Comparison of Contemporary Evolutionary Algorithms on the Rotated Klee-Minty Problem

Michael Hellwig
Vorarlberg University of Applied Sciences, Research Center PPE
Dornbirn, Austria
michael.hellwig@fhv.at

Patrick Spettel
Vorarlberg University of Applied Sciences, Research Center PPE
Dornbirn, Austria
patrick.spettel@fhv.at

Hans-Georg Beyer
Vorarlberg University of Applied Sciences, Research Center PPE
Dornbirn, Austria
hans-georg.beyer@fhv.at

ABSTRACT

The Rotated Klee-Minty Problem represents an advancement of the well-known linearly constrained Klee-Minty Problem that was introduced to illustrate the worst case running time of the Simplex algorithm. Keeping the linearity as well as the complexity of the original Klee-Minty Problem, the Rotated Klee-Minty Problem remedies potential biases with respect to Coordinate Search and turns out to be a suitable constrained test environment for randomized search heuristics. The present paper is concerned with the comparison of recent evolutionary algorithm variants for constrained optimization on this respective test bed. The considered algorithm variants include the most successful participants of the CEC Competition on Single Objective Real-parameter optimization and other selected strategies that are not directly applicable to the CEC test suite. Taking into account the diverse constraint handling approaches, the performance results of all search heuristics are interpreted. It turns out that most strategies that have been successful in the CEC competitions do have severe problems on the RKMP.

CCS CONCEPTS

• **Theory of computation → Optimization with randomized search heuristics**; *Bio-inspired optimization*; Linear programming;

KEYWORDS

Randomized Search Heuristics, Constrained Optimization, Benchmarking, Rotated Klee-Minty Problem

ACM Reference Format:
Michael Hellwig, Patrick Spettel, and Hans-Georg Beyer. 2019. Comparison of Contemporary Evolutionary Algorithms on the Rotated Klee-Minty Problem. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3319619.3326805

1 INTRODUCTION

The development and the assessment of Evolutionary Algorithms (EA) widely rely on benchmarking due to the limited availability of theoretical performance results. One goal of such benchmark environments is supporting the selection of that algorithm best suitable for a given real-world applications. Other aims would be gathering experimental insights into the working principles of algorithmic ideas and fostering the algorithm development for specific tasks. Further, benchmarks can be used to verify theoretically derived performance results.

Regarding single-objective constrained optimization benchmarks for randomized search heuristics, the most elaborated test suites have been provided in the context of the IEEE Congress on Evolutionary Computation (CEC) competitions (2006, 2010, and 2017) on constrained real-parameter optimization. These test functions as well as the associated rules for algorithm evaluation are commonly used to promote novel algorithmic ideas in the EA community. However, these benchmark sets mainly consist of very complex non-linear constraint and objective functions. Yet, rather simple constrained problem formulations still may present a difficult task for EAs. Additionally, the CEC benchmarks are missing a clear structure that would allow for allocating specific problem classes to especially beneficial algorithmic ideas in retrospect. While other constrained test function sets do exist, the respective problems do not come with properly defined standards for algorithm comparison and usually do not scale with the search space dimension. An exception is the COCO BBOB-constrained test suite [4] which is currently under development. It is based on the unconstrained COCO suite and makes use of 8 distinct objective functions. Yet, it is restricted to varying numbers of (almost) linear inequality constraints (i.e. the boundary of the feasible set is subject to small nonlinear perturbations).

Compared to the vast number of distinct constrained problem features, only a small number of well-designed benchmarks exist. This makes it hard to determine those features that do make a constrained optimization problem hard for a specific search algorithm. Hence, a benchmark suite needs to take into account consistent subgroups of conceivable problems. Based on these considerations, the Rotated Klee-Minty Problem (RKMP) was proposed in [7]. It scales with the search space dimensionality, and, by construction, causes the Simplex-Algorithm to exhibit an exponential worst-case running time. While EAs usually cannot compete with custom-built linear problem (LP) solvers, the RKMP represents a reasonably complex problem even for certain LP solvers (e.g. interior-point methods (IPM) or basis-exchange pivoting algorithms). Interestingly, by realizing comparable or even improved precision, some EA variants for constrained optimization are able to compete with LP solvers on the RKMP [7, 13].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326805

The present paper is concerned with the assessment and the performance comparison of selected EA algorithms for constrained optimization. Interpreting the gathered data, we come across the interesting observation that most of the algorithms that are considered to be successful in the CEC competitions exhibit severe problems on the RKMP. Furthermore, the comparison gains insights into the working principles of those strategies using the so-called ϵ-level constraint handling [15]. This constraint handling technique is part of the recently most successful algorithms for constrained optimization, but it slows down the algorithm performance in the context of the RKMP. The present paper comes up with an explanation of this ϵ-level constraint handling drawback.

The rest of this paper is organized as follows: We recap the Rotated Klee-Minty Problem and the comparison methodology in Sec. 2. Section 3 then introduces the algorithms considered in this study. The algorithm results are presented and discussed in Sec. 4. The paper is concluded in Sec. 5.

2 BENCHMARK FORMULATION

This section recaps the Rotated Klee-Minty benchmark definition introduced in [7] and states the corresponding benchmarking conventions as well as the criteria considered for algorithm assessment. The source code of the Rotated Klee-Minty Problem is available in a separate Github.com branch [14].

2.1 The Rotated Klee-Minty Problem

The Rotated Klee-Minty Problem represents a scalable linear constrained optimization problem [7]. The optimization problem is determined by

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^N} \quad & \boldsymbol{d}^\top \boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{A}\cdot\boldsymbol{R}(\boldsymbol{y}-\boldsymbol{t}) \le \boldsymbol{b}, \\ & \check{\boldsymbol{y}} \le \boldsymbol{y} \le \hat{\boldsymbol{y}}, \end{aligned} \tag{1}$$

with linear objective function given by $\boldsymbol{d} = (0, 0, \ldots, 0, 1)^\top \in \mathbb{R}^N$. Note that the terms objective function value and fitness are used synonymously, as essentially evolutionary methods are considered in this study.

The feasible region of (1) represents a perturbed unit hyper cube in the $\mathbb{R}^N$ which is defined by a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$ as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}+\boldsymbol{I} \\ \boldsymbol{B}-\boldsymbol{I} \end{pmatrix} \in \mathbb{R}^{2N\times N}, \text{ and } \boldsymbol{b} = \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{2N\times 1}, \tag{2}$$

with identity matrix $\boldsymbol{I} \in \mathbb{R}^{N\times N}$,

$$\boldsymbol{B} = \begin{pmatrix} 0 & 0 & \ldots & 0 & 0 & 0 \\ \epsilon & 0 & \ldots & 0 & 0 & 0 \\ 0 & \epsilon & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \epsilon & 0 & 0 \\ 0 & 0 & \ldots & 0 & \epsilon & 0 \end{pmatrix} \in \mathbb{R}^{N\times N}, \text{ and } \epsilon = 1/10. \tag{3}$$

By application of the translation vector $\boldsymbol{t} = N^3\cdot(1, 1, \ldots, 1)^\top \in \mathbb{R}^N$ and the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{N\times N}$ ($\det(\boldsymbol{R}) = 1$), the feasible region of problem (1) is relocated within the N-dimensional domain.[1] For the definition of the rotation matrix $\boldsymbol{R}$ refer to [7]. The optimal solution is moved to $\boldsymbol{y}^* = N^3\cdot\boldsymbol{1}$ with $f(\boldsymbol{y}^*) = \boldsymbol{d}^\top\boldsymbol{y}^* = N^3$. The relocation of the optimal solution away from the origin removes undesired problem characteristics that would prevent reasonable algorithm comparisons. An illustration of the transformation in the 2-dimensional case is provided in Fig. 1.

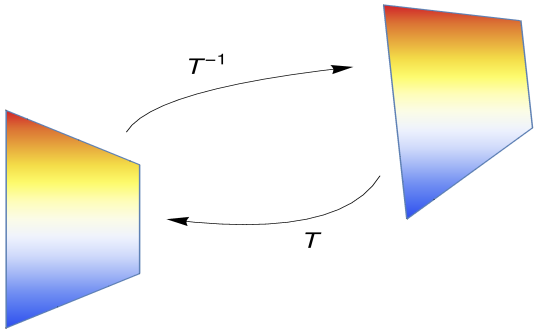

Figure 1: Translation and rotation of the 2-dimensional Klee-Minty cube. The contour lines of the objective function are indicated by the color (or grey-scale) gradient.

Additionally, we provide component-wise lower and upper parameter bounds, or box-constraints, to constitute the domain of eligible input values for black-box optimization strategies and to facilitate the generation of initial populations. These are defined by $\check{\boldsymbol{y}} = \boldsymbol{0} \in \mathbb{R}^N$ and $\hat{\boldsymbol{y}} = 5N^3\cdot\boldsymbol{1} \in \mathbb{R}^N$.

2.2 Algorithm Assessment

This section introduces the benchmarking principles as well as the performance indicators used in the context of the Rotated Klee-Minty Problem. The proposed benchmark problem takes into account search space dimensions $N \in \{2, 3, 5, 10, 20, 40\}$. Accordingly, six distinct constrained functions are considered. For each dimension, the following three performance indicators are derived from 15 independent algorithm runs:

i) The algorithm runtime is measured with respect to the function evaluations consumed. This information is displayed by use of empirical cumulative distribution functions (ECDF), or performance profiles [11], respectively.
ii) We assess the effectiveness of each algorithm by taking into account its feasibility ratio (FR) as well as the deviation of its median fitness realization (among all final feasible solutions) from the known optimal function value.
iii) Effectiveness is also evaluated in the search space by taking into account the combination of the FR and the mean deviation of all feasible solutions from the known optimizer.

Depending on the search space dimension, the total budget of function evaluations per run is predefined as $2\cdot 10^4\cdot N$. One single function evaluation is consumed regardless of whether the objective function, a single constraint function, or the whole constrained problem is evaluated for a single parameter vector $\boldsymbol{y}$ at a time.

The final candidate solutions realized in different algorithm runs are compared by use of a *lexicographic ordering* $\preceq_{\text{lex}}$ that takes

[1] The deterministically chosen motions represent a trade-off between problem hardness for LP solvers and unbiasedness of EAs.

into account the objective function value $f(\boldsymbol{y})$ as well as the corresponding amount of constraint violation $\nu(\boldsymbol{y})$. The respective order relation is defined by

$$\boldsymbol{y} \preceq_{\text{lex}} \boldsymbol{z} \Leftrightarrow \begin{cases} f(\boldsymbol{y}) \leq f(\boldsymbol{z}), & \text{if } \nu(\boldsymbol{y}) = \nu(\boldsymbol{z}), \\ \nu(\boldsymbol{y}) < \nu(\boldsymbol{z}), & \text{else.} \end{cases} \tag{4}$$

In the context of the RKMP, the objective function value corresponds to $f(\boldsymbol{y}) \coloneqq \boldsymbol{d}^\top \boldsymbol{y}$. The constraint violation value $\nu(\boldsymbol{y})$ can be measured as the sum of the deviation over all inequality constraints

$$\nu(\boldsymbol{y}) \coloneqq \sum_{i=1}^{N} \max\left\{0, \left(\boldsymbol{ARy} - \boldsymbol{ARt} - \boldsymbol{b}\right)_i\right\}. \tag{5}$$

The lexicographic order relation permits to define the quality indicators that can be used to assess and compare algorithm performance on problem (1). Note that the lexicographic order is also known as Deb's method or as the Superiority of Feasibility (SOF) principle [5].

Taking into account different search space dimensions, the effectiveness indicators (ii) and (iii) of each algorithm are presented in a table. Facilitating the comparison of multiple algorithms, we use a graphical representation. To this end, the indicators are plotted against the dimension. Note that, due to the limitation of the page number, the tabular presentation of the individual algorithm results is omitted in the paper but can be found in the supplementary material. Regarding performance indicator (i), the running time of meta-heuristic algorithms can be directly identified with the number of function evaluations needed to satisfy a number of predefined targets. This definition is adopted from [11]. It is used in the context of the COCO BBOB benchmarks. A more detailed explanation of the ECDF construction and interpretation is provided in [6]. We also apply a bootstrapping procedure to obtain more accurate performance profiles while minimizing the benchmark execution time. The pseudo code of the bootstrapping used is displayed in Alg. 1. Yet, the target definition used for the Rotated Klee-Minty Problem is different. Instead of defining targets only for the feasible region of the search space, we allocate about half of the targets to the infeasible region. This supports the illustration of algorithm behavior within the infeasible region. Refer to Sec. 4 for the illustration of the algorithm results. In total, 103 target values are defined.

The 52 targets in the infeasible region are defined with respect to the constraint violation of a candidate solution. They are uniformly distributed between 10^4 and 10^{-6} as well as 0. Realizing a candidate solution with constrained violation below a target definition for the first time, a target value is considered to be hit.

The logarithmically equidistant targets in the feasible region range from 10^0 to 10^{-8} and take into account the fitness value realized by a strategy. The targets are reached after having realized a feasible candidate solution with an objective function value that deviates from the known optimal value by a number smaller than a certain target value. For example, target $t_i, (i \in 1, \ldots, 51)$ is hit after the algorithm generates a candidate solution $\boldsymbol{y}$ which satisfies $|f(\boldsymbol{y}) - f(\boldsymbol{y}^*)| < t_i$ for the first time. The ECDF plots display the percentage of the targets reached in all 15 algorithm runs for any number of function evaluations. Accordingly, an algorithm realizes a 100% target ratio if and only if it manages to yield a feasible final solution with higher accuracy than specified by the final target precision 10^{-8} in all 15 runs. This way they provide a notion of algorithm performance: Smaller upper left areas indicate faster algorithm running times [11].

Algorithm 1 Pseudo-code for the bootstrapping.

Initialize the maximum number of tries K for sampling a successful run and the number of bootstrapped runs S to generate
for all targets t **do**
  if at least one of the 15 runs reached target t **then**
    for s **in** $\{1, \ldots, S\}$ **do** $k \leftarrow 0$
      success $\leftarrow$ **false**
      $rl \leftarrow 0$
      while ((**not** success) **and** $k < K$) **do**
        Sample one run r of the 15 runs
        with replacement uniformly at random
        $rl \leftarrow rl$ + fevals of run r to reach target t
        if r was a successful run **then**
          success $\leftarrow$ **true**
        end if
        $k \leftarrow k + 1$
      end while
      if success **then**
        Runlength for the bootstrapped run s
        considering target t is rl
      end if
    end for
  end if
end for

3 METHODS

This section introduces the eight search strategies considered for comparison on the Rotated Klee-Minty Problem. Among the five considered Differential Evolution (DE) variants, the selection of algorithms comprises the CEC competition winners of 2010 (ϵDEag), 2017 (LSHADE44), and 2018 (iUDE), respectively. The other two DE variants, and the ϵMAg Evolution Strategy (ES), also turned out to be successful on many CEC benchmark problems. In particular, the ϵMAg-ES showed superior performance with regard to high-dimensional search spaces of the CEC 2018 competition. The other ES variants have not been developed to solve the CEC benchmark problems. Both instead originate from theoretical considerations on constrained optimization problems. For example, the Active-Set-ES and the lcCMSA-ES assume that linear constraints are provided in explicit form. As all the algorithms treat the RKMP problem as a black-box problem, a preprocessing step that determines the RKMP linear constraint system is necessary to compare those two with the other strategies. The remainder of this section briefly introduces all search algorithms considered in the present study and provides their references. Due to the space restrictions, this paper is confined to the comparison of EA variants on RKMP. We refer to [7] for results of the Octave interior point LP-solver *glpk*. That respective paper also indicates very poor performance of Random Search on RKMP. Hence, a comparison of the EA variants with Random Search is omitted in this paper.

ConSaDE – The Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization (ConSaDE) has been proposed in [9]. It incorporates constraint handling into the Self-adaptive Differential Evolution (SaDE) algorithm. The two central ideas of the SaDE algorithm are the use of multiple mutation operators and self-adaptive parameter adaptation. The mutation operators DE/rand/1, DE/current-to-best/2, DE/rand/2, and DE/current-to-rand/1 are used together with binary crossover. Each of them is applied with equal probability in the beginning. The probability is then adapted from one generation to the next according to the number of successful individuals created by the respective mutation operators. Additionally, local search (Sequential Quadratic Programming) is used every 500 generations for convergence speedup. The extension ConSaDE uses the lexicographic ordering for handling constraints the selection step.

ECHT-DE – The algorithm Differential Evolution with Ensemble of Constraint Handling Techniques (ECHT-DE) [10] combines multiple constraint handling methods into a single DE algorithm with the idea that some constraint handling approaches are better suited for some problems than others. The constraint handling approaches used are Superiority of Feasiblility, Self-Adaptive Penalty, ϵ-level constraint handling, and Stochastic Ranking. Every of those constraint handling approaches has its own population with a set of parameters and generates offspring. For the computation of the next generation's population of a specific constraint handling method, the offspring populations of the other constraint handling methods are considered in addition to the parental population of the constraint handling method under consideration.

ϵDEag – The algorithm Constrained Optimization by the ϵ Constrained Differential Evolution with an Archive and Gradient-Based Mutation (ϵDEag) has been presented in [16]. It integrates four main ideas into a DE algorithm. First, an archive of individuals is kept throughout the optimization. Individuals from the archive form a part of the trial vector generation with the intention to maintain diversity. Second, if an offspring is not better than its parent, the corresponding parental individual is used to generate a second offspring. Third, the ϵ-level constraint handling approach is used for dealing with constrained optimization problems. And fourth, gradient-based mutation is performed to mutate in a way that reduces the constraint violation.

L-SHADE44 – L-SHADE with Competing Strategies Applied to Constrained Optimization (LSHADE44) [12] is an algorithm that enhances the Success History based DE with linear population size reduction (L-SHADE) [17]. It incorporates four different combinations of mutation and crossover operators that compete with each other for the creation of a trial point. Specifically, DE/current-to-pbest/1/bin, DE/current-to-pbest/1/exp, DE/randrl/1/bin, and DE/randrl/1/exp are used. For the constraint handling, the strategy uses the lexicographic ordering approach.

iUDE – The Unified Differential Evolution (UDE) algorithm for constrained optimization problems has been proposed in [18]. An *improved* version of this algorithm, the iUDE, represents the winner strategy of the IEEE CEC competition on single objective real-parameter optimization. While the respective paper was not yet published in the context of the CEC, a technical report [19] as well as the Matlab source code of iUDE are available on the CEC2018 competition website[2]. The algorithm combines a multitude of ideas from DE algorithms. It makes use of the three trial vector generation approaches DE/rand/1, DE/current-to-rand/1, and DE/current-to-pbest/1. A dual population approach with strategy adaptation is used. For constraint handling, a combination of lexicographic and ϵ-level constraint handling is used.

ϵMAg-ES – Based on the Matrix Adaptation (MA) ES [3], the ϵMAg-ES [7] incorporates basically three constraint handling techniques. It uses a reflection approach for the box-constraints and the ϵ-level constraint handling to rank the individuals of a current population. Additionally, it occasionally makes use of a repair approach that is based on estimated gradients for equality and inequality constraints in order to prevent premature convergence. The ϵMAg-ES achieved the second over-all rank in the CEC 2018 competition. In particular, it showed superior performance in high-dimensional search spaces.

Active-Set-ES – The Active-Set-ES [1] is a $(1+1)$-ES interleaved with an evolution of the active set of constraints. In every generation one feasible offspring is generated. It is then projected either onto the whole feasible region or onto a reduced search space. The choice is done uniformly at random with a fixed probability. The reduced search space is created by making all the inequality constraints equality ones that are active at the parent. It replaces the parent if its objective function value is better.

lcCMSA-ES – The Linear Constraint CMSA-ES (lcCMSA-ES) [13] is an interior point method that specializes on linear constraints. It is based on the Covariance Matrix Self-Adaptation (CMSA) ES [2] and extends it with constraint handling. It turns a set of linear (in)equality constraints into a linear equality constraint system with a non-negativity constraint. Then, the main idea of the approach is to evolve in the null space of the constraint matrix. By doing this, the mutated individuals do not violate the linear equality constraints. Possible violation of the non-negativity constraint is dealt with using repair by projection onto the non-negative orthant.

The last two ES variants need information about the problem specific constraints. As the RKMP is regarded to be a black-box optimization problem, this information needs to be gathered in a preprocessing step and the associated function evaluations consumed need to be taken into account appropriately to ensure a fair competition. The preprocessing step reconstructs the matrix/vector form of the RKMP constraints by querying the black-box constraint function. For this, the constraints are evaluated for a given number of randomly sampled vectors in the search space to create a (possibly overdetermined) system of equations. Solving it (possibly using the pseudoinverse) allows computing the original RKMP linear constraint system. With this preprocessing, the last two ES variants are able to evolve in such a way that the individuals are always feasible. They start with an initially feasible solution which is obtained by projecting onto the feasible region.

The original algorithm codes are used for our comparisons where possible. These are partly available on the CEC competition website or are published by the authors themselves. Only the code of the ϵDEag algorithm needed to be transferred from a C to a

[2] http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2018/Constrained

Matlab implementation. In order to establish a fair comparison, we slightly adjust the termination criteria to fit the Rotated Klee-Minty framework. That is, the search is stopped after the maximal function evaluation budget is consumed or after having found a feasible candidate solution the fitness of which deviates from the optimal fitness value by less than 10^{-8}. The strategy parameters recommended by the algorithm developers have been used.

4 RESULTS

Having introduced the algorithms of interest, the corresponding test results obtained on the Rotated Klee-Minty benchmark are presented and discussed in Sec. 4.1. Interestingly, in comparison to the outcome of the past CEC competitions, one observes very different performance results of those strategies that use the so-called ϵ-level constraint handling method. The reason for this performance deterioration is examined in Sec. 4.2.

4.1 Algorithm Comparison

Aiming at a performance comparison on the rotated Klee-Minty Problem (1), Fig. 2 displays the performance profiles (ECDF plots) of the eight algorithms under consideration (Sec. 3) for the dimensions $N \in \{2, 3, 5, 10, 20, 40\}$. Each plot displays the percentage of targets reached in all 15 runs for each number of function evaluations. We use two distinct black-rimmed markers (∇ and Δ) to illustrate the transition from the infeasible region to the feasible region for each algorithm. The markers must be interpreted as follows: The ∇ marker displays the expected number of function evaluations for which an algorithm has reached a single feasible target for the first time in at least one of the considered runs. Whereas, the Δ indicates the expected number of function evaluations after which at least one feasible solution was found *in each algorithm run*. Hence, the Δ maker indicates the expected number of function evaluations to realize a feasibility ratio of one.

Taking into account the respective markers (∇ and Δ), one obtains a clear ranking of all algorithms in each dimension.[3] The effect of the preprocessing for lcCMSA-ES and for Active-Set-ES is clearly visible. In the implementation for the experiments, an overdetermined system consisting of $10 \cdot N$ equations is used to reconstruct the RKMP linear constraint system. Because the lcCMSA-ES and the Active-Set-ES then project onto this system, the step observed in Fig. 2 is the result.[4] That is, on average those two algorithms reach the feasible region earliest. Among the other strategies, ConSaDE appears to realize feasible solutions first (especially in high dimensions). However, the algorithm faces difficulties to satisfy all feasible targets. Except for Active-Set-ES and iUDE, all algorithms can be considered rather resilient in terms of their search behavior in the infeasible region. In contrast, due to numerical approximation errors, the Active-Set-ES struggles to find feasible solutions in each run (Δ) for higher dimensions. That is, many runs of the Active-Set-ES faces stagnation in the vicinity of the boundary of the feasible region.

[3]Note that the close proximity (in terms of function evaluations) of both data points for most algorithms is augmented by the logarithmic scaling and by the resolution of the x-axis.

[4]A number of further evaluations (constant w.r.t. N) are used up in the preprocessing for tasks such as determining the number of constraints. Hence, the step observed in Fig. 2 at around 10^1 is slightly above 10^1 for the smaller dimensions.

Regarding the infeasible targets for the remaining algorithms, the transitioning into the feasible region occurs rather rapidly. For dimension 2, the LSHADE44 reaches the feasible region first, followed by the ConSaDE, the ECHT-DE, the ϵDEag, the ϵMAg-ES, and the iUDE. For dimension 3, it is very similar, except that the ϵMAg-ES reaches the feasible region before the ϵDEag. In dimensions 5 and 10 the order is ConSaDE, LSHADE44, ECHT-DE, ϵMAg-ES and ϵDEag, however, iUDE does not reach it. The situation for dimensions 20 and 40 is very similar, except that ECHT-DE performs worse than for small N.

Considering also the targets in the feasible region, the ConSaDE reaches all targets in dimensions 2, 3, 5, 20, and about 90% of the targets for dimensions 10 and 40. The ECHT-DE reaches all targets for the dimensions 2, 3, and 5. For the dimensions 10, 20, and 40, about 70%, 60%, and 50% of the targets are reached, respectively. In higher dimensions, it takes considerably more function evaluations to reach the same number of targets. The ϵDEag reaches all the targets for dimensions 2 and 3. In dimensions 5, 10, 20, and 40 the strategy satisfies about 98%, 70%, 58%, and 55% of the targets, respectively. The LSHADE44 reaches all the targets in all dimensions. The Active-Set-ES reaches all the targets for the dimensions 2, 3, 5, and 10. It reaches about 78% and 72% of the targets for the dimensions 40 and 20, respectively. The lcCMSA-ES reaches all the targets for the dimensions 2, 3, 5, and 10. It reaches about 73% and 67% of the targets for the dimensions 20 and 40, respectively. Considering the ϵMAg, it reaches all the targets in all the dimensions. The iUDE reaches all the targets only for dimensions 2 and 3. This is very surprising as it was the winner of the IEEE CEC 2018 competition. Further investigations in this regard revealed that the ϵ constraint handling is a reason for this behavior on the RKMP (see Sec. 4.2 for a more detailed discussion of this aspect).

In overall, the LSHADE44, the ϵMAg-ES, and the ConSaDE are the best performing algorithms when considering all the dimensions and default parameters on the RKMP (1). For dimensions 2, 3, 5, and 10, the lcCMSA-ES and the Active-Set-ES can also compete with the LSHADE44 and the ϵMAg-ES, but they require more function evaluations to reach all the targets. However, they are able to beat the ConSaDE for dimension 10. For the dimension 40, only the LSHADE44 and the ϵMAg-ES are able to reach all the targets. Regarding the strategies that make use of the ϵ-level constraint handling, the ϵMAg-ES exhibits the best performance on the RKMP in all dimensions $N \geq 3$.

Figs. 3 and 4 show plots comparing the errors in the objective function space and the search space, respectively, for the different algorithms and the dimensions considered. From them, the same observations as for the ECDF plots can be concluded, however, from a different perspective. The median objective function error plot shows that the LSHADE44 and the ϵMAg-ES reach all the targets for all the dimensions in the median (as they are on the dashed black line). Apart from that, one can see that the median error to the optimum in the objective function increases with increasing dimension for the other algorithms. This also holds for the mean norm error in the parameter space. Note that only the algorithms that achieve feasible solutions in all the runs are considered in those two figures. Information on the feasibility ratio of the other strategies and on the performance indicators ii) and iii) corresponding

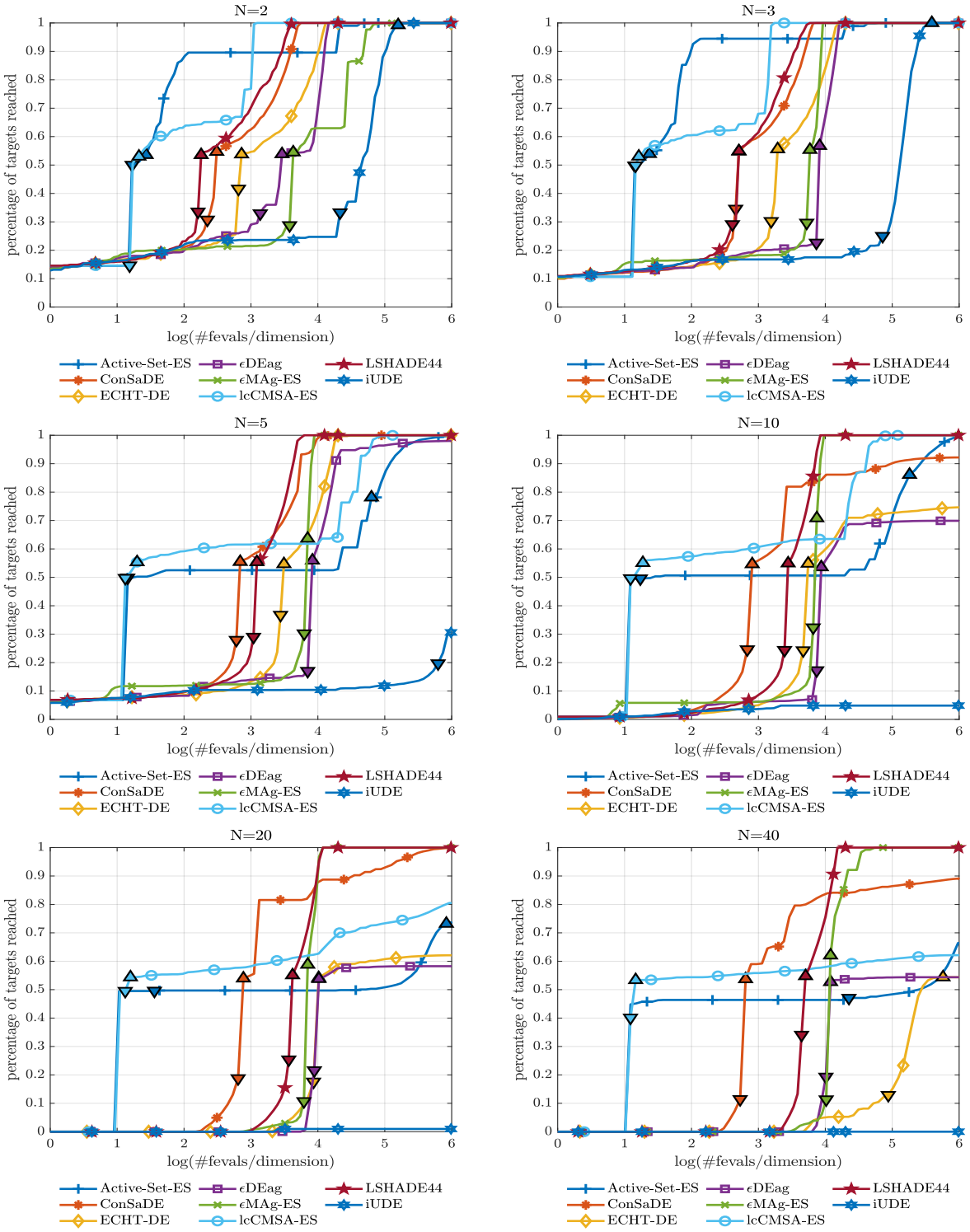

Figure 2: Comparison of the performance profiles (ECDF plots) of the eight algorithms presented in Sec. 3 obtained on problem (1) in different dimensions. The black-rimmed markers are obtained based on the bootstrapped data. The ∇ marker displays the event of hitting the first feasible target, and Δ indicates number of function evaluations for which all runs have reached the feasible region, respectively.

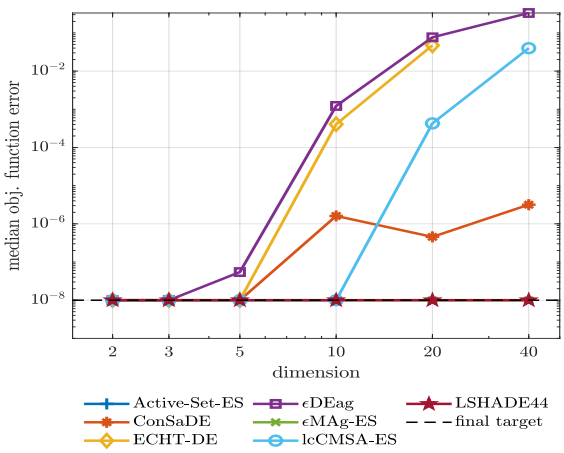

Figure 3: Comparison of the median fitness value deviations from the known optimal objective function value N^3. The algorithm realizations are plotted against the search space dimension N. The final target defines a precision of 10^{-8}. Data points for each algorithm are only shown if all runs of that algorithm resulted in feasible solutions for the given dimension. In $N = 2$ and $N = 3$ all data points do interleave. Only these two data points are obtained for the Active-Set-ES. Notice that all the data points obtained by ϵMAg-ES and by LSHADE44 overlap with the final target for all dimensions.

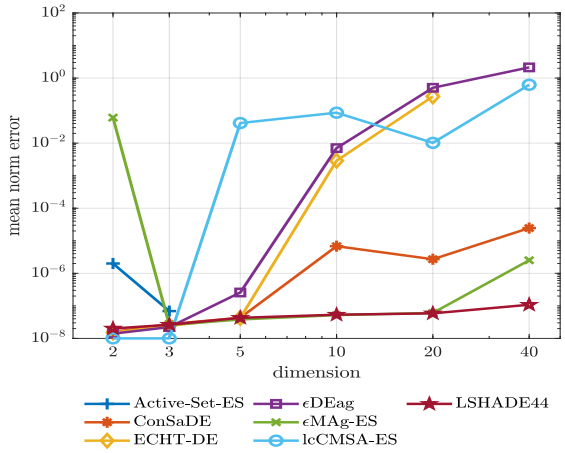

Figure 4: Comparison according to the mean distance deviation from the known optimal solution of (1). The algorithm results are plotted against the search space dimension N. Data points are only shown if all runs of a particular algorithm resulted in feasible solutions for the given dimension.

to Fig. 3 and Fig. 4 are available in the supplementary material. As suggested in [7], the results are presented in tabular form.

In order to precisely distinguish two similar performing algorithms, particular events may be examined for significant differences by making use of statistical tests, e.g. differences in the amount of function evaluations needed to reach the final target. Depending on the number of compared algorithms and the test objective, the Wilcoxon rank-sum test or the Friedman test are well-known non-parametric hypothesis tests for this purpose.

4.2 Additional Investigations

In this section we compare only those algorithms that incorporate the ϵ-level constraint handling approach [15]. This constraint handling represents a relaxation of the lexicographic ordering presented in (4). The ϵ-level constraint handling softens the principle that feasible solutions are always considered superior to infeasible solutions. It enables the algorithm to treat infeasible candidate solutions with constraint violation below a specific $\epsilon^{(g)}$ threshold as feasible. The threshold $\epsilon^{(g)}$ is continuously reduced to zero with the number of iterations g. Hence, the strategy is able to move outside the feasible region within the early phase of the search process which can potentially support the convergence to the optimizer $\boldsymbol{y}^*$ in some cases. This constraint handling technique turned out to be especially successful in the context of the CEC competitions. In fact, the LSHADE44 algorithm (CEC 2017) is the only competition winner that refrained from using the ϵ-level constraint handling.

Preliminary experiments suggest that the performance deterioration on the Rotated Klee-Minty Problem (1) can at least partly be attributed to the use of ϵ-level constraint handling. In order to support this claim, we execute the following experiment. The performance profiles of iUDE, ϵDEag, and ϵMAg-ES, are compared to those ECDF plots that are obtained by running the exact same algorithms after turning off the ϵ-level constraint handling method. To this end, the ϵ threshold is initially set to zero in all algorithm runs, i.e. the ϵ-level ordering is replaced with the lexicographic ordering (4). The resulting algorithm variants are denoted as lexiUDE, lexDEag, and lexMAg-ES, respectively. Due to the space limitation, the comparison of the corresponding ECDF plots is only illustrated for dimension $N = 40$ in Fig. 5. Yet, the results in lower dimensions look very similar. One observes in Fig. 5 that setting the ϵ threshold to zero initially improves the original counterparts for the algorithms considered. The lexDEag transitions into the feasible region earlier but is not able to reach more feasible targets than the ϵDEag. The lexMAg-ES reaches all targets with less evaluations compared to the ϵMAg-ES. The most surprising result is achieved by the lexiUDE. Whereas the iUDE is not even able to reach the infeasible targets, its lexicographic variant lexiUDE performs similarly to the lexMAg-ES[5].

In order to further investigate the influence of ϵ, an additional experiment is conducted. While Fig. 5 shows the two extremes of setting ϵ to zero initially and using the original ϵ decay, Figure 6 displays the results obtained by accelerating the ϵ reduction within the iUDE. An acceleration of the ϵ decay is expected to result in a performance in between those exhibited by the two extremes (see Fig. 5). Considering the three corresponding iUDE variants (iUDE, lexiUDE, and iUDE (with faster ϵ decay)), the empirical results substantiate our proposition. As intuitively expected, the lexiUDE achieves the best performance, the original iUDE performs worst, and the iUDE with faster ϵ threshold reduction shows a performance that is in

[5]Note that, in terms of the ECDF plots, the lexMAg-ES performance and the lexiUDE performance, respectively, turn out to be slightly superior to the LSHADE44 performance on the RKMP.

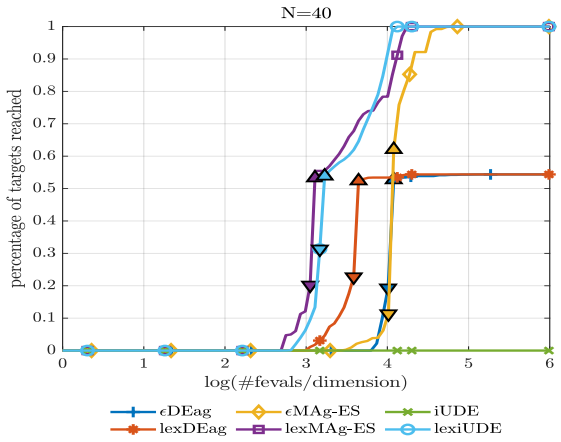

Figure 5: Investigation of the ϵ-level constraint handling impact on the algorithm performance. The black-rimmed markers display the first (∇) observation of hitting a feasible target, and the number of function evaluations when all (Δ) runs have reached the feasible region, respectively.

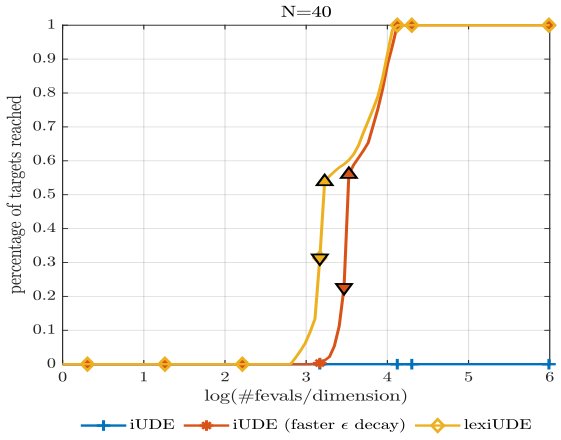

Figure 6: Investigation of the ϵ-level constraint handling influence on the algorithm performance of the iUDE.

between the former two variants. Searching for an explanation of this weakness, we come to the conclusion that the benefit of using the ϵ-level constraint handling is strongly connected to the analytical structure of the objective function. The objective function of problem (1) is a linear function and thus strictly monotonic. Hence, the optimal solution is always located on the boundary of the feasible region. In such situations, a certain relaxation of the constraints that define the boundary of the feasible set will result in the relocation of the optimal solution. For simplicity, the emerging optimal solution is referred to as *the relaxed optimal solution*. Accordingly, when starting with comparably large $\epsilon^{(0)}$-threshold, a strategy employing the ϵ-level constraint handling will approach towards the relaxed optimal solution. By gradually reducing the $\epsilon^{(0)}$ with the number of iterations, the relaxed optimal solution will approach the real optimal solution. However, this process might already consume a large amount of function evaluations that would have been needed at the end of the algorithm run for the search within the feasible region, or even for the detection of the feasible region, respectively.

Hence, boundary relaxation techniques like ϵ-level constraint handling can be disadvantageous in similar environments, i.e. on constrained problems with strictly monotonic decreasing objective functions in at least one dimension of the search domain.

5 CONCLUSION

In this paper, the results of evaluating multiple randomized search methods on the RKMP [7] have been presented. To this end, the benchmark function has been recapped (1) and a selection of prominent algorithms for constrained optimization have been described. Regarding the corresponding empirical algorithm results yields an interesting observation: search algorithms that rely on the ϵ-level constraint handling technique can exhibit a comparably weak performance on the RKMP. This is in contrast to the results obtained in the context of the CEC competitions on single objective constrained real-parameter optimization, where these strategies commonly perform well. Whereas the ϵ constraint handling is beneficial in solving the CEC constrained benchmark problems, it is disadvantageous for the considered Rotated Klee-Minty Problem.

The RKMP still represents a quite new benchmark proposal. Taking into account the survey in [8], the design of elaborated test suites for benchmarking randomized search heuristics on constrained optimization problems still is in its infancy. Hence, the benchmarking conventions and the considered performance indicators must be regarded as first suggestions. For example, the questions of which performance indicators need to be captured to provide a full picture of the algorithm performance, how these should be parameterized, and how the results should be presented best in order to provide easy performance interpretations are not conclusively answered. They should to be thoroughly discussed and examined theoretically if possible.

However, due to the number of distinct constrained problem features, the work towards a sophisticated algorithm selection for real-world problems requires an extension of the available benchmark environments. The RKMP contributes to this line of research and the present paper supplies a first comparison of selected EAs on this linearly constrained problem class. In future studies, we aim to explore the incremental progression of the RKMP towards nonlinear objective functions and disconnected feasible regions (by adding multiple hyper cubes to the search domain).

Finally, an elaborated ranking approach that determines the best algorithm in a certain dimension is still missing. To this end, the question of how to combine the individual performance indicators in an unbiased way still needs to be answered.

ACKNOWLEDGMENTS

The authors thank all algorithm developers for making their code openly available. This work was supported by the Austrian Science Fund FWF under grant P29651-N32.

REFERENCES

[1] Dirk V. Arnold. 2016. An Active-Set Evolution Strategy for Optimization with Known Constraints. In *International Conference on Parallel Problem Solving from Nature*. Springer, 192–202.

[2] Hans-Georg Beyer and Bernhard Sendhoff. 2008. Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy –. In *Parallel Problem Solving from Nature – PPSN X: 10th International Conference, Dortmund, Germany, September 13-17, 2008. Proceedings*. Springer, Berlin, Heidelberg, 123–132.

[3] Hans-Georg Beyer and Bernhard Sendhoff. 2017. Simplify Your Covariance Matrix Adaptation Evolution Strategy. *IEEE Transactions on Evolutionary Computation* 21, 5 (Oct 2017), 746–759.

[4] Dimo Brockhoff, Nikolaus Hansen, Tea Tušar, O. Mersmann, Phillipe Rodrigues Sampaio, Anne Auger, and Asma Atamna *et al.* 2016. COCO Documentation Repository. (2016). http://github.com/numbbo/coco-doc/tree/gh-pages/docs.

[5] Kalyanmoy Deb. 2000. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* 186, 2 (2000), 311 – 338. https://doi.org/10.1016/S0045-7825(99)00389-8

[6] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, Dejan Tusar, and Tea Tusar. 2016. COCO: Performance Assessment. *CoRR* abs/1605.03560 (2016). http://arxiv.org/abs/1605.03560

[7] Michael Hellwig and Hans-Georg Beyer. 2018. A Linear Constrained Optimization Benchmark for Probabilistic Search Algorithms: The Rotated Klee-Minty Problem. In *Theory and Practice of Natural Computing*, David et al. Fagan (Ed.). Vol. 11324. Springer International Publishing, Cham, 139–151. https://doi.org/10.1007/978-3-030-04070-3_11

[8] Michael Hellwig and Hans-Georg Beyer. 2019. Benchmarking evolutionary algorithms for single objective real-valued constrained optimization âĂŞ A critical review. *Swarm and Evolutionary Computation* 44 (2019), 927 – 944. https://doi.org/10.1016/j.swevo.2018.10.002

[9] Vicky Ling Huang, A Kai Qin, and Ponnuthurai N Suganthan. 2006. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 17–24.

[10] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. 2010. Differential Evolution with Ensemble of Constraint Handling Techniques for Solving CEC 2010 Benchmark Problems. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[11] Jorge J Moré and Stefan M Wild. 2009. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization* 20, 1 (2009), 172–191.

[12] Radka Polakova and Josef Tvrdík. 2017. L-SHADE with competing strategies applied to constrained optimization. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 1683–1689.

[13] Patrick Spettel, Hans-Georg Beyer, and Michael Hellwig. 2018. A Covariance Matrix Self-Adaptation Evolution Strategy for Optimization under Linear Constraints. *IEEE Transactions on Evolutionary Computation* (2018), 1–1. https://doi.org/10.1109/TEVC.2018.2871944

[14] Patrick Spettel and Michael Hellwig. 2019. EA comparison on the Rotated Klee-Minty Problem. Source code. (2019). Retrieved April 18, 2019 from https://github.com/patsp/RotatedKleeMintyProblem/tree/ea_comparison

[15] Tetsuyuki Takahama and Setsuko Sakai. 2006. Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*. 1–8. https://doi.org/10.1109/CEC.2006.1688283

[16] Tetsuyuki Takahama and Setsuko Sakai. 2010. Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. 1–9. https://doi.org/10.1109/CEC.2010.5586484

[17] Ryoji Tanabe and Alex S. Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *IEEE Congress on Evolutionary Computation (CEC)*. 1658–1665. https://doi.org/10.1109/CEC.2014.6900380

[18] Anupam Trivedi, Krishnendu Sanyal, Pranjal Verma, and Dipti Srinivasan. 2017. A unified differential evolution algorithm for constrained optimization problems. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 1231–1238. https://doi.org/10.1109/CEC.2017.7969446

[19] Anupam Trivedi, Dipti Srinivasan, and Nimagna Biswas. 2018. *An Improved Unified Differential Evolution Algorithm for Constrained Optimization Problems*. Technical Report. http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2018/Constrained/Improved_Unified_Differential_Evolution_CEC_2018_Report.pdf (Technical Report accessed on Mar 11, 2019).