors created in the first stage (mutation) with the original population members to generate offspring solutions. A new offspring is denoted by \( T_{i,g} \) where \( i \) is the
Figure 2.4: current to best mutation with one difference vector

index and $g$ refers to the generation. There are two basic crossover strategies [25]: binomial and exponential.

The binomial crossover employs Eq. (2.9) to create an offspring solution.

$$T_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } \text{rand}[0, 1] < CR \text{ or } j = j_{\text{rand}} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \quad (2.9)$$

where $j$ stands for the index of every parameter in a vector, $CR$ is the probability of the recombination and $j_{\text{rand}}$ is a randomly selected integer between 1 and $N_p$ to ensure that the offspring will not be the same as the parent. In binomial crossover, the inherited parts from the mutant vector are randomly distributed across all dimensions. An example is given in Fig. 2.5.

![Binomial Crossover Example](image)

Figure 2.5: Example of binomial crossover

The exponential crossover constructs the offspring vector containing a sequence of consecutive components (in a circular manner) taken from the mutant vector. The offspring vector is derived, in Eq. (2.10), as follows
Chapter 2. Background

\[ T_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } j \in \{k, \{k + 1\}_n, \{k + 2\}_n, \ldots, \{k + L - 1\}_n\} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \]  

(2.10)

where \( k \) and \( L \) are two randomly selected integers in \( \{1, 2, \ldots, (n - 1)\} \), \( n \) stands for the dimension of the problem and \( L \) represents the number of elements that will be taken from the mutant vector. \( \{x\}_n \) is \( x \) if \( x \leq n \) and \( x - n \) if \( x > n \). An example (with \( k=3, L=3 \)) is given in Fig. 2.6.

![Example of exponential crossover (k=3,L=3)](image)

**Figure 2.6: Example of exponential crossover (k=3,L=3)**

**SELECTION.** In this last step, the offspring is compared with its parent, and the stronger one will enter the population for the next generation. Therefore, if the problem to solve is a minimization problem, the next generation is created according to Eq. (2.11).

\[ X_{i,g+1} = \begin{cases} T_{i,g} & \text{if } f(T_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \]  

(2.11)

where \( X_{i,g} \) is an individual in the old population, \( X_{i,g+1} \) is the individual in the next generation, \( T_{i,g} \) is the offspring, \( f \) stands for the objective function of the optimization problem.
Chapter 3

Related work

This chapter presents a survey of the related works. Section 3.1 presents an overview of the previous works on adaptations of the main parameters of DE. Section 3.2 outlines different local search methods that have been used in cooperation with Differential Evolution.

3.1 Parameter adaptation in Differential Evolution

Researchers have proposed different methods to adapt the main control parameters: mutation factor (F), crossover rate (CR) and population size (NP) for Differential Evolution. Qin and Suganthan proposed the SaDE algorithm [26], as one of the earliest works to adapt parameters in DE. Their algorithm uses normal distributions to assign mutation factor and crossover rate to an individual in the population. SaDE also maintains a memory of the successful CR values, the median of them is used as the mean of the distribution for CR. Other works that are relevant to SaDE can be found in [27, 28].

The idea of using successful values (for F and CR) in order to change the mean of a distribution has been commonly practiced by many researchers. In the JADE algorithm [29] proposed by Zhang and Sanderson, Cauchy distribution and normal distribution are used to determine the mutation factor and the crossover rate, respectively. The mean values of both distributions are updated after each generation using Lehmer mean and arithmetic mean of the successful parameter values respectively. Similarly Islam and Das proposed the MDE_pBX algorithm [30], which uses the same distributions but a different mean (called pow) to update the centres of both distributions. Likewise Wang proposed the dn_DADE algorithm [31], in which the the mean and the standard deviation of the normal distributions are updated. An improved version of
JADE was given in [32] in which the standard deviation of the distributions is modified in a similar way to that in dn_DADE.

More recently Tanabe and Fukunaga proposed the SHADE algorithm [33], in which the same distributions as in JADE are used to generate control parameters. The difference between both algorithms is that JADE uses a single pair of mean values to guide the parameter adaptation, while in SHADE a historical record of successful parameter settings are updated in a cycle and also used during the evolutionary process.

A different method termed as EPSDE was proposed by Mallipedi and Suganthan [34]. Therein the mutation factor and crossover rate are selected from different pools of values. The pool of F values is taken in the range 0.4-0.9 with steps of 0.1, and the pool of CR values is taken in the range 0.1-0.9 in steps of 0.1. Successful combinations of F and CR values are stored and they will be used in future generations. If the combination for an individual is not successful, it will be reinitialized with a new random combination or with a combination randomly selected from the storage of successful combinations. Similar work was conducted by Wang et al. [35] in the development of the CoDE method, which only uses one pool containing three different combinations of F and CR values. Brest et al. proposed the jDE algorithm [36] for self-adapting control parameters in DE. The key to this algorithm is the extension of individuals in the population with parameter values. Thus the adjustment of control parameters is implemented by evolution on the individual level. Generally, better parameter values result in new individuals of higher quality and thereby acquiring more chances to propagate themselves.

Liu and Lampinen proposed a fuzzy adaptive differential evolution algorithm (FADE) [37]. This method adapts the control parameters (F and CR) of DE employing fuzzy logic controllers whose inputs are the relative objective function values and individuals of successive generations.

Besides, population size adaptation has also received some attention from researchers. In [38], Teo proposed two versions of an algorithm for self-adapting populations (DESAP-Rel and DESAP-Abs). In DESAP-Abs the population size of a subsequent generation is set as the average of the population size attribute for all members of the current population. However, DESAP-Rel takes the current population size plus an increasing/decreasing percentage in terms of the population growth rate.
3.2 Local Search in Differential Evolution

Incorporating a local search (LS) heuristic is often very useful in the design of an effective evolutionary algorithm for global optimization. Local search techniques can be divided into two groups according to Lozano et al. [39]:

- **Local improvement process (LIP) oriented LS (LLS):** Local search heuristics in this group applies local improvements on one or more individuals in the population. Some well known techniques, like hill-climbing, gradient descent, or some random or predefined perturbations can be used here. LIPs can be applied after a certain number of generations and possibly with some probability of doing it.

- **Crossover-based LS (XLS):** In this second group, the local search is implemented by recombination operators, in which the individuals from the population are used as parents to produce offspring around them. These techniques have an attractive property of self-adaptation [40], since the population is evolving as the search progresses. As in LIPs, XLS can be applied after a certain number of generations.

Next, we shall review the previous works of LS in these two classes that are incorporated into a global DE algorithm.

3.2.1 LLS in DE

There are many works that belong to LLS. One of the ideas is to use the chaos theory [41] in order to create random perturbations for local search in DE. Pei-chong et al. [42] proposed DECH algorithm, a hybridization of DE with a chaos local search (CLS) approach. In this algorithm, CLS is applied to the best 20% of the individuals after a regular DE iteration. If the perturbation makes the new individual better, it will enter the population as a member of the next generation. The other individuals are discarded and then regenerated randomly. A similar work was performed by Jia et al. [43], where CLS is used after every generation on the best individual.

The local search in [44] performs a perturbation $p$ to the individual on every single dimension. This perturbation first is done in one direction and only if it does not improve, then the perturbation will be performed in the opposite direction. If the final solution does not improve after trying on all dimensions, the current step size $p$ will be halved for the next iteration. This local search method uses a large number of function evaluations, so a local limitation has been made to avoid over usage of computing resource by LS.
3.2.2 XLS in DE

Crossover based local search methods have also attained much research attention. Noman and Iba [45] proposed the algorithm called DEfirSPX, in which a simplex crossover (SPX) is used. The best individual and (p-1) random individuals are selected. SPX is applied with a number of times to these individuals, and then the best solutions obtained will participate in the selection process inside the DE procedure. An improved version of this algorithm was proposed in [46], which is called DEahcSPX. Different from DEfirSPX which uses a fixed search length, the search length is adapted in DEahcSPX using an adaptive hill climbing strategy. This strategy makes use of feedback from the search in order to select a suitable length of search in the dynamic process.

Gu and Gu [47] proposed a local search method that uses difference between the current individual and the best individual in the population to modify the current individual. After every generation this LS operator is applied with a probability to all individuals in the population. Alternatively, two other approaches were proposed in [48]. The first method, called DETLS, uses a trigonometric local search, in which a weighted average of three points from the population (the best individual and two other random individuals) is calculated as the new individual. The second method, called DEILS, is based on quadratic interpolation. As in DETLS, the same three individuals are selected, but in this case a parabolic curve is created and then the minimum point in the curve is identified. In both algorithms (DEILS and DETLS) the local search is applied after every generation and it continues until no further improvement.

Peng and Wu [49] proposed a heterozygous differential evolution with Taguchi local search method (THDE). Taguchi method uses two random individuals from the population and the middle point between them (each parent is divided into four parts or factors) in order to create nine new individuals. The effect of the three parents is calculated and for each factor, the part of the best parent is copied to the offspring. Subsequently a random individual in the population is replaced by the resultant offspring. A similar approach proposed in [19] focuses on the worst individual in the population, which is replaced by the resultant offspring if it is weaker than the offspring.

3.2.3 Strategies of Applying LS in DE

Researches follow different strategies when they apply a LS technique in DE. In the methods such as DECH [42], DECLS [43], DETLS and DEILS [48] as explained above, local search is applied after every generation of DE. Different from these methods, there are some hybridizations that perform LS after a cer-
tain number of generations. Two different approaches, that follow this strategy, were proposed in [50], in which local search chains (LSC) [20] are applied to SaDE [26]. The first method, SaDE\textsubscript{LAM}, updates the population after performing the LSC. In the second method (SaDE\textsubscript{BAL}), the population is not updated after LCS is applied, but the new created individuals are saved in an external archive. In order to apply LSC again, the best individual from the external archive and the population will be used as the starting point. This strategy is also used in [44].

Another key issue to take into account is to which individuals of the population the local search technique should be applied. In some algorithms, the LS technique is only applied to the best individual in the population [43, 48]. Applying LS to the best individual is the most commonly adopted option by the researchers, but others alternatives can also be used. For instance, in [42], LS is applied on a percentage of the best individuals in the population. Alternatively, in [44], LS is applied to the individuals with their fitness values higher than the average. In another option presented in [47], LS is applied with a probability to all individuals of the complete population.
Chapter 4

Overview of the Papers

- **Paper A: Greedy adaptation of control parameters in differential evolution for global optimization problems**: Miguel Leon, Ning Xiong, Evolutionary Computation (CEC), 2015 IEEE Congress on, p. 385-392.

**Abstract:** Differential evolution (DE) is a very attractive evolutionary and meta-heuristic technique to solve many optimization problems in various real-world scenarios. However, the proper setting of control parameters of DE is highly dependent on the problem to solve as well as on the different stages of the search process. This paper proposes a new greedy adaptation method for dynamic adjustment of mutation factor and crossover rate in DE. The proposed method is based on the idea of greedy search to find better parameter assignment in the neighborhood of a current candidate. Our work emphasizes reliable evaluation of candidates via applying a candidate with a number of times in the search process. As our purpose is not merely to increase the success rate (the survival of more trial solutions) but also to accelerate the speed of fitness improvement, we suggest a new metric termed as progress rate to access the quality of candidates in support of the greedy search. This greedy parameter adaptation method has been incorporated into basic DE, leading to a new DE algorithm called Greedy Adaptive Differential Evolution (GADE). GADE was tested on 25 benchmark functions in comparison with five other DE variants. The results of evaluation demonstrate that GADE is strongly competitive: it obtains the best ranking among the counterparts in terms of the summation of relative errors across the benchmark functions.

**My Contribution:** I am the main author of the paper contributing with
the idea, the algorithm, the implementation and the experiments. Also, I wrote the major draft of the paper.


**Abstract:** Model calibration represents the task of estimating the parameters of a process model to obtain a good match between observed and simulated behaviours. This can be considered as an optimization problem to search for model parameters that minimize the discrepancy between the model outputs and the corresponding features from the historical empirical data. This chapter investigates the use of Differential Evolution (DE), a competitive class of evolutionary algorithms, to solve calibration problems for nonlinear process models. The merits of DE include simple and compact structure, easy implementation, as well as high convergence speed. However, the good performance of DE relies on proper setting of its running parameters such as scaling factor and crossover probability, which are problem dependent and which can even vary in the different stages of the search. To mitigate this issue, we propose a new adaptive DE algorithm that dynamically adjusts its running parameters during its execution. The core of this new algorithm is the incorporated greedy local search, which is conducted in successive learning periods to continuously locate better parameter assignments in the optimization process. In case studies, we have applied our proposed adaptive DE algorithm for model calibration in a Furnace Optimized Control System. The statistical analysis of experimental results demonstrate that the proposed DE algorithm can support the creation of process models that are more accurate than those produced by standard DE.

**My Contribution:** I am the main author of the paper contributing with the idea, the algorithm, the implementation and the experiments. Also, I wrote the major draft of the paper.

Abstract: Differential evolution (DE) presents a class of evolutionary computing techniques that appear effective to handle real parameter optimization tasks in many practical applications. However, the performance of DE is not always perfect to ensure fast convergence to the global optimum. It can easily get stagnation resulting in low precision of acquired results or even failure. This paper proposes a new memetic DE algorithm by incorporating Eager Random Search (ERS) to enhance the performance of a basic DE algorithm. ERS is a local search method that is eager to replace the current solution by a better candidate in the neighborhood. Three concrete local search strategies for ERS are further introduced and discussed, leading to variants of the proposed memetic DE algorithm. In addition, only a small subset of randomly selected variables is used in each step of the local search for randomly deciding the next trial solution. The results of tests on a set of benchmark problems have demonstrated that the hybridization of DE with Eager Random Search can substantially augment DE algorithms to find better or more precise solutions while not requiring extra computing resources.

My Contribution: I am the main author of the paper contributing with the idea, the algorithm, the implementation and the experiments. Also, I wrote the major draft of the paper.


Abstract: Differential evolution (DE), as a class of biologically inspired and meta-heuristic techniques, has attained increasing popularity in solving many real world optimization problems. However, DE is not always successful. It can easily get stuck in a local optimum or an undesired stagnation condition. This paper proposes a new DE algorithm Differential Evolution with Alopex-Based Local Search (DEALS), for enhancing DE performance. Alopex uses local correlations between changes in individual parameters and changes in function values to estimate the gradient of the landscape. It also contains the idea of simulated annealing that uses temperature to control the probability of move directions during the search process. The results from experiments demonstrate that the use of Alopex as local search in DE brings substantial performance improvement over the standard DE algorithm. The proposed DEALS
algorithm has also been shown to be strongly competitive (best rank) against several other DE variants with local search.

**My Contribution:** I am the main author of the paper contributing with the idea, the algorithm, the implementation and the experiments. Also, I wrote the major draft of the paper.
Chapter 5

Conclusion and Future Work

In this thesis, we have proposed different methods to enhance the search power of DE. First, the Greedy Adaptive Differential Evolution (GADE) algorithm has been developed. GADE is able to adapt its mutation factor and crossover rate online using the greedy search scheme as well as a new metric on progress rate to evaluate the candidates of parameter assignment. The results of the experiments demonstrate that the adaptation of the parameters improves the performance of DE and that our proposed GADE algorithm is rather competitive to the state-of-the-art algorithms. Additionally, GADE has been applied and tested in a practical industrial problem with good results that outperform the standard DE algorithm. GADE should be attractive in engineering applications, since it is easy to use without requiring prior knowledge of DE.

Secondly, four different LS strategies have been investigated when being incorporated into DE. Three of them can be encapsulated into a general group called Eager Random Search (ERS). The three approaches in ERS share the same principle in that random perturbations are made to the individuals in order to improve the performance of solutions. The difference among the three lies in the different distributions used. Using normal or Cauchy distribution leads to discovery of better results in unimodal functions, since the perturbations made to individuals are more likely to be small. Using a uniform distribution is expected to give good results in multimodal functions containing multiple local optima. The forth proposed LS technique uses the Alopex (ALS) method in order to enable big moves at the beginning and small moves at the late stages of the search. Hence it can automatically balance exploration and exploitation with the evolutionary process. DEALS is shown to outperform some other memetic algorithms (MAs) in a set of benchmark problems.

So far the research in the thesis employs a single population for DE. A multi-population based DE could help avoiding premature convergence in the
early stages of the search. If one of the populations gets stuck into a local optimum, a second population could help the first population to move out of the stagnation. Hence, combination of different mutation strategies via cooperative sub-populations could be investigated in the future research. Moreover, as mutation strategy is a sensitive factor to the problems of interest, a study of all the existing mutation strategies has to be done to acquire new knowledge of which mutation strategy would be most suitable in a given situation.

Another interesting work in the future would be integration of parameter adaptation into a memetic DE algorithm (with local search) to reach more competent DE performance in solving complex and large scale optimization problems.
Bibliography


global optimization. In IEEE Conference on Evolutionary Compu-
tation, pages 806–812, 2014.

for differential evolution. In 2013 IEEE Congress on Evolutionary Com-
putation (CEC), Cancun, Mexico, pages 71–78, 2013.

[34] R. Mallipeddi, P.N. P.N. Suganthan.and K.Q. Pan, and M.F. Tasgetiren.
Differential evolution algorithm with ensemble of parameters and muta-

vector generation strategies and control parameters. IEEE Transaction on

control parameters in differential evolution: A comparative study on nu-
umerical benchmark problems. In IEEE Transaction on Evolutionary Com-


[38] J. Teo. Exploring dynamic self-adaptive populations in differential evo-

memetic algorithms with crossover hill-climbing. Evolutionary Compu-

[40] H.G. Beyer and K. Deb. On self-adaptive ffeature in real-parameter evo-
lutionary algorithms. IEEE Transaction on Evolutionary Computation,

[41] B. Jiang. Optimizationion complex functions by chaos search. Cibernet-

Wuhan, China, pages 1–4, 2009.


