A Modified Matrix Adaptation Evolution Strategy with Restarts for Constrained Real-World Problems

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

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

***Abstract*—In combination with successful constraint handling techniques, a Matrix Adaptation Evolution Strategy (MA-ES) variant (the ϵMAg-ES) turned out to be a competitive algorithm on the constrained optimization problems proposed for the CEC 2018 competition on constrained single objective real-parameter optimization. A subsequent analysis points to additional potential in terms of robustness and solution quality. The consideration of a restart scheme and adjustments in the constraint handling techniques put this into effect and simplify the configuration. The resulting BP-ϵMAg-ES algorithm is applied to the constrained problems proposed for the IEEE CEC 2020 competition on Real-World Single-Objective Constrained Optimization. The novel MA-ES variant realizes improvements over the original ϵMAg-ES in terms of feasibility and effectiveness on many of the real-world benchmarks. The BP-ϵMAg-ES realizes a feasibility rate of 100% on 44 out of 57 real-world problems and improves the best-known solution in 5 cases.**

***Index Terms*—evolution strategies, constrained optimization, real-world competition, population control, restart strategy**

I. Introduction

Constrained optimization considers the search for the optimal solution of an objective function subject to restrictive constraints on the parameter vectors. Such constraints can result from limited resources, problem-specific trade-offs, appropriate physical boundaries, and many other sources. Particularly, in black-box optimization, the involvement of constraints into an optimization task increases the problem complexity.

In recent years, a growing number of meta-heuristics have been designed for solving constrained black-box optimization problems. Accordingly, there is a need for elaborate benchmark collections on which the effectiveness and efficiency of the proposed algorithms can be validated and compared. So far, constrained benchmark environments were limited to sets of, more or less, artificial test functions [1]–[5]. As these artificial test problems often do not contain the full complexity of real-world problems, the results are usually not directly transferable to practical applications. For this reason, a novel benchmark collection [6] that comprises a set of constrained real-world optimization problems for various industrial applications is worth taking into account. It might help to choose the best among the compared algorithms for a specific kind of real-world problem.

With support of the Austrian Science Fund FWF under grant P29651-N32.

In the context of the IEEE Congress on Evolutionary Computation 2020 (CEC 2020), the competition on Real-World Single-Objective Constrained Optimization is inviting competitive algorithms for solving these constrained real-world benchmark collection [7]. By changing the benchmark functions, this competition represents a further development of previous competitions. The evaluation criteria are based on the calculation and comparison of performance statistics.

As one of the first Evolution Strategies (ES) in the constrained competitions, a Matrix Adaptation Evolution Strategy (MA-ES) [8] variant, the so-called ϵMAg-ES [9], was able to achieve the overall second place at the CEC 2018 competition on constrained single-objective real parameter optimization in 2018. In addition, the algorithm turned out to be particularly successful in the high dimensional setting ($N = 100$). In [10], a detailed analysis of the algorithmic ϵMAg-ES components revealed that in many cases the best solution found during a single run is obtained after a relatively small number of function evaluations. That is, instead of stagnating during the consumption of the remaining budget, the use of a restart strategy seems to be more appropriate. By re-initialization in different search space areas, disjoint feasible sets might be explored and the best solution found can potentially be improved. In the case of connected feasible sets, premature convergence might be remedied and the robustness of the algorithm can be increased. Further, the recommended population size in [9] was based on the population sizes of comparable DE algorithms. The analysis showed that the best choice for the offspring population size depends very much on the structure of the problem. On average, however, smaller population sizes appeared to be beneficial in the constrained CEC 2018 benchmark settings. Following these observations, the ϵMAg-ES strategy is augmented with a restart mechanism that varies the population size. To this end, the BiPop-MA-ES implementation [8] for unconstrained problems, which is based on the ideas of [11] and [12], is used as a design guideline.

The analysis in [10] also revealed that the ϵ-level constraint handling approach used by ϵMAg-ES can turn out disadvantageous in some cases. In particular, this is true in situations where the objective function turns out to be monotonic in at least one dimension [5].

The presented algorithm aims at solving the real-valued

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

constrained real-world optimization problems specified in [6] for the CEC competition on constrained real-world problems in 2020. It is referred to as BP-ϵMAg-ES. By making use of restarts and varying the population sizes, BP-ϵMAg-ES represents an advancement of the ϵMAg-ES customized for the benchmarking competition. Additionally, the ϵ-level reduction is adjusted to rely on the ratio of ϵ-feasible solutions within the offspring population of the ES rather than on the number of generations elapsed. The design idea goes back to [13], but was further modified for the use within ϵMAg-ES.

The remainder of this paper is organized as follows: The general form of the constrained optimization problems and some useful notations are presented in Sec. II. Section III then introduces the BP-ϵMAg-ES for constrained optimization. Due to page limitations, it focuses primarily on the changes with respect to ϵMAg-ES [9], in particular. After that, the experimental setting of the CEC 2020 competition on constrained real-world problems as well as the parameter settings for the BP-ϵMAg-ES are summarized in Sec. IV. Section V presents the final performance statistics as specified in [7]. The paper concludes with a brief discussion of the observations.

II. General problem formulation

This paper considers constrained real-world optimization problems [6] of the general form

$$\begin{aligned} \min\ & f(\boldsymbol{y}) \\ s.t.\ & g_i(\boldsymbol{y}) \leq 0, \quad i = 1, \dots, l, \\ & h_j(\boldsymbol{y}) = 0, \quad j = 1, \dots, k, \\ & \check{\boldsymbol{y}} \leq \boldsymbol{y} \leq \hat{\boldsymbol{y}}. \end{aligned} \tag{1}$$

That is, the optimization goal is w.l.o.g. the minimization of the real-valued objective function $f(\boldsymbol{y})$ with respect to an N-dimensional search space parameter vector $\boldsymbol{y} \in \mathbb{R}^N$. The N-dimensional vectors $\check{\boldsymbol{y}}$ and $\hat{\boldsymbol{y}}$ define lower and upper box constraints of each parameter vector component $\boldsymbol{y}_k$, $k = 1, \dots, N$. Specifying reasonable ranges of the components $\boldsymbol{y}_k$, the box-constraints are provided in the source code that comes with [7].

The feasible region of (1) is additionally restricted by $m = l + k$ real-valued constraint functions. These constraint functions consist of inequality constraints $g_i(\boldsymbol{y})$, $i = 1, \dots, l$ and equality constraints $h_j(\boldsymbol{y})$, $j = 1, \dots, k$. A vector $\boldsymbol{y} \in S$ that satisfies all constraints simultaneously is called feasible. When considering problem (1) a measure of infeasibility is useful for ranking potentially infeasible candidate solutions. To this end, we compute the constraint violation $\nu(\boldsymbol{y})$ of a candidate solution $\boldsymbol{y}$ as

$$\nu(\boldsymbol{y}) = \frac{\sum_{i=1}^{l} G_i(\boldsymbol{y}) + \sum_{j=1}^{k} H_j(\boldsymbol{y})}{l + k}, \tag{2}$$

with functions $G_i(\boldsymbol{y})$ and $H_j(\boldsymbol{y})$ defined by

$$G_i(\boldsymbol{y}) \coloneqq \max\left(0, g_i(\boldsymbol{y})\right), \qquad \text{and} \tag{3}$$

$$H_j(\boldsymbol{y}) \coloneqq \begin{cases} |h_j(\boldsymbol{y})|, & \text{if } |h_j(\boldsymbol{y})| - \delta > 0 \\ 0, & \text{if } |h_j(\boldsymbol{y})| - \delta \leq 0 \end{cases}. \tag{4}$$

In order to be able to satisfy the equality constraints, the δ term introduces the necessary error margin. In accordance with [7], this paper considers $\delta = 10^{-4}$.

III. Algorithm

The MA-ES represents an algorithmically reduced CMA-ES variant [14] that was already implemented as a building block in a multi-restart approach similar to the BiPop-CMA-ES in [8]. The newly proposed BP-ϵMAg-ES transfers this approach designed for multi-modal problems to constrained optimization problems (1).

The Meta-ES approach presented in Alg. 1 utilizes a modified version of the ϵMAg-ES as its "search driver", see Alg. 2. Until the total budget of function evaluations is exhausted, the ϵMAg-ES is restarted with different population sizes and randomly initialized starting populations. In line 6 of the BP-ϵMAg-ES, the parental population size is updated. The truncation ratio $\vartheta = \mu/\lambda$ is held constant regardless of the offspring population size adjustment. Right after initialization, Alg. 1 executes a first ϵMAg-ES run with the initial parameter choices. In case that the first run already consumes the total budget, the Meta-ES approach will terminate before considering any restart at all. Hence, Alg. 1 is basically similar to Alg. 2. For this reason, the restart scheme will not affect the results of the ϵMAg-ES algorithm in terms of effectiveness. The choice whether to restart the ϵMAg-ES with small or large population size is balanced. This is performed in such a way that basically half of the total budget will be consumed in the branch working with small populations λ_S, and the other half in the branch using large populations λ. Therefore, the parameters b_S and b_L accumulate the corresponding number of function evaluations used. In the small population branch, the population size λ_S is randomly derived on the basis of the updated λ. The probabilistic choice is introduced by the uniformly distributed random number $u(0, 1)$ in the range [0,1]. In contrast to [8], the mutation strength σ is not probabilistically chosen. It is reset to σ_{Init} for every restart.

The case differentiation introduced in lines 8–11 of Alg. 1 controls the number of repair steps θ_r as well as the order relation used in the next ϵMAg-ES run. As long as $\nu_{\text{bsf}} > 0$, the *lexicograpic* ordering, i.e. $\epsilon^{(0)} = 0$, and $\theta_r = 20$ is used in every second restart. In all other cases, the settings according to Sec. IV are considered.

To avoid the consumption of large budget amounts in a single ϵMAg-ES runs, the search is aborted with respect to three different termination criteria. This way, wasting function evaluations in case of local convergence can be mitigated. Termination is enforced as soon as (TC1) the mutation strength is lower than a threshold below which only marginal improvements are expected ($\sigma < 10^{-12}$), as soon as (TC2) more than 10% of the total budget is spent without improvement of the best solution found so far, or as soon as (TC3) the total budget is exhausted. It must be noted that these termination criteria also affect the overall performance of the ϵMAg-ES. However, the final results appear to substantiate the parameter selections made for the CEC 2020 competition.

In a black-box scenario, the algorithm has no knowledge about the appropriate step-size at a random location in the search space. That is, unlike [8], the algorithm initially samples a uniformly distributed population $\mathcal{P}$ of λ candidate solutions $\boldsymbol{y}_j \in \mathbb{R}^N$, $j = 1, \ldots, \lambda$ within the predefined box-constraints $\check{\boldsymbol{y}}$ and $\hat{\boldsymbol{y}}$, respectively. This is implemented according to line 4 in Alg. 2. There, $\boldsymbol{u}(0,1)$ denotes an N-dimensional vector with uniformly distributed components in $(0,1)$. The starting population is also used to determine the initial $\epsilon^{(0)}$ value, in line 7 of Alg. 2, by considering the constraint violation of the median solution with respect to the *lexicographic ordering* relation (see (6) with $\epsilon \equiv 0$ const.). Notice, $\chi_{T>0}$ denotes the indicator function ensuring that the initial ϵ-level is set to zero in case $T = 0$.

After determination of $\epsilon^{(0)}$, the parent population is selected by choosing the μ best candidate solutions from the offspring population of size λ. Note that $\boldsymbol{y}_{m;\lambda}$ denotes the mth best out of λ with respect to the order relation $\leq_\epsilon$, see Eq. (6). The recombination of the weighted parents forms the initial parental centroid in line 9 of Alg. 2 and the best found candidate solution so far $\boldsymbol{y}_{\text{bsf}}$ is memorized in line 10. Standard weights w_i of the MA-ES are used as described in Sec. IV. The comparison of the candidate solutions involves the evaluation of the constrained problem. We account one function evaluation per evaluation of a constrained problem, i.e. including the objective function and all related constraints.

Algorithm 1 Pseudo code of the BP-ϵMAg-ES for constrained real-parameter optimization.

1: **Initialize:** μ_{Init}, λ_{Init}, $\sigma_{\text{Init}}, budget_{\max}, \vartheta \leftarrow \mu/\lambda$
2: $n \leftarrow 0$, $n_S \leftarrow 0$, $b_L \leftarrow 0$, $b_S \leftarrow 0$,
3: $[\boldsymbol{y}_{\text{bsf}}, f_{\text{bsf}}, \nu_{\text{bsf}}, evals] \leftarrow \epsilon$MAg-ES$(\mu_{\text{Init}}, \lambda_{\text{Init}}, \sigma_{\text{Init}})$
4: $budget_{\text{used}} \leftarrow evals$
5: **while** $budget_{\text{used}} < budget_{\max}$ **do** $n \leftarrow n + 1$
6: $\lambda \leftarrow 2^{n-n_S} \cdot \lambda_{\text{Init}}$
7: $\mu \leftarrow \lceil \lambda\vartheta \rceil$
8: **if** $\nu_{\text{bsf}} > 0$ **AND** $\text{mod}(n, 2) = 1$ **then**
9: $T \leftarrow 0$, $\theta_r \leftarrow 20$
10: **else** $T \leftarrow 500$, $\theta_r \leftarrow 3$
11: **end if**
12: **if** $n > 2$ **AND** $b_S < b_L$ **then**
13: $\lambda_S \leftarrow \lfloor \lambda_{\text{Init}} \left(\frac{1}{2}\lambda/\lambda_{\text{Init}}\right)^{\text{u}(0,1)} \rfloor$
14: $\mu_S \leftarrow \lceil \lambda_S\vartheta \rceil$
15: $[\boldsymbol{y}, f, \nu, evals] \leftarrow \epsilon$MAg-ES$(\mu_S, \lambda_S, \sigma_{\text{Init}}, T, \theta_r)$
16: $b_S \leftarrow b_S + evals$
17: $n_S \leftarrow n_S + 1$
18: **else**
19: $[\boldsymbol{y}, f, \nu, evals] \leftarrow \epsilon$MAg-ES$(\mu, \lambda, \sigma_{\text{Init}}, T, \theta_r)$
20: $b_L \leftarrow b_L + evals$
21: **end if**
22: $budget_{\text{used}} \leftarrow budget_{\text{used}} + evals$
23: **if** $\boldsymbol{y} \leq_{lex} \boldsymbol{y}_{\text{bsf}}$ **then** $\boldsymbol{y}_{\text{bsf}} \leftarrow \boldsymbol{y}$, $f_{\text{bsf}} \leftarrow f$, $\nu_{\text{bsf}} \leftarrow \nu$
24: **end if**
25: **end while**
26: **return** $[\boldsymbol{y}_{\text{bsf}}, f_{\text{bsf}}, \nu_{\text{bsf}}]$

Until one of the three termination conditions (TC1), (TC2), or (TC3) is satisfied, the ϵMAg-ES is repeatedly generating new offspring populations. To this end, λ offspring are generated out of the parental recombinant of the previous generation in the offspring procreation loop of each generation g. The corresponding mutation vector $\boldsymbol{d}_l^{(g)}$ of each offspring is obtained by multiplication of the transformation matrix $\boldsymbol{M}^{(g)}$ and a vector $\boldsymbol{z}_l^{(g)}$ with standard normally distributed components, lines 17 and 18 of Alg. 2. An offspring is generated in line 19 by adding the product of the mutation strength $\sigma^{(g)}$ and $\boldsymbol{d}^{(g)}$ to the recombinant $\boldsymbol{y}^{(g)}$ of the previous generation. Offspring individuals that are generated outside the box constraints, are component-wise reflected into the box. In accordance with Eq. (5), the reflection is denoted by `KeepRange(.)` within the pseudo code. It is performed before each function evaluation (see line 20 and line 25).

Taking into account [15], an additional repair step (lines 22 to 28) is performed with probability θ_p in generations g that are multiples of the dimensionality N. If applicable, infeasible offspring candidate solutions $\boldsymbol{y}_l^{(g)}$ are repaired based on the approximated gradients of the constraint functions. The repair step requires N function evaluations per execution plus one single evaluation of the repaired candidate solution. It is expected to reduce the constraint violation of a candidate solution and will potentially guide the search into a beneficial region of the search space, or promote the final step towards the optimizer, respectively. This operation called `GradientBasedRepair(.)` does not guarantee feasibility in a single step. Hence, it is repeated up to θ_r times.

Offspring candidate solutions $\boldsymbol{y}_l^{(g)}$ that have been adjusted by `KeepRange(.)` or `GradientBasedRepair(.)` will differ from the originally sampled $\bar{\boldsymbol{y}}$. The corresponding mutation vector $\boldsymbol{d}_l^{(g)}$ and $\boldsymbol{z}_l^{(g)}$ of $\boldsymbol{y}_l^{(g)}$ have to be readjusted to take into account the correct quantities in the transformation matrix update (line 38). This back-calculation is executed in lines 30 and 31 of Alg. 2. While the adjustment of $\boldsymbol{d}_l^{(g)}$ is simple, the correction of $\boldsymbol{z}_l^{(g)}$ involves the inverse of the transformation matrix $\boldsymbol{M}^{(g)}$. Since $\boldsymbol{M}^{(g)}$ might turn out to be a singular matrix, its pseudo inverse $\boldsymbol{M}^{-1}$ is used in line 31. It is computed in line 12 at the beginning of each generation.

In the event that the $\boldsymbol{M}^{(g)}$ update results in a matrix that is ill-suited for the determination of the pseudo inverse $\boldsymbol{M}^{-1}$, a reinitialization step is applied in line 13 to 15. It prevents the algorithm from breaking off due to numerical instabilities. Using the implementation in [9], $\boldsymbol{M}^{(g)}$ is reset to $\boldsymbol{I}$, and $\boldsymbol{p}_\sigma^{(g)}$ to $\mathbf{1} \in \mathbb{R}^N$, if applicable. This way, the adaptation mechanism will start anew in the current location of the search space.

After the computation of all λ offspring, the best offspring is compared to the best-found solution so far $\boldsymbol{y}_{\text{bsf}}$ in line 34. In line 36, the parental recombinant $\boldsymbol{y}^{(g)}$ is updated. The update involves the selection of the best μ mutation vectors with respect to $\leq_\epsilon$. Note that in the context of the updates (line 36 to 38), those vectors $\boldsymbol{z}_l^{(g)}$, and $\boldsymbol{d}_l^{(g)}$, that contribute to the mth best candidate solution $\boldsymbol{y}_{m;\lambda}$ are considered the mth best,

Algorithm 2 Pseudo code of a modified version of ϵMAg-ES [9]. Modifications are highlighted by grey boxes.

1: **Initialize:** μ, λ, T, $\boldsymbol{w}$, $\boldsymbol{p}_\sigma^{(0)} \leftarrow \mathbf{1}$, $\boldsymbol{M}^{(0)} \leftarrow \boldsymbol{I}$, $g \leftarrow 0$,
2: $\sigma^{(0)} \leftarrow \sigma_{\text{Init}}$, $\sigma_{\max} \leftarrow \max(\check{\boldsymbol{y}} - \hat{\boldsymbol{y}})/2$
3: **for** $j \leftarrow 1\colon \lambda$ **do** ▷ Sample initial population
4: $\boldsymbol{y}_j \leftarrow \check{\boldsymbol{y}} + (\hat{\boldsymbol{y}} - \check{\boldsymbol{y}}) \circ \boldsymbol{u}(0,1)$
5: $\mathcal{P} \leftarrow \mathcal{P} \cup \{\boldsymbol{y}_j\}$
6: **end for**
7: $\epsilon^{(0)} \leftarrow \text{median}\left(\{\nu(\boldsymbol{y}_k), k = 1, \ldots, \lambda\}\right) \cdot \chi_{T>0}$ acc. to $\leq_{lex}$
8: $evals \leftarrow \lambda$
9: $\boldsymbol{y}^{(0)} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i;\lambda}$ according to $\leq_\epsilon$
10: $\boldsymbol{y}_{\text{bsf}} \leftarrow \boldsymbol{y}_{1;\lambda}$
11: **repeat**
12: $\boldsymbol{M}^{-1} \leftarrow \texttt{PseudoInverse}(\boldsymbol{M}^{(g)})$
13: **if** $\texttt{PseudoInverse}(\boldsymbol{M}^{(g)})$ fails **then**
14: $\boldsymbol{M}^{(g)} \leftarrow \boldsymbol{I}, \quad \boldsymbol{p}_{\boldsymbol{\sigma}}^{(g)} \leftarrow \mathbf{1}$
15: **end if**
16: **for** $l \leftarrow 1\colon \lambda$ **do** ▷ Create offspring population
17: $\boldsymbol{z}_l^{(g)} \leftarrow \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
18: $\boldsymbol{d}_l^{(g)} \leftarrow \boldsymbol{M}^{(g)} \boldsymbol{z}_l^{(g)}$
19: $\bar{\boldsymbol{y}} \leftarrow \boldsymbol{y}^{(g)} + \sigma^{(g)} \boldsymbol{d}_l^{(g)}$
20: $\boldsymbol{y}_l^{(g)} \leftarrow \texttt{KeepRange}(\bar{\boldsymbol{y}})$
21: $evals \leftarrow evals + 1$
22: **if** $\text{mod}(g, N) = 0$ **AND** $u(0,1) < \theta_p$ **then** $h \leftarrow 1$
23: **while** $h \leq \theta_r$ **AND** $\nu(\boldsymbol{y}_l^{(g)}) > 0$ **do** $h \leftarrow h + 1$
24: $\tilde{\boldsymbol{y}} \leftarrow \texttt{GradientBasedRepair}(\boldsymbol{y}_l^{(g)})$
25: $\boldsymbol{y}_l^{(g)} \leftarrow \texttt{KeepRange}(\tilde{\boldsymbol{y}})$
26: $evals \leftarrow evals + N + 1$
27: **end while**
28: **end if**
29: **if** $\bar{\boldsymbol{y}} \neq \boldsymbol{y}_l^{(g)}$ **then**
30: $\boldsymbol{d}_l^{(g)} \leftarrow \left(\boldsymbol{y}_l^{(g)} - \boldsymbol{y}^{(g)}\right)/\sigma^{(g)}$
31: $\boldsymbol{z}_l^{(g)} \leftarrow \boldsymbol{M}^{-1} \boldsymbol{d}_l^{(g)}$
32: **end if**
33: **end for**
34: **if** $\boldsymbol{y}_{1;\lambda}^{(g)} \leq_\epsilon \boldsymbol{y}_{\text{bsf}}$ **then** $\boldsymbol{y}_{\text{bsf}} \leftarrow \boldsymbol{y}_{1;\lambda}^{(g)}$
35: **end if**
36: $\boldsymbol{y}^{(g+1)} \leftarrow \boldsymbol{y}^{(g)} + \sigma^{(g)} \sum_{i=1}^{\mu} w_i \boldsymbol{d}_{i;\lambda}^{(g)}$
37: $\boldsymbol{p}_\sigma^{(g+1)} \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{\mu_w c_\sigma (2 - c_\sigma)} \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i;\lambda}^{(g)}$
38: $\boldsymbol{M}^{(g+1)} \leftarrow \boldsymbol{M}^{(g)} + \frac{c_1}{2}\boldsymbol{M}^{(g)} \left(\boldsymbol{p}_\sigma^{(g)} (\boldsymbol{p}_\sigma^{(g)})^\top - \boldsymbol{I}\right) \ldots$
$+ \frac{c_\mu}{2}\boldsymbol{M}^{(g)} \left(\sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i;\lambda}^{(g)} (\boldsymbol{z}_{i;\lambda}^{(g)})^\top - \boldsymbol{I}\right)$
39: $\sigma^{(g+1)} \leftarrow \min\left(\sigma^{(g)} \exp\left[\frac{c_\sigma}{2}\left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|^2}{N} - 1\right)\right], \sigma_{\max}\right)$
40: **if** $g < T$ **AND** $FR_{\epsilon^{(g)}} > \theta_{\text{FR}}$ **then**
41: $\epsilon^{(g+1)} \leftarrow \epsilon^{(g)}(1 - \frac{g}{T})^2$
42: **else if** $g < T$ **AND** $FR_{\epsilon^{(g)}} \leq \theta_{\text{FR}}$ **then**
43: $\epsilon^{(g+1)} \leftarrow (1 + \theta_\epsilon)\epsilon^{(g)}$
44: **else** $\epsilon^{(g+1)} \leftarrow 0$
45: **end if**
46: $g \leftarrow g + 1$
47: **until** termination in acc. with (TC1), (TC2), or (TC3)
48: **return** $[\boldsymbol{y}_{\text{bsf}}, f(\boldsymbol{y}_{\text{bsf}}), \nu(\boldsymbol{y}_{\text{bsf}}), evals]$

i.e. $\boldsymbol{z}_{m;\lambda}$, and $\boldsymbol{d}_{m;\lambda}$, respectively.

The ϵMAg-ES adapts its search path $\boldsymbol{p}_\sigma^{(g)}$ in line 37. $\|\boldsymbol{p}_\sigma^{(g)}\|$ indicates whether the mutation strength $\sigma^{(g)}$ should be decreased or increased in the next generation. The search path also contributes to the transformation matrix $\boldsymbol{M}^{(g)}$ update performed in line 38. The corresponding strategy parameters are chosen according to the recommendations in [8]. The mutation strength $\sigma^{(g)}$ is then updated in line 39. It is bounded from above by the parameter $\sigma_{\max}$. Motivated by empirical observations, $\sigma_{\max}$ was set to a fixed value for the CEC 2018 competition in [9]. The modified ϵMAg-ES initially computes $\sigma_{\max}$ on the basis of the problem-specific box constraints in line 2 and thus omits additional parameter tuning.

Finally, the $\epsilon^{(g)}$-threshold is controlled in lines 40 to 45. The $\epsilon^{(g)}$ control differs substantially from [9]. Instead of simply reducing the threshold towards zero, it includes a rule for raising the ϵ-level. The idea to control $\epsilon^{(g)}$ is inspired by [13], but tailored to ES characteristics and using less strategy parameters. That is, the ϵ-level, i.e. the magnitude of constraint relaxation, is decreased as long as the ratio of ϵ-feasible solutions $FR_{\epsilon^{(g)}}$ in the parental population of size μ exceeds a minimum value θ_{FR}. Otherwise, the current ϵ-level is increased by a factor of $1 + \theta_\epsilon$ with the intention of giving the ES more flexibility to find better points through stronger relaxation. Still, after a predefined number of generations T, the ϵ-level is ultimately set to zero in late search phases. This control procedure adopts the accuracy with which the $\leq_\epsilon$ order relation distinguishes feasible from infeasible candidate solutions (see Eq. (6) below).

A. Constraint handling approaches

This subsection briefly recaps the constraint handling methods used in ϵMAg-ES and BP-ϵMAg-ES.

a) Treatment of box-constraints: The satisfaction of the box-constraints is ensured for every single candidate solution $\boldsymbol{y} \in \mathbb{R}^N$ prior to its evaluation. Regarding the upper and lower parameter bounds ($\hat{\boldsymbol{y}}, \check{\boldsymbol{y}} \in \mathbb{R}^N$) of the constrained function, each exceeding component $i \in \{1, \ldots, N\}$ of $\boldsymbol{y}$ is reflected into the box according to

$$y_i = \begin{cases} \check{y}_i + \left((\check{y}_i - y_i) - \left\lfloor \frac{\check{y}_i - y_i}{\omega_i} \right\rfloor \omega_i\right), & \text{if } y_i < \check{y}_i, \\ \hat{y}_i - \left((y_i - \hat{y}_i) - \left\lfloor \frac{y_i - \hat{y}_i}{\omega_i} \right\rfloor \omega_i\right), & \text{if } y_i > \hat{y}_i, \\ y_i, & \text{else,} \end{cases} \tag{5}$$

with component-wise distance $\omega_i = (\hat{y}_i - \check{y}_i)$ between $\hat{\boldsymbol{y}}$, and $\check{\boldsymbol{y}}$. In Alg. 2, the approach is denoted by $\texttt{KeepRange}(.)$.

b) The ϵ-level ordering: The selection operator within ES needs to rank the generated offspring individuals of a single generation. In that regard, feasible solutions are considered superior to infeasible solutions. The usual *lexicographic ordering* primarily ranks two candidate solutions according to their constraint violations and secondly with respect to their objective function values.

Except in line 7, the ϵMAg-ES uses another ordering relation: the ϵ-level ordering introduced by [15]. It represents a relaxation that enables the ES to treat infeasible candidate

solutions with constraint violation below a specific $\epsilon^{(g)}$ level as feasible. A candidate solution $\boldsymbol{y}$ is called ϵ-feasible, if its constraint violation $\nu(\boldsymbol{y})$ does not exceed the $\epsilon^{(g)}$ threshold in generation g. Originally, the $\epsilon^{(g)}$-level is continuously reduced to zero with the number of generations. Hence, the strategy is able to move outside the feasible region within the early phase of the search which can potentially support the convergence to the optimizer $\boldsymbol{y}^*$. Conversely, the search outside the feasible region might misdirect the algorithm in case of objective functions that are monotonic in at least one axial direction [5].

Consider two candidate solutions of problem (1), $\boldsymbol{y}_i \in \mathbb{R}^N$ and $\boldsymbol{y}_j \in \mathbb{R}^N$, as well was their corresponding pairs of objective function and constraint violation values $(f_i, \nu_i) = (f(\boldsymbol{y}_i), \nu(\boldsymbol{y}_i))$, and (f_j, ν_j), respectively. The ϵ-level order relation denoted by $\leq_\epsilon$ is then defined by

$$\boldsymbol{y}_i \leq_\epsilon \boldsymbol{y}_j \Leftrightarrow \begin{cases} f_i \leq f_j, & \text{if} \quad (\nu_i \leq \epsilon^{(g)}) \wedge (\nu_j \leq \epsilon^{(g)}), \\ f_i \leq f_j, & \text{if} \quad \nu_i = \nu_j, \\ \nu_i < \nu_j, & \text{otherwise.} \end{cases} \tag{6}$$

Hence, candidate solutions are compared according to the following criteria: Two ϵ-feasible solutions are ranked with respect to their objective function values. Two ϵ-infeasible solutions are ordered on the basis of their constraint violations. Ties are resolved by considering the objective function values.

In line 7 of Alg. 2, $\epsilon^{(0)}$ is determined as the median constraint violation of the initial population. During the search, it is adjusted with each generation until it is set to zero after a fixed number of generations T. For $\epsilon \equiv 0$, the ϵ level ordering equals the *lexicographic ordering* which is denoted by $\leq_{lex}$.

c) Gradient-based repair: In addition to ϵ-level order relation, the ϵMAg-ES erratically repairs infeasible candidate solutions based on the gradient of the constraint functions. In these situations, the Jacobian is approximated by use of finite differences. The consumed function evaluations have to be accounted properly. If the correction is not resulting in a feasible solution, the gradient-based repair step is repeated at most θ_r times. In case the strategy cannot find a feasible solution after θ_r repair steps, the last infeasible candidate solution is considered as the new offspring candidate solution. A detailed description of the repair approach is provided in [9].

IV. Experiments

This section recaps the experimental setup and provides the parameter settings used by BP-ϵMAg-ES, Alg. 1. All experiments are carried out in accordance with the specifications of the CEC competition provided in [7][1]. Consequently, the algorithm executes 25 independent runs on each constrained real-world problem $i \in \{1, \ldots, 57\}$. During the runs, equality constraints are considered to be satisfied if the absolute deviation is below the error margin of $\delta = 10^{-4}$, cf. Eq. (4). To fully provide the required statistics, the algorithm returns the objective function and constrained violation (2) values of the best candidate solution found after having consumed 100% of the allowed budget of function evaluation. The ϵMAg-ES uses the standard parameters recommended for the MA-ES in [8]. The recombination weights are

$$w_i = \frac{\ln(\mu + 0.5) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu + 0.5) - \ln j)}, \text{ for } i \in \{1, \ldots, \mu\} \tag{7}$$

and the corresponding effective population size is given as $\mu_w = 1/\sum_{i=1}^{\mu} w_i^2$. The learning rates of the mutation strength, the search path, and the transformation matrix update are

$$c_\sigma = \frac{\mu_w + 2}{N + \mu_w + 5}, \quad c_1 = \frac{2}{(N + 1.3)^2 + \mu_w}, \text{ and} \tag{8}$$

$$c_\mu = \min\left[1 - c_1, \frac{2(\mu_w - 2 + 1/\mu_w)}{(N + 2)^2 + \mu_w}\right] \tag{9}$$

In contrast to [9], the initial population size parameters $\lambda = 4 + \lfloor 3\log(N) \rfloor$ and $\mu = \lceil \lambda\vartheta \rceil$ with $\vartheta = 1/3$ are considered in this paper. This choice is motivated by the investigations in [10] and by the use of the restart scheme that gradually elevates the population in one of the branches.

Except for the cases specified in line 8, Alg. 1, the ϵ-threshold of $\leq_\epsilon$ is adopted during the first $T = 500$ generations using $\theta_{\text{FR}} = 0.2$ and $\theta_\epsilon = 0.1$, see lines 40–45 of Alg. 2. The gradient-based repair is applied at most $\theta_r = 3$ times with probability $\theta_p = 0.2$ every Nth generation. As initial mutation strength $\sigma_{\text{Init}} = 1$ is used in the BP-ϵMAg-ES.

Table I
PC Configuration and algorithm complexity

PC:	Intel Haswell Desktop
CPU:	Intel Core i7-4770 3.40GHz×8
RAM:	16 GB
OS / Language:	Linux / Matlab (2018b)
Algorithm:	BP-ϵMAg-ES

Computational complexity

Algorithm	$T1(s)$	$T2(s)$	$(T2-T1)/T1$
BP-ϵMAg-ES	10.39	15.13	0.46

V. Results

As requested in [7], information on the PC configuration and the measured computational complexity of the BP-ϵMAg-ES are presented in Table I. There, $T1$ represents the computation time of 10^5 objective function evaluations of a single candidate solution averaged over all 57 test functions. On the other hand, $T2$ refers to the average computation time needed by the complete BP-ϵMAg-ES for 10^5 function evaluations. The algorithm complexity is identified with the relative difference $(T2 - T1)/T1$ of these quantities.

The final performance statistics of the BP-ϵMAg-ES on the constrained real-world benchmarks [6] for the CEC 2020 competition are presented in Tables II to V. Each table displays the final results after having consumed 100% of the function evaluations budget. They have been experimentally obtained in 25 independent algorithm runs on all 57 constrained real-world problems. The detailed results, i.e. measured in steps

[1]Note that [6] initially specified larger function evaluation budgets (for $N < 50$). In a late revision the budget was reduced by half ($N \leq 30$), and by one third ($30 < N \leq 50$), respectively. Yet, the final algorithm results provided have not been revised accordingly. To ensure comparability to the published results in [6], the results presented in Sec. V are based on the original budgets. Of course, the algorithm results submitted to the CEC competition are prepared using the budgets defined in the latest version of [7].

of 10% of the total budget, are submitted to the CEC 2020 competition in electronic form.

The performance statistics involve the objective function values f and the constraint violation values ν of the best, median, and worst candidate solutions found in all 25 independent algorithm runs on each single constrained real-world problem. Hence, the final best-found solutions are lexicographically ranked, i.e. primarily according to their constrained violations, and secondly with respect to their objective function values.

The mean objective function value and the mean constraint violation are also displayed together with the corresponding standard deviations. In addition, the tables give information on the feasibility rate FR and on the tuple c. The former (FR) is obtained as the ratio of the number of algorithm runs in which at least one feasible solution is found and the total number of algorithm runs. The latter provides a triplet the number of unsatisfied constraints with constraint violation larger than 1, in between 1 and 10^{-2}, and below 10^{-2}, respectively.

The BP-ϵMAg-ES found feasible solutions on 50 of 57 constrained real-world problems. A feasible median solution could be computed in 49 cases. A feasibility rate of $FR = 100\%$ was obtained on 44 benchmark problems. Only on seven problems, no feasible solution was found at all. That is, problems RC07, RC51, RC52, and RC54 to RC57, present the biggest challenge for the BP-ϵMAg-ES among the constrained real-world benchmarks. On five problems (RC25, RC35, RC36, RC45, and RC50) the BP-ϵMAg-ES even improves the best-known feasible solution (see Table 3 in [6]).

Comparing the BP-ϵMAg-ES results to the results of the original ϵMAg-ES provided in [6], one observes reasonable performance improvements in terms of an increased feasibility rate or mean objective function values on 22 constrained problems, e.g. RC02, RC06, or problems RC35 to RC40. On another 26 problems the performances of both approaches are comparable. The results on the remaining 9 benchmark functions reveal slight performance degradations of the BP-ϵMAg-ES over the original ϵMAg-ES approach without restarts. In comparison with the two Differential Evolution (DE) variants displayed in [6], the BP-ϵMAg-ES is able to realize improvements on 24 real-world benchmarks.

VI. Discussion

The paper introduces an advancement of the ϵMAg Evolution Strategy designed for the CEC 2020 competition on constrained real-world optimization problems. The BP-ϵMAg-ES combines the Matrix Adaptation Evolution Strategy for unconstrained optimization with well-known constraint handling techniques and a restart scheme capable of efficiently using the total budget of function evaluations. The approach exhibits great performance on most of the test problems. The BP-ϵMAg-ES realizes a feasibility rate of 100% on 44 real-world problems. Only 7 problems could never be solved satisfactorily. That represents an improvement over the algorithms tested in [6] including the original ϵMAg-ES version [9]. Many of the ϵMAg-ES results could be substantiated or even refined. The BiPop approach manages to find feasible solutions more regularly and reduces the scattering of the results over the 25 algorithm runs. Further investigations will be concerned with the evaluation of the impact of the termination conditions on the performance. Premature abortion due to (TC2) may considerably affect the effectiveness. Hence, the best termination criteria still need to be determined. Also, questions regarding the sensitivity to other parameter settings will have to be answered in a more detailed study. While BP-ϵMAg-ES always assumes a black-box setting, the competition allows to treat the constraints as white box [7]. Hence, utilizing constraint information may offer further room for improvements.

References

[1] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006," 2006, Technical Report, Nanyang Technological University, Singapore, https://github.com/P-N-Suganthan.

[2] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," 2010, Technical Report, Nanyang Technological University, Singapore, https://github.com/P-N-Suganthan/CEC2010-Constrained.

[3] G. H. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Single Objective Real-Parameter Optimization," 2017, Technical Report, Nanyang Technological University, Singapore. [Online]. Available: https://github.com/P-N-Suganthan/CEC2017

[4] D. Brockhoff, N. Hansen, T. Tušar, O. Mersmann, P. R. Sampaio, A. Auger, and A. Atamna *et al.*, "COCO documentation repository." [Online]. Available: http://github.com/numbbo/coco-doc

[5] M. Hellwig, P. Spettel, and H.-G. Beyer, "Comparison of contemporary evolutionary algorithms on the rotated klee-minty problem," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19*. New York, NY, USA: ACM, 2019.

[6] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm and Evolutionary Computation*, p. 100693, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650219308946

[7] ——, "Guidelines for Real-World Single-Objective Constrained Optimisation Competition," Tech. Rep., 2019, https://github.com/P-N-Suganthan/2020-RW-Constrained-Optimisation/blob/master/Guidelines_Real_World_Constrained.pdf.

[8] H.-G. Beyer and B. Sendhoff, "Simplify your covariance matrix adaptation evolution strategy," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 746–759, Oct 2017.

[9] M. Hellwig and H.-G. Beyer, "A matrix adaptation evolution strategy for constrained real-parameter optimization," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. Rio de Janeiro, Brazil: IEEE, 2018.

[10] ——, "Analyzing design principles for competitive evolution strategies in constrained search spaces," *Transactions on Evolutionary Learning and Optimization*, vol. 1, 2020, (submitted).

[11] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed," in *GECCO '09*, 2009.

[12] I. Loshchilov, "CMA-ES with restarts for solving CEC 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 369–376.

[13] Z. Fan, Y. Fang, W. Li, Y. Yuan, Z. Wang, and X. Bian, "LSHADE44 with an Improved ϵConstraint-Handling Method for Solving Constrained Single-Objective Optimization Problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8.

[14] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, March 2003.

[15] T. Takahama and S. Sakai, "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, pp. 1–9.

Table II

FINAL PERFORMANCE STATISTICS OVER 25 INDEPENDENT ALGORITHM RUNS ON THE INDUSTRIAL CHEMICAL PROCESS (RC01–RC07) AND ON THE PROCESS SYNTHESIS & DESIGN (RC08–RC14) PROBLEMS .

Problem		**RC01**	**RC02**	**RC03**
Best	f	1.8932E+02	7.0490E+03	-1.4272E+02
	ν	0	0	0
Median	f	1.8936E+02	7.0490E+03	7.6269E+01
	ν	0	0	0
Mean	f	1.8945E+02	7.0490E+03	-3.4578E+00
	ν	0	0	0
Worst	f	1.9005E+02	7.0490E+03	2.4932E+02
	ν	0	0	0
Std	f	1.9648E-01	8.1165E-08	1.1900E+02
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC04**	**RC05**	**RC06**
Best	f	-3.8792E-01	-3.5674E+02	1.9543E+00
	ν	0	0	0
Median	f	-3.8730E-01	-1.5944E+02	2.5398E+00
	ν	0	0	0
Mean	f	-3.8723E-01	-1.7769E+02	2.3059E+00
	ν	0	0	9.3095E-04
Worst	f	-3.8572E-01	-7.7454E+01	2.0895E+00
	ν	0	0	2.1343E-02
Std	f	4.9182E-04	6.3134E+01	2.4938E-01
	ν	0	0	4.2614E-03
FR		100	100	68
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC07**	**RC08**	**RC09**
Best	f	1.9630E+00	2.0000E+00	2.5577E+00
	ν	1.3133E-02	0	0
Median	f	1.7735E+00	2.0000E+00	2.5577E+00
	ν	4.7908E-02	0	0
Mean	f	1.9696E+00	2.0000E+00	2.5577E+00
	ν	1.2636E-01	0	0
Worst	f	1.9958E+00	2.0000E+00	2.5577E+00
	ν	6.5835E-01	0	0
Std	f	2.5311E-01	0	1.3597E-15
	ν	1.8664E-01	0	0
FR		0	100	100
c		(0,7,13)	(0,0,0)	(0,0,0)
Problem		**RC10**	**RC11**	**RC12**
Best	f	1.0765E+00	1.0276E+02	2.9248E+00
	ν	0	0	0
Median	f	1.0765E+00	1.0878E+02	2.9248E+00
	ν	0	6.5604E-141	0
Mean	f	1.0765E+00	1.1043E+02	2.9857E+00
	ν	0	3.8615E-29	0
Worst	f	1.0765E+00	9.9949E+01	3.7749E+00
	ν	0	9.6535E-28	0
Std	f	4.5325E-16	5.4879E+00	1.7560E-01
	ν	0	1.9307E-28	0
FR		100	28	100
c		(0,0,0)	(0,0,2)	(0,0,0)
Problem		**RC13**	**RC14**	
Best	f	2.6887E+04	5.8505E+04	
	ν	0	0	
Median	f	2.6887E+04	5.8505E+04	
	ν	0	0	
Mean	f	2.6887E+04	5.8505E+04	
	ν	0	0	
Worst	f	2.6887E+04	5.8505E+04	
	ν	0	0	
Std	f	1.1139E-11	7.3322E-06	
	ν	0	0	
FR		100	100	
c		(0,0,0)	(0,0,0)	

Table III

FINAL PERFORMANCE STATISTICS OVER 25 INDEPENDENT ALGORITHM RUNS ON THE MECHANICAL ENGINEERING PROBLEMS (RC15–RC29). FOR THE RESULTS ON RC30–RC33 REFER TO TABLE IV.

Problem		**RC15**	**RC16**	**RC17**
Best	f	2.9944E+03	3.2213E-02	1.2665E-02
	ν	0	0	0
Median	f	2.9944E+03	3.7255E-02	1.2665E-02
	ν	0	0	0
Mean	f	2.9944E+03	4.3887E-02	1.2665E-02
	ν	0	0	0
Worst	f	2.9944E+03	1.1517E-01	1.2665E-02
	ν	0	0	0
Std	f	4.6413E-13	1.9276E-02	2.2500E-11
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC18**	**RC19**	**RC20**
Best	f	6.0597E+03	1.6702E+00	2.6390E+02
	ν	0	0	0
Median	f	6.0597E+03	1.6702E+00	2.6390E+02
	ν	0	0	0
Mean	f	6.0671E+03	1.6702E+00	2.6390E+02
	ν	0	0	0
Worst	f	6.0905E+03	1.6702E+00	2.6390E+02
	ν	0	0	0
Std	f	1.3431E+01	4.5325E-17	0
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC21**	**RC22**	**RC23**
Best	f	2.3524E-01	5.2628E-01	1.6070E+01
	ν	0	0	0
Median	f	2.3524E-01	5.2815E-01	1.6070E+01
	ν	0	0	0
Mean	f	2.3524E-01	5.2884E-01	1.6070E+01
	ν	0	0	0
Worst	f	2.3524E-01	5.3287E-01	1.6070E+01
	ν	0	0	0
Std	f	1.1331E-16	1.8187E-03	1.6790E-14
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC24**	**RC25**	**RC26**
Best	f	2.5439E+00	1.6161E+03	3.9151E+01
	ν	0	0	0
Median	f	2.5473E+00	1.6222E+03	4.6372E+01
	ν	0	0	0
Mean	f	2.5499E+00	1.6499E+03	4.7945E+01
	ν	0	0	8.3188E-05
Worst	f	2.5757E+00	1.7673E+03	5.0778E+01
	ν	0	0	2.0797E-03
Std	f	7.4647E-03	4.7396E+01	5.2664E+00
	ν	0	0	4.1594E-04
FR		100	100	96
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC27**	**RC28**	**RC29**
Best	f	5.2445E+02	1.4614E+04	2.9649E+06
	ν	0	0	0
Median	f	5.2445E+02	1.4614E+04	2.9649E+06
	ν	0	0	0
Mean	f	5.2472E+02	1.4615E+04	2.9649E+06
	ν	0	0	0
Worst	f	5.3057E+02	1.4633E+04	2.9649E+06
	ν	0	0	0
Std	f	1.2225E+00	3.7925E+00	1.4258E-09
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)

Table IV
FINAL PERFORMANCE STATISTICS OVER 25 INDEPENDENT ALGORITHM RUNS ON THE MECHANICAL ENGINEERING PROBLEMS (RC30–RC33) AND ON THE POWER SYSTEMS PROBLEMS (RC34–RC44).

Problem		**RC30**	**RC31**	**RC32**
Best	f	2.6139E+00	0	-3.0666E+04
	ν	0	0	0
Median	f	2.6139E+00	0	-3.0666E+04
	ν	0	0	0
Mean	f	2.6139E+00	0	-3.0666E+04
	ν	0	0	0
Worst	f	2.6139E+00	0	-3.0666E+04
	ν	0	0	0
Std	f	1.0386E-13	0	3.7130E-12
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC33**	**RC34**	**RC35**
Best	f	2.6395E+00	1.9266E-01	8.3277E-02
	ν	0	0	0
Median	f	2.6449E+00	4.7804E-01	9.3446E-02
	ν	0	0	0
Mean	f	2.6457E+00	5.0622E-01	9.7245E-02
	ν	0	0	0
Worst	f	2.6552E+00	8.7114E-01	1.3920E-01
	ν	0	0	0
Std	f	4.5003E-03	1.9468E-01	1.3134E-02
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC36**	**RC37**	**RC38**
Best	f	6.5159E-02	3.1889E-02	3.3550E+00
	ν	0	0	0
Median	f	1.0595E-01	4.1814E-01	5.2696E+00
	ν	0	0	0
Mean	f	1.0745E-01	4.7536E-01	5.7540E+00
	ν	0	0	0
Worst	f	1.7258E-01	1.5526E+00	9.0946E+00
	ν	0	0	0
Std	f	2.3787E-02	4.1428E-01	1.4384E+00
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC39**	**RC40**	**RC41**
Best	f	3.6914E+00	1.1671E-18	7.6732E-21
	ν	0	0	0
Median	f	6.4649E+00	1.2574E-15	2.2331E-20
	ν	0	0	0
Mean	f	6.9625E+00	4.8998E-11	3.6146E-20
	ν	0	0	0
Worst	f	1.0617E+01	6.4324E-10	1.2044E-19
	ν	0	0	0
Std	f	1.7710E+00	1.4480E-10	3.2582E-20
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC42**	**RC43**	**RC44**
Best	f	1.7721E-01	1.2413E-01	-6.1271E+03
	ν	0	0	0
Median	f	5.9240E+01	3.8803E+01	-5.9674E+03
	ν	0	0	0
Mean	f	4.4066E+01	4.3098E+01	-5.9654E+03
	ν	2.7008E-02	5.1356E-04	0
Worst	f	-2.0399E+00	2.2633E+01	-5.8194E+03
	ν	2.8314E-01	1.2839E-02	0
Std	f	3.1818E+01	2.3633E+01	8.9263E+01
	ν	6.9332E-02	2.5678E-03	0
FR		76	96	100
c		(0,0,0)	(0,0,0)	(0,0,0)

Table V
FINAL PERFORMANCE STATISTICS OVER 25 INDEPENDENT ALGORITHM RUNS ON THE POWER ELECTRONIC (RC45–RC50) AND LIVE STOCK RATION OPTIMIZATION (RC51–RC57) PROBLEMS.

Problem		**RC45**	**RC46**	**RC47**
Best	f	3.6065E-02	2.3592E-02	1.1408E-02
	ν	0	0	0
Median	f	3.9926E-02	3.1118E-02	2.1222E-02
	ν	0	0	0
Mean	f	4.6045E-02	3.5846E-02	2.1246E-02
	ν	0	0	0
Worst	f	1.0114E-01	8.3595E-02	3.4599E-02
	ν	0	0	0
Std	f	1.7446E-02	1.4628E-02	5.1362E-03
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC48**	**RC49**	**RC50**
Best	f	1.6788E-02	1.0519E-02	1.4970E-02
	ν	0	0	0
Median	f	3.5294E-02	2.7958E-02	1.5797E-02
	ν	0	0	0
Mean	f	3.4488E-02	2.6114E-02	1.8026E-02
	ν	0	0	0
Worst	f	8.3839E-02	3.7347E-02	6.8336E-02
	ν	0	0	0
Std	f	1.5087E-02	7.6237E-03	1.0543E-02
	ν	0	0	0
FR		100	100	100
c		(0,0,0)	(0,0,0)	(0,0,0)
Problem		**RC51**	**RC52**	**RC53**
Best	f	4.4222E+03	3.7636E+03	5.6593E+03
	ν	3.0264E-03	0	2.4659E-04
Median	f	4.3540E+03	4.7342E+03	5.1442E+03
	ν	7.3361E-03	0	7.6991E-03
Mean	f	4.2331E+03	4.8245E+03	5.3356E+03
	ν	1.5850E-02	2.9673E-03	1.6343E-02
Worst	f	4.1360E+03	6.0512E+03	5.3852E+03
	ν	1.1500E-01	2.4198E-02	7.6665E-02
Std	f	1.5683E+02	6.7631E+02	2.7711E+02
	ν	2.3600E-02	6.0413E-03	1.8654E-02
FR		0	76	0
c		(0,1,0)	(0,0,0)	(0,3,1)
Problem		**RC54**	**RC55**	**RC56**
Best	f	3.6391E+03	7.8318E+03	1.4353E+04
	ν	2.1221E-03	2.4056E-03	4.2292E-03
Median	f	4.4356E+03	5.1067E+03	1.5549E+04
	ν	1.5963E-02	5.2585E-03	1.2621E-02
Mean	f	4.3174E+03	6.3419E+03	1.3031E+04
	ν	4.4050E-02	5.5204E-03	1.1893E-02
Worst	f	4.4653E+03	9.4135E+03	1.2804E+04
	ν	2.2344E-01	9.3768E-03	1.9663E-02
Std	f	1.0552E+03	1.2380E+03	1.6771E+03
	ν	5.7884E-02	1.6408E-03	4.0553E-03
FR		0	0	0
c		(0,2,1)	(0,1,3)	(0,3,1)
Problem		**RC57**		
Best	f	1.0164E+04		
	ν	8.3190E-04		
Median	f	5.3843E+03		
	ν	2.1675E-03		
Mean	f	6.6273E+03		
	ν	2.4252E-03		
Worst	f	5.9084E+03		
	ν	4.3802E-03		
Std	f	1.7481E+03		
	ν	8.2663E-04		
FR		0		
c		(0,1,1)		