

# Memetic Viability Evolution for Constrained Optimization

Andrea Maesani, Giovanni Iacca, *Member, IEEE*, and Dario Floreano, *Senior Member, IEEE*

**Abstract**—The performance of evolutionary algorithms can be heavily undermined when constraints limit the feasible areas of the search space. For instance, while Covariance Matrix Adaptation Evolution Strategy is one of the most efficient algorithms for unconstrained optimization problems, it cannot be readily applied to constrained ones. Here, we used concepts from Memetic Computing, i.e. the harmonious combination of multiple units of algorithmic information, and Viability Evolution, an alternative abstraction of artificial evolution, to devise a novel approach for solving optimization problems with inequality constraints. Viability Evolution emphasizes elimination of solutions not satisfying viability criteria, defined as boundaries on objectives and constraints. These boundaries are adapted during the search to drive a population of local search units, based on Covariance Matrix Adaptation Evolution Strategy, towards feasible regions. These units can be recombined by means of Differential Evolution operators. Of crucial importance for the performance of our method, an adaptive scheduler toggles between exploitation and exploration by selecting to advance one of the local search units and/or recombine them. The proposed algorithm can outperform several state-of-the-art methods on a diverse set of benchmark and engineering problems, both for quality of solutions and computational resources needed.

**Index Terms**—Constrained Optimization, Covariance Matrix Adaptation, Differential Evolution, Memetic Computing, Viability Evolution.

## I. INTRODUCTION

SEVERAL real-world optimization problems are characterized by the presence of one or more inequality constraints that limit the feasible region of the search space. When defined in a continuous domain, such problems can be formulated as:

$$\min f(\mathbf{x}), \text{ s.t. : } \begin{cases} l_i \leq x_i \leq u_i, & i = 1, 2, \dots, n \\ g_j(\mathbf{x}) \leq 0, & j = 1, 2, \dots, m \end{cases} \quad (1)$$

where  $f(\mathbf{x})$  is the objective (fitness) function to be optimized and  $\mathbf{x} \in \mathbb{R}^n$  is a vector of design variables  $[x_1, x_2, \dots, x_n]$ . The search space is delimited by box constraints defining the admissible range  $[l_i, u_i]$ ,  $l_i \in \mathbb{R}$ ,  $u_i \in \mathbb{R}$  of each variable  $x_i$ ,  $i \in 1, 2, \dots, n$ . Within the search space, the inequality

constraints  $g_j(\mathbf{x})$  determine the feasible region that contains all the solutions satisfying the constraints of the problem.<sup>1</sup>

Examples of constrained optimization problems can be found in many fields where physical, geometrical or resource requirements may limit the feasibility of the solutions. Due to the vast range of applications, constrained optimization has attracted the interest of a large part of the Computational Intelligence research community<sup>2</sup>. Pioneering studies can be traced back to research on Evolutionary Algorithms (EA), particularly Genetic Algorithms (GA) [1]–[4] and Evolution Strategies (ES) [5], while more recently Particle Swarm Optimization (PSO) has gained attention [6]–[11], likewise other Swarm Intelligence techniques [12]–[16]. Parallel research lines have explored the use of several constraint handling techniques [17]–[19], as adaptive penalty functions [20], [21], repair mechanisms for infeasible solutions [22], [23], stochastic ranking of solutions [24],  $\epsilon$ -constrained optimization [25], feasibility rules to rank solutions [26], and surrogate models [27]. Recent empirical studies analyzed the utility of retaining infeasible solutions during evolution [28], [29], whereas others have considered the use of multi-objective techniques where the constraint violations are minimized as separate objectives together with the problem objective function [30]–[33]. Importantly, competitions for constrained optimization organized in the context of the IEEE Congress on Evolutionary Computation (CEC) [34], [35] have finally provided standard benchmarks for comparing the performance of the various algorithms in the field.

Despite these advances, the increasing number of computationally intensive applications still need to be matched by computationally efficient algorithms. This is of utmost importance in contexts where evaluations are computationally expensive, as in the case of complex simulations, or where the optimization process has to be run in a limited time, as in the case of hardware-in-the-loop evolution. In these scenarios, optimization algorithms should ideally deliver high performance (in terms of quality of discovered solutions) using only a limited amount of function evaluations. However, state-of-the-art Computational Intelligence algorithms for constrained optimization typically require a large number of evaluations to

A. Maesani, G. Iacca and D. Floreano are with the Laboratory of Intelligent Systems, Institute of Microengineering, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 1015, Switzerland e-mail: {andrea.maesani, giovanni.iacca, dario.floreano}@epfl.ch

Manuscript received August 28, 2014; revised December 29, 2014 and March 11, 2015; accepted April 25, 2015.

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to [pubs-permissions@ieee.org](mailto:pubs-permissions@ieee.org).

<sup>1</sup>Constrained optimization problems can also include equality constraints defined as  $h_k(\mathbf{x}) = 0$ ,  $k = 1, 2, \dots, p$ , that can reduce the feasible areas down to zero-volume regions. Dedicated literature covers this class of problems proposing specific techniques for handling them. Handling equality constraints is out of the scope of this paper.

<sup>2</sup>A constantly updated list of references on the topic, maintained by Carlos A. Coello Coello, is available at: <http://www.cs.cinvestav.mx/~constraint/>.

converge, especially if the problem is high-dimensional, multi-modal and severely constrained.

Among the most effective algorithms for single objective optimization, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [36] has gained considerable attention in the last decade due to its ability of solving highly non-separable, ill-conditioned, and multi-modal functions. Although some attempts of extending CMA-ES to constrained optimization problems have been made [37]–[41], CMA-ES is not yet competitive on these types of problems: the self-adaptation of the algorithm's parameters is not suitable in constrained landscapes [42]. Moreover, due to the use of a single search distribution in CMA-ES, it is difficult to explore disconnected feasible areas unless restarts occur.

Arnold and Hansen [43] recently introduced in a (1+1)-CMA-ES [44] a novel covariance matrix adaptation rule. This rule exploits constraint violations to learn information and adapt the covariance matrix to decrease the likelihood of sampling other infeasible solutions. However, the method works only when an initial feasible solution is provided. In [45] we extended this method to allow its use also when started from infeasible solutions, by taking inspiration from Viability Evolution principles [46], [47]. Viability Evolution is an alternative abstraction of artificial evolution that operates by eliminating individuals not satisfying a set of criteria. These criteria, called viability boundaries, are defined on the problem objectives or constraints, and are adapted during evolution. Similar to what is done in a Viability Evolution algorithm, in [45] the viability boundaries defined on the constraints of the problem are relaxed or tightened to drive the search towards feasible areas. This method, called (1+1)-ViE-CMA-ES, or in short (1+1)-ViE, was also enriched with a novel mechanism to adapt the step-size based on information collected at each constraint violation. The method displayed excellent performance on unimodal constrained optimization problems but, since it is based on a version of CMA-ES that samples a single offspring per generation, its performance was quite poor on problems characterized by disconnected feasible areas or highly multi-modal fitness landscapes.

One possible way to partially overcome these limitations is to reformulate the rules for covariance matrix [43] and step-size adaptation [45] designed for (1+1)-CMA-ES into rules for  $(\mu, \lambda)$ -CMA-ES. However, this solution would still rely on a single search distribution, thus having limited capability of exploring disconnected feasible areas. An alternative solution is the use of multiple independent local search units made of (1+1)-ViE, that may potentially explore disconnected feasible regions. The main burden of this second option is represented by the need for an efficient algorithm that determines the optimal allocation of function evaluations among the local search units, as simply running all of them in parallel may result in a waste of computational resources. On the other hand, once an appropriate algorithmic scheme for balancing the function evaluations among the different local search units is found, it could be possible to re-use it also with different types of local searchers.

In this paper, we explored this latter idea by proposing a novel memetic computing approach, called *memetic*

*Viability Evolution* (mViE), that uses several local search units, constituted by (1+1)-ViE [45], that can be recombined by Differential Evolution (DE) [48] evolutionary operators. Differential Evolution was chosen as global search operator due to its intrinsic ability of adjusting the effect of mutations during the search. Such feature makes DE a good candidate for recombining the information learned by the local search units, while performing at the same time global exploration. This intuition was also supported by previous research [49] that showed through a large comparative study that DE tends to outperform several alternative meta-heuristics (such as PSO, GA, and ES) on constrained optimization problems. Furthermore, we improved the balance between exploration and exploitation by using an adaptive scheduler that allocates dynamically the function evaluations to either the local search units or to global search.

To assess mViE, we performed numerical experiments on thirteen CEC 2006 benchmark problems with inequality constraints [34], as well as four classical mechanical engineering design problems [3], [50]. We then compared our method against an extensive collection of state-of-the-art algorithms for constrained optimization. In the experiments, mViE showed consistent performance gains in terms of function evaluations needed to reach the optimum on almost all the tested problems.

The paper is organized as follows. Section II surveys the literature related to constrained optimization. Section III briefly describes (1+1)-CMA-ES and recent advances for the adaptation of covariance matrix and step-size. Moreover, this section discusses the introduction of Viability Evolution principles in CMA-ES. Section IV describes in details the proposed method. The experimental setup and the algorithmic parameter setting are discussed in section V, while the numerical results are presented in section VI. Finally, discussion and conclusions are presented in section VII.

## II. RELATED WORK: CONSTRAINED OPTIMIZATION

In this section we first review related literature on constrained optimization algorithms based on CMA-ES and DE, the building blocks of our proposed method mViE. Then, we summarize recent advances in memetic computing, focusing on approaches specifically designed for constrained problems.

### A. Methods based on CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [36] is considered nowadays the state-of-the-art in unconstrained single-objective optimization. In the presence of constraints, however, the step-size control used by CMA-ES to refine the search does not work properly [42], an effect known also in standard ES. Recent research efforts have been devoted to overcome this difficulty. The use of adaptive penalty functions was investigated in [38], where the weight of each constraint in the penalty function was modified according to the number of iterations during which that constraint was violated. Another penalty function has been proposed in [41], where the constraint violation of all the solutions in the population is used to scale the relative violation of each solution.

CMA-ES has been also integrated with ASCHEA, an approach proposed in [51] to adapt the tolerances on the equality constraints [39]. Other approaches rank individuals based on three independent rankings [52], namely objective function, constraint violation, and number of violated constraints, or use surrogate models to learn information about constraints [37], [40]. A repair mechanism was used in a problem-specific variant of CMA-ES for financial optimization [42].

Among the most effective approaches to date for CMA-ES-based constrained optimization, Arnold and Hansen [43] proposed a modification of (1+1)-CMA-ES [44] specifically designed for unimodal problems. Starting from a feasible solution, the algorithm maintains a low-pass filtered vector representing the direction of violations of each constraint and consequently uses this information to reduce the variance of the search ellipsoid along the detected direction of violation. In our previous study [45], we further improved the performance of this (1+1)-CMA-ES scheme, by collecting information on single constraint violations and using it to adapt the step-size.

### B. Methods based on Differential Evolution

Earlier research on DE-based constrained optimization considered the use of feasibility rules and diversity preservation mechanisms [53], [54] and the incorporation of domain knowledge [55]. With the introduction of the CEC benchmarks on constrained optimization [34], [35], that created a common environment for assessing the performance of novel constrained optimization algorithms, DE became a popular choice for solving constrained problems. A number of DE-based methods proposed for solving these benchmarks are now considered the state-of-the-art in evolutionary constrained optimization. Among these methods,  $\varepsilon$ -DE [56] ranks, by objective value, the solutions that are feasible or violate at most by an  $\varepsilon$ -value the constraints, while preferring feasible solutions over infeasible solutions, which are instead compared based on their constraint violation. Notably,  $\varepsilon$ -DE won both the CEC 2006 and 2010 competitions for constrained optimization. Competitive results were also obtained by two other variants of DE, namely MDE [57], which uses an *ad-hoc* mutation that incorporates information from both best and parent individuals, and SADE [58], a self-adaptive version of DE. Both algorithms employ the three feasibility rules presented in [26] for handling constraints. Later research on optimal DE parameter control [59], [60] led to devise robust self-adapting schemes, such as those presented in [61] and [62]. Furthermore, specific mutation operators were introduced in [63], [64]. Also, DE was combined with (adaptive) penalty functions in [65], [66], Lagrangian methods for handling equality constraints in [67], or even ensembles of constraint handling techniques [68]. More recently,  $\varepsilon$ -DE was further extended through the introduction of ranking [69] and surrogate models by using kernel regression [70].

### C. Memetic Computing approaches

Lastly, it is worth mentioning recent research on constrained optimization by Memetic Computing (MC) approaches. Memetic Computing is an emerging trend in Computational Intelligence whose focus is on algorithms composed of

multiple interacting operators, also named *memes*. Originally inspired by the diffusion of ideas in evolving populations of learning agents, where individuals undergo genetic evolution but also show learning capabilities and adaptation [71], MC nowadays embraces a broad plethora of methods consisting of multiple (possibly heterogeneous) units of “algorithmic information” whose synergistic coordination is used for problem-solving [72]. As such, the modern MC concept is not limited only to optimization problems. It also goes far beyond the initial definition of Memetic Algorithms (MA), where a population-based algorithm (e.g. an EA) is coupled with one or more individual learning units (local search) [71], as it includes several algorithmic schemes characterized by various levels of adaptation and decision-making [73]–[75], coevolution [76], local surrogate models [77], or purposely simple cascades of single-solution search units [78]. For a thorough survey of MC, we refer the interested reader to [79], [80].

So far, MC has been successfully applied mostly to unconstrained optimization, either continuous or combinatorial [80]. On the other hand, only few studies tackled constrained problems by means of memetic techniques. Presumably, this might be due to the difficulty in coordinating and balancing global exploration with local search, a crucial aspect in MC [81], when the fitness landscape is highly constrained and the search approaches the boundary of the feasible region. An interesting example of a MC method for constrained optimization is given by the agent-based memetic algorithm [82], [83], where a society of agents co-evolves by recombination only and each individual selects, independently, a local search technique from a predefined pool. Similar ideas can be found in [84], where global search (by means of DE operator), stochastic ranking and a crossover-based local search are coordinated in a memetic fashion. Other hybrid algorithms combine GA and Artificial Immune System (AIS) [85], PSO and Simulated Annealing (SA) [86], PSO and GA [87], PSO and DE [88]. In two  $(\mu+\lambda)$ -DE approaches [89], [90], multiple mutation operators are applied while the algorithms modify at runtime the policy used for ranking solutions, according to the composition of the population in terms of feasible/infeasible solutions.

More recently, some studies proposed extremely powerful memetic algorithms that make use of gradient-based information, either on the fitness or on the constraints, thus making an implicit assumption of continuous and differentiable functions. An example of such methods is given in [91], where an Estimation of Distribution Algorithm (EDA) is combined with a classic gradient-based local optimizer (DONLP2). Similarly, in [92] a memetic algorithm is proposed that combines a GA, Sequential Quadratic Programming (SQP) with second-order functional approximations, and modelling of the feasibility region through Support Vector Machine. A consensus-based variant of GA is combined with SQP also in [93], but in this case the algorithm uses gradient information on the constraints. Finally, a hybrid algorithm combining PSO, DE, CMA-ES, gradient-based mutation and constraint handling with  $\varepsilon$ -level comparison was proposed in [94].

### III. RELATED WORK: (1+1)-ViE-CMA-ES

In this section we summarize the main concepts behind covariance matrix [43] and step-size [45] adaptation in CMA-ES applied to constrained optimization. Furthermore, we summarize our previous work on the application of Viability Evolution principles to (1+1)-CMA-ES [45], resulting in the method named (1+1)-ViE.

#### A. (1+1)-CMA-ES for constrained optimization

Arnold and Hansen [43] introduced a variant of (1+1)-CMA-ES for constrained optimization. In the (1+1)-CMA-ES, an offspring solution  $\mathbf{y} \leftarrow \mathbf{x} + \sigma \mathbf{A} \mathbf{z}$  is sampled at each generation from the parent solution  $\mathbf{x} \in \mathbb{R}^n$ , where  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ ,  $\mathbf{A}$  is the Choleski decomposition of the covariance matrix  $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ , and  $\sigma \in \mathbb{R}_+$  is the global step-size. During the search, the step-size  $\sigma$  and the covariance matrix  $\mathbf{C}$  are adapted using information learned while sampling solutions.

The step-size  $\sigma$  is adapted with the 1/5<sup>th</sup> rule [44], which increases or decreases  $\sigma$  observing the probability of generating successful solutions  $P_{succ} \in [0, 1]$ , as follows:

$$\sigma \leftarrow \sigma \cdot \exp\left(\frac{P_{succ} - \frac{P_{target}}{1 - P_{target}}(1 - P_{succ})}{d}\right) \quad (2)$$

where the damping factor  $d$  controls the step-size variation, and  $P_{target}$  marks the threshold between decrease or increase in  $\sigma$ . At each iteration,  $P_{succ}$  is updated as follows:

$$P_{succ} \leftarrow (1 - c_p)P_{succ} + c_p \mathbb{1}_{f(\mathbf{y}) \leq f(\mathbf{x})} \quad (3)$$

where  $\mathbb{1}_{f(\mathbf{y}) \leq f(\mathbf{x})}$  is 1 if the offspring solution improves or equals the parent solution or 0 otherwise, and the learning rate  $c_p \in (0, 1]$  determines the fading of  $P_{succ}$ .

The covariance matrix is adapted using two separate rules. The first rule, the original rank-one update of CMA-ES, is applied whenever an offspring solution improves the parent. In this case, the variance in the direction  $\mathbf{v}$  is increased from one iteration  $g$  to the following one as in:

$$\mathbf{C}^{(g+1)} = \alpha \mathbf{C}^{(g)} + \beta \mathbf{v}^{(g)} \mathbf{v}^{(g)T} \quad (4)$$

The (1+1)-CMA-ES uses as direction  $\mathbf{v}$  the vector of fading successful steps  $\mathbf{s}$ , called evolution path, which is updated based on the value of the probability  $P_{succ}$ , and the parameters  $c$ , and  $c_{cov}^+$  defined in Table I. More specifically, if  $P_{succ} < P_{thresh}$  then  $\mathbf{s} \leftarrow (1 - c)\mathbf{s} + \sqrt{c(2 - c)}\mathbf{A} \mathbf{z}$  and  $\alpha = 1 - c_{cov}^+$ . On the contrary, if  $P_{succ} \geq P_{thresh}$  then  $\mathbf{s} \leftarrow (1 - c)\mathbf{s}$  and  $\alpha = 1 - c_{cov}^+ + c_{cov}^+ c(2 - c)$ . In both cases,  $\beta$  is set equal to  $c_{cov}^+$ . This update increases the likelihood of sampling offspring solutions in the direction of previous successful steps. Rather than updating the covariance matrix using the original rank-one update (Equation 4), the Choleski factor  $\mathbf{A}$  can be directly modified as shown in [44]:

$$\mathbf{A} \leftarrow \sqrt{\alpha} \mathbf{A} + \frac{\sqrt{\alpha}}{\|\mathbf{w}\|^2} \left( \sqrt{1 + \frac{\beta}{\alpha} \|\mathbf{w}\|^2} - 1 \right) \mathbf{s} \mathbf{w}^T \quad (5)$$

where  $\mathbf{w} = \mathbf{A}^{-1} \mathbf{s}$ . In addition, the algorithm of Arnold and Hansen [43] uses also the ‘‘active’’ covariance matrix update

rule presented in [95]. The covariance matrix is updated to decrease the variance in the direction of a particularly unsuccessful step, specifically whenever an offspring is worse than the fifth ancestor of the current parental solution, following the rule:

$$\mathbf{A} \leftarrow \sqrt{\alpha} \mathbf{A} + \frac{\sqrt{\alpha}}{\|\mathbf{z}\|^2} \left( \sqrt{1 - \frac{\beta}{\alpha} \|\mathbf{z}\|^2} - 1 \right) \mathbf{A} \mathbf{z} \mathbf{z}^T \quad (6)$$

where in this case  $\alpha = \sqrt{1 + c_{cov}^-}$  and  $\beta = c_{cov}^-$ , with  $c_{cov}^-$  defined as in Table I.

In the specific case of constrained optimization, Arnold and Hansen [43] applied a similar rule as that of Equation 6 also for decreasing the variance in the direction of constraint violations. The vector of directions of constraint violations  $\mathbf{v}_j \leftarrow (1 - c_c)\mathbf{v}_j + c_c \mathbf{A} \mathbf{z}$  is updated at every constraint  $j$  violated by step  $\mathbf{A} \mathbf{z}$ , with  $j \in 1, 2, \dots, m$ . The rule is applied when at least one constraint is violated, updating the covariance matrix according to:

$$\mathbf{A} \leftarrow \mathbf{A} - \frac{B}{\sum_{j=1}^m \mathbb{1}_{g_j(\mathbf{y}) > 0}} \sum_{j=1}^m \mathbb{1}_{g_j(\mathbf{y}) > 0} \frac{\mathbf{v}_j \mathbf{w}_j^T}{\mathbf{w}_j \mathbf{w}_j^T} \quad (7)$$

where  $\mathbf{w}_j = \mathbf{A}^{-1} \mathbf{v}_j$ .

TABLE I  
PARAMETERS OF THE (1+1)-CMA-ES WITH ACTIVE COVARIANCE MATRIX UPDATES [43]

$c = \frac{2}{n+2}$	$c_c = \frac{1}{n+2}$	$c_p = \frac{1}{12}$
$d = 1 + \frac{n}{2}$	$B = \frac{0.1}{n+2}$	$c_{cov}^+ = \frac{2}{n^2+6}$
$P_{thresh} = 0.44$	$P_{target} = \frac{2}{11}$	$c_{cov}^- = \frac{0.4}{n^{1.6}+1}$

#### B. Viability Evolution

Viability Evolution [46], [47] abstracts artificial evolutionary processes as a set of individuals, or candidate solutions, that must satisfy a number of viability criteria for surviving ever-changing environmental conditions. Viability criteria are defined as ranges of admissible values on problem objectives and constraints, the so-called viability boundaries. These boundaries, representing environmental conditions, are adapted during the evolutionary process to drive the evolving population towards desired regions of search space. At the beginning of the search the boundaries are relaxed to encompass all randomly generated initial solutions. Then, the boundaries are gradually tightened. Once viability boundaries reach the desired target, boundaries are not tightened further, and the evolutionary process is considered complete.

Although in its simplest implementation the Viability Evolution paradigm operates by eliminating non-viable individuals, the abstraction is fully compatible with classical competition-based evolutionary algorithms. Elimination by viability and competition by objective functions can be modelled at the



same time in the viability framework to determine the fitness of individuals. However, it is important to note that here fitness is intended as *a-posteriori* reproduction capability and not as an *a-priori* measurable function as currently implemented in the evolutionary computation practice.

Interestingly, some of the methods for constrained optimization, such as  $\varepsilon$ -DE [56], ASCHEA [51] and the constraint adaptation approach [96], can be loosely seen under the Viability Evolution abstraction. More specifically,  $\varepsilon$ -DE defines a tolerance  $\varepsilon$  for comparing individuals by objective or constraints. Somehow, such a threshold could be seen as a viability boundary defined on the constraint violation. However, to be fully compatible with the viability paradigm,  $\varepsilon$ -DE should discard solutions violating the  $\varepsilon$  tolerance on the constraint violations. Similarly, ASCHEA dynamically changes a tolerance on equality constraints for driving solutions towards the feasibility region determined by the equality constraint. In a similar way, the constraint adaptation approach progressively shrinks the feasible region during the evolutionary process.

### C. Introducing viability principles in CMA-ES and new rules for the adaptation of step-size: (1+1)-ViE-CMA-ES

The definition of viability boundaries, which determine admissible (viable) regions of the search space at each generation, combined with the active covariance matrix updates for constrained optimization proposed in [43], allows the search to be driven towards feasible areas of the search space by updating those boundaries. In [45], we proposed a modified version of (1+1)-CMA-ES that is based on this idea. The resulting viability evolutionary algorithm, named (1+1)-ViE, makes use of a simple rule to update the viability boundaries, relax them to encompass the initial solutions at the first generation and tighten them during evolution, as better solutions, closer to the feasible region, are generated.

Each constraint  $g_j(\mathbf{x}) \leq 0, j = 1, 2, \dots, m$  is associated to a boundary  $b_j$ , that is updated according to:

$$\mathbf{b}_j \leftarrow \max\left(0, \min\left(b_j, g_j(\mathbf{y}) + \frac{b_j - g_j(\mathbf{y})}{2}\right)\right) \quad (8)$$

whenever a successful offspring solution  $\mathbf{y}$  is generated.

Furthermore, we extended the method presented in [43] to collect independent probabilities  $p_{succ,j}$  of success for each constraint  $j$ . These probabilities allow the determination of which boundary is more likely to be violated. We then reduced or increased the global  $P_{succ}$  probability, used to adapt the step-size  $\sigma$ , following a simple heuristic: if the probability of success is lower than 50% for at least one boundary, the step-size should be reduced (as per Equation 2). The  $P_{succ}$  probability is in this case reduced as:

$$P_{succ} \leftarrow (1 - c_p)P_{succ}. \quad (9)$$

Instead, whenever a viable solution is generated, the  $P_{succ}$  and  $p_{succ,j}$  probabilities are increased as follows:

$$P_{succ} \leftarrow (1 - c_p)P_{succ} + c_p \quad (10)$$

$$p_{succ,j} \leftarrow (1 - c_p)p_{succ,j} + c_p \quad (11)$$

In [43], not adapting  $P_{succ}$  on failure can lead to the use of outdated information for the step-size adaptation, whereas

our heuristic tries to maintain and use only the most updated information, resulting in consistent performance gains on a set of unimodal constrained test functions [45].

## IV. MEMETIC VIABILITY EVOLUTION (mViE)

Although the (1+1)-ViE-CMA-ES [45] showed good performance on unimodal constrained optimization problems, its capabilities of exploring the search space are limited by the use of a single offspring solution that is sampled at each iteration. This may strongly restrict its applicability on multi-modal landscapes or when disconnected feasible areas are present.

To overcome these limitations, we introduce here a memetic computing approach, called mViE, that couples multiple local search units, based on (1+1)-ViE, with a global search operator used to recombine locally learned information. The multiple local search units can independently explore the constrained landscape, as illustrated in Fig. 1a. When local search units sample solutions that violate viability boundaries, defined on the problem constraints, the search unit learns the direction of violation of the boundary [43] and adapts covariance matrix and step-size, as shown in Fig. 1b. The adaptation of the viability boundaries, tightened during the search until they match the actual problem constraints, coupled with covariance matrix and step-size adaptation (as seen in Section III-C), drives the local search units towards feasible areas (Fig. 1c). To enable global search, we combine local search units by using operators inherited from Differential Evolution. We recombine the mean of the search distributions of the (1+1)-ViE-CMA-ES units by using rand/1 mutation and exponential crossover [97]. The search parameters of the new units (i.e., all the CMA-ES internal variables) are inherited from the closest of the search units participating in the rand/1 mutation, as illustrated in Fig. 1d. However, if the closest search unit has converged, the parameters of the offspring unit are reinitialized to their default values. Crucially for the performance of the method, an adaptive scheduler adjusts the allocation of function evaluations to local search units or to global search operators (Fig. 1e).

The flowchart of our method, implementing the operations described in Fig. 1, is presented in Fig. 2. At the beginning of execution, mViE samples the desired number  $pop_{size}$  of local search units and initializes internal variables that are used by the scheduler during the execution. At each iteration, the scheduler (Fig. 2, block with thick green border, gray in print) allocates a function evaluation either to a local search unit (Fig. 2, left branch) or to the global recombination operator (Fig. 2, right branch).

If a local search step is executed, the most promising unit in the population is selected, by ranking the local search units by fitness and constraints violation and selecting the first active (non-converged) top-ranking search unit. The (1+1)-ViE local search unit is then allowed to perform a single function evaluation. Finally, the moving average  $P_{succ,local}$ , i.e. the success probability of the local search units at improving the global best solution, is updated. To compare new candidate solutions with the current global best solution found, we employ the three feasibility rules presented in [26]. Each local

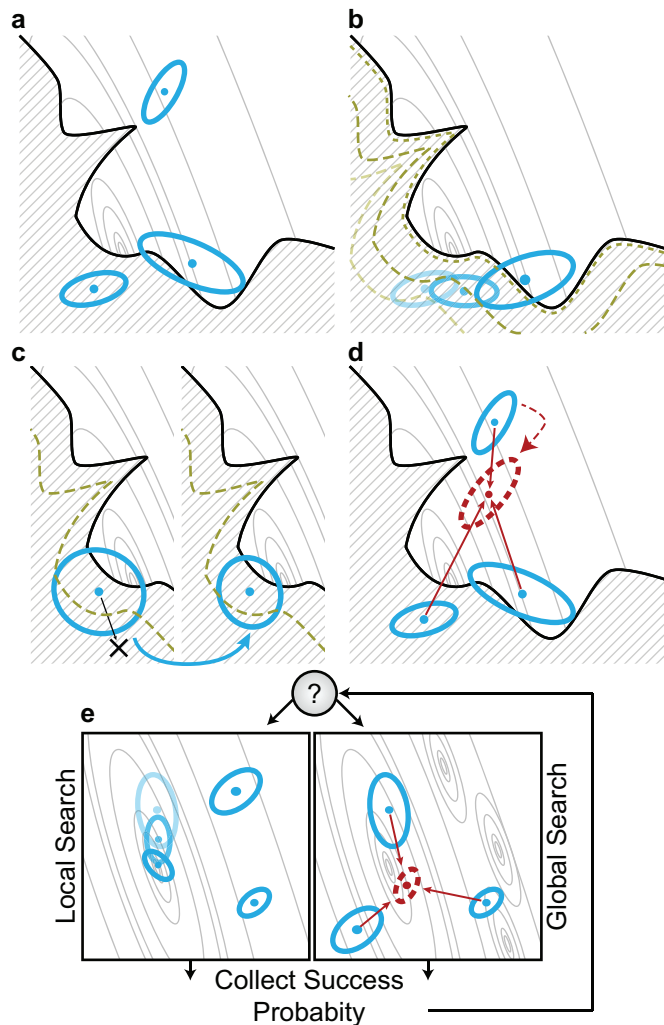


Fig. 1. Graphical representation of the main features of mViE on a simplified two-dimensional search landscape. Non-linear constraints on the search domain are represented as solid black lines, which outline the boundaries of the infeasible area marked with oblique gray hatching. Objective function contour lines are represented in the feasible areas as thin gray lines. Panels **a**, **b** and **c** represents the local search component of mViE, while panel **d** the global search one. **a**) Multiple search units (solid cyan ellipsoids, gray in print) sample solutions from different areas of the search landscape. A single search distribution is represented as an ellipsoid with its mean shown as a dot. **b**) Constraints can be described as viability boundaries (dashed gold lines, gray in print) and can be relaxed. By progressively tightening the boundaries, while adapting the search distribution to the viability boundary violations, it is possible to drive the search units within the feasible areas. **c**) When a search distribution samples a solution that violates the viability boundaries (left), the most-likely direction of boundary violation is learned and used to adapt the search distribution to reduce the likelihood of searching in that direction (right). **d**) Global search component. A new local search unit (dashed red ellipsoid, dark gray in print) is created by applying Differential Evolution mutation and crossover operators (solid arrows) on multiple search distributions. The parameters of the new distribution are inherited from the closest of the recombinant search units (dashed arrow). **e**) The probability of improving solutions generated by local or global search operators are collected and used to adapt the activation frequencies of global or local search, either stepping a local search unit (left) or recombining information using a global search operator (right).

search unit maintains its viability boundaries and adapts them every time a better solution is sampled. The boundary on the objective is updated only when operating in the feasible

area. Additional information on the probability of generating solutions that satisfy each constraint is maintained by each local search unit and used for updating the step-size, as described in Section III-C. After every execution of a local search unit, the unit is checked for convergence and disabled if one of the convergence criteria is met (see Section IV-B).

On the other hand, when the global search step is executed, the search unit to be replaced is selected by picking two search units at random and choosing the worse one. The selected local search unit is then replaced if a better one is generated by means of Differential Evolution operators (applied to three randomly chosen search units).

After the local or global search steps are executed, the method collects statistics on the performance of the executed branch (Fig. 2, variable updates marked by dashed boxes). More specifically, it collects the number of function evaluations  $N_{evals,\{local,global\}}$  and the number of improvements to the global best solution  $N_{succ,\{local,global\}}$  of each search step. To put into effect the different meanings of “success” for the local and global search steps, i.e. to weigh the relative “importance” of an improvement obtained by the two steps, the moving averaged probabilities of success of local and global search components are regulated by two additional parameters,  $c_\alpha$  and  $c_\beta$ . In particular, when a solution better than the global best is found by a local search unit, the local step’s probability of success is increased according to:

$$P_{succ,local} \leftarrow (1 - c_\alpha) \cdot P_{succ,local} + c_\alpha \cdot \quad (12)$$

On the contrary, when a local search unit samples a solution that does not improve the global best, the probability is decreased according to:

$$P_{succ,local} \leftarrow (1 - c_\alpha) \cdot P_{succ,local} \cdot \quad (13)$$

However, if the solution violates boundaries defined on some constraints, to account for the function evaluations needed for adapting to the current boundaries, we discount the original  $c_\alpha$  coefficient by a factor  $\beta_R$ , replacing it with a reduced coefficient  $c_\beta = \beta_R \cdot c_\alpha$ .

As for the global search step, the probability  $P_{succ,global}$  is increased in two conditions. When a solution better than the global best solution is found, the probability is modified with a rule similar to Equation 12:

$$P_{succ,global} \leftarrow (1 - c_\alpha) \cdot P_{succ,global} + c_\alpha \cdot \quad (14)$$

The probability is also increased when a local search unit better than the parent unit is discovered, although this second increase is lower (we use the same coefficient  $c_\beta$ ). In the other cases, the probability is decreased similar to Equation 13:

$$P_{succ,global} \leftarrow (1 - c_\alpha) \cdot P_{succ,global} \cdot \quad (15)$$

#### A. Scheduler for selection of local/global search operators

The efficient allocation of function evaluations to local (Fig. 1a-c; Fig. 2, left branch) or global (Fig. 1d; Fig. 2, right branch) search is ensured by the scheduler presented in Algorithm 1 (Fig. 1e; Fig. 2, block with thick green border, gray in print). The scheduler uses information collected during

**mViE parameters and internal variables used by scheduler**

$pop_{size}$	Number of local search units
$c_\alpha$	Fading parameter of local/global probability of success
$c_\beta = \beta_R c_\alpha$	Fading parameter of local/global probability of success, controlled by the user using $\beta_R$
$L$	Controls global/local relative frequency of execution
$P_{succ,local}$	Probability of success of local units
$P_{succ,global}$	Probability of success of global operator
$C_{active,local}$	Number of active local search units
$N_{evals,local}$	Number of FES allocated to local search units
$N_{evals,global}$	Number of FES allocated to global search operator
$N_{succ,local}$	Number of FES when local search unit was successful
$N_{succ,global}$	Number of FES when global search operator was successful

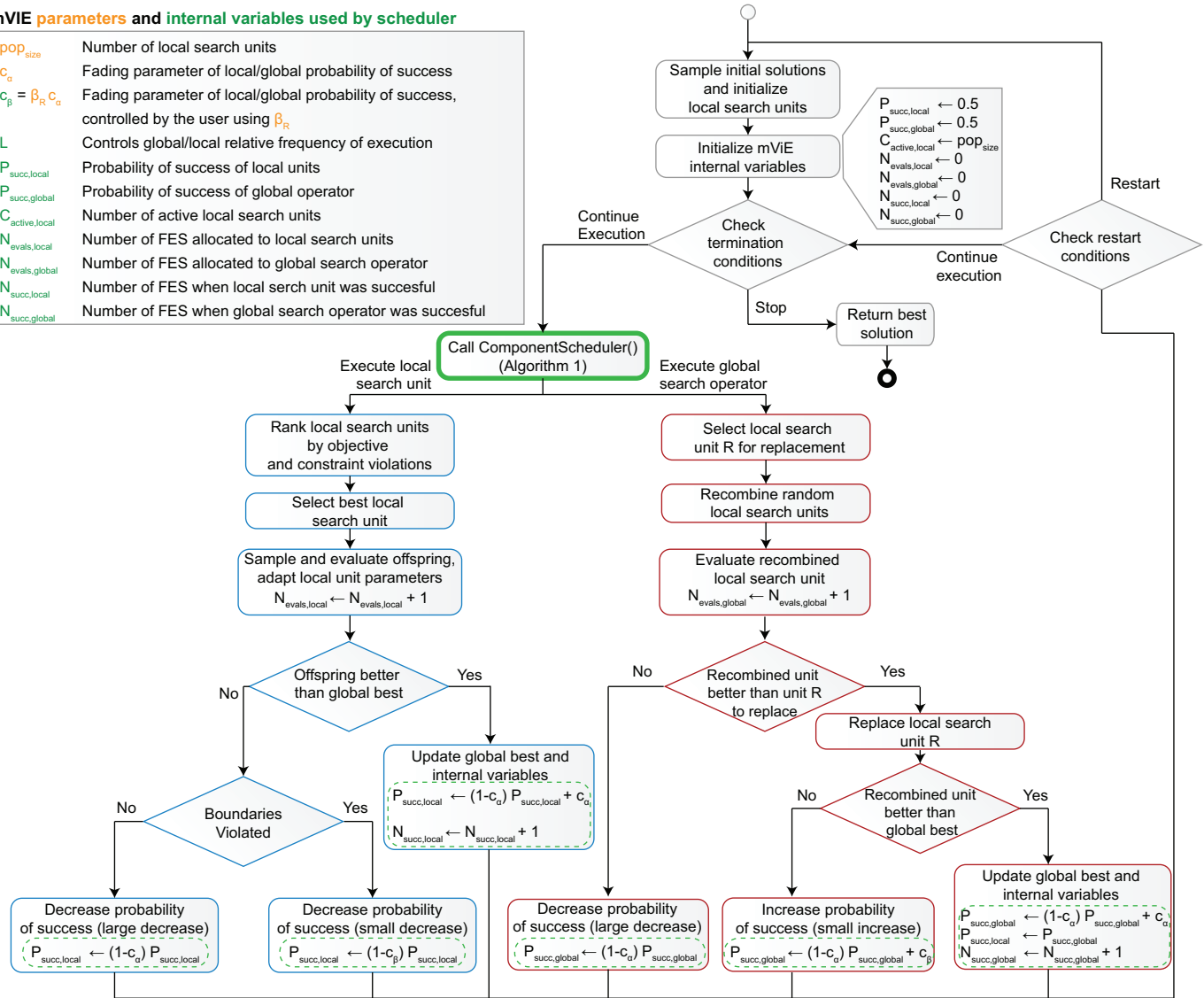


Fig. 2. Flowchart of mViE. The algorithm's main operation can be divided in three main functional blocks. The scheduler (block with thick green border, gray in print) allocates a function evaluation either to a local search unit (left branch) or to the global search operator (right branch). Statistics used by the scheduler are collected after the execution of the two branches (dashed boxes).

the search process, i.e. the moving average probability of success  $P_{succ,\{local,global\}}$ , the total number of global best improvements  $N_{succ,\{local,global\}}$  and the total number of function evaluations  $N_{evals,\{local,global\}}$  allocated to the local and global search components.

During the first  $100 \times n$  function evaluations (empirically set during preliminary experiments) both components are always used to ensure an initial learning phase and avoid unbalancing the search towards local or global search because of initial evaluations. Then, the total probability of success  $P_{\{local,global\}} = \frac{N_{succ,\{local,global\}}}{N_{evals,\{local,global\}}}$ , which considers all the previous history of the evolutionary run, is computed. We aggregate this information about total probability of success with the information on the current probability of success ( $P_{succ,\{local,global\}}$ ) by multiplication. To prevent the frequency of execution of one of the two components from falling to zero, therefore disabling the component until the end of the

search process, we limit the minimum frequency of execution of one component to a relative fraction  $L$  of the other component's frequency. Thus, the frequency  $f$  of execution of the two components is always limited  $\frac{L}{1+L} \geq f \geq \frac{1}{1+L}, 0 \leq L \leq 1$ . In the case where all the local search units have converged, i.e. the number of active local search units  $C_{active,local}$  is zero, the local search step is disabled and only global search is performed.

**B. Termination conditions**

Our method uses both local convergence conditions to disable local search units, and global convergence conditions to restart the algorithm. We disable a local search unit according to the standard stopping criteria for (1+1)-ViE, namely when:

- 1) the evolution path  $s$  multiplied by the step-size  $\sigma$  is smaller than  $10^{-12}$ ;

**Algorithm 1** Selection of local/global search operator

---

```

function COMPONENTSCHEDULER( $N_{evals,local}$ ,
 $N_{evals,global}$ ,  $N_{succ,local}$ ,  $N_{succ,global}$ ,  $L$ ,  $C_{active,local}$ )

if ( $N_{evals,local} + N_{evals,global}$ ) <  $100 \cdot n$  then
    return ▷ Set for execution local and global search
end if
if  $N_{evals,local} = 0$  then
     $P_{local} \leftarrow 0$ 
else
     $P_{local} \leftarrow P_{curr,local} \cdot \frac{N_{succ,local}}{N_{evals,local}}$ 
end if
if  $N_{evals,global} = 0$  then
     $P_{global} \leftarrow 0$ 
else
     $P_{global} \leftarrow P_{curr,global} \cdot \frac{N_{succ,global}}{N_{evals,global}}$ 
end if
 $P_1 \leftarrow \max(P_{local}, L \cdot P_{global})$ 
 $P_2 \leftarrow \max(P_{global}, L \cdot P_{local})$ 

if  $\text{RAND}() < \frac{P_1}{P_1 + P_2} \wedge C_{active,local} > 0$  then
    return ▷ Set for execution local search
else
    return ▷ Set for execution global search
end if

end function

```

---

- 2) the maximum diagonal element on the covariance matrix multiplied by the step-size  $\sigma$  is larger than  $10^8$ ;
- 3) the condition number of the covariance matrix is larger than  $10^{14}$ .

After a local search unit has been disabled, it can only be replaced with another active local search unit by the global search operator. The algorithm is restarted when all the local search units have converged to the same solution. This check is performed by measuring the difference between the mean of the objective and constraint violations of the local search units and the best constraint violation objective of the best solution.

## V. EXPERIMENTAL SETUP

To assess the performance of the proposed method, we selected all the optimization problems with only inequality constraints from the CEC 2006 benchmark [34]<sup>3</sup>. A summary of the tested benchmark functions, with their relative features (problem dimension, kinds of constraints, number of constraints active at the optimum) is reported in Table II<sup>4</sup>

<sup>3</sup>Although a newer CEC benchmark is available [35], with some interesting features such as scalable functions, the CEC 2006 benchmark still represents the most popular tool for testing new constrained optimization algorithms, given the availability of results for several methods from the literature.

<sup>4</sup>We should note that our algorithm is specifically designed for handling optimization problems with inequality constraints only. A more detailed analysis of the behaviour of mViE on problems including (also) equality constraints, together with the original MATLAB<sup>®</sup> source code of all our experiments, is available at <http://lis.epfl.ch/mViE>.

As specified in [34], we executed our algorithm for 500000 function evaluations on each test function. For each function, we measured the success rate (SR) over 25 runs of mViE the number of function evaluations (NFES) needed for solving the problem at the desired accuracy (set to  $10^{-4}$ ). Furthermore, we tested the method on a set of four engineering problems reported in Fig. 3. The engineering problems were run for a maximum of 200000 function evaluations.

In the experiments, we set the algorithm's main parameters after performing a preliminary parameter analysis, reported in Appendix A. The identified parameter values, used in the rest of this paper, are  $c_\alpha = 0.1$ ,  $\beta_R = 0.05$ ,  $L = 0.18$ ,  $pop_{size} = 40$ . For the DE operators, we used the standard values reported in the literature [97], namely  $F = 0.5$  and  $CR = 0.9$ . The parameters of the local search component, based on (1+1)-CMA-ES, are as set in [43], reported in Table I. All the experiments were performed on Intel<sup>®</sup> machines with Core<sup>™</sup> i7-2600 CPU @ 3.40GHz and 8GB of RAM.

TABLE II  
PROBLEMS WITH INEQUALITY CONSTRAINTS FROM THE CEC 2006 COMPETITION ON CONSTRAINED OPTIMIZATION. WE SHOW THE NUMBER OF DIMENSIONS ( $n$ ), LINEAR (LI), NON-LINEAR (NI) AND ACTIVE CONSTRAINTS AT THE OPTIMUM ( $a$ ).

Problem	n	LI	NI	a
g01	13	9	0	6
g02	20	0	2	1
g04	5	0	6	2
g06	2	0	2	2
g07	10	3	5	6
g08	2	0	2	0
g09	7	0	4	2
g10	8	3	3	6
g12	3	0	1	0
g16	5	4	34	4
g18	9	0	13	6
g19	15	0	5	0
g24	2	0	2	2

## VI. RESULTS

We first review the results on CEC 2006 problems, then we move forward to analyze the results on the engineering problems. Moreover, we show a more detailed analysis of few selected runs of mViE, to exemplify its behaviour on different fitness landscapes. Finally, we dissect the performance of mViE by assessing the contribution of each of its components.

### A. CEC 2006 problems

On all the 25 runs of every CEC 2006 test problem with only inequality constraints, our method exhibited 100% success rate (SR), i.e. it could reach in every run the target fitness difference from the optimum ( $10^{-4}$ ). The number of function evaluations (NFES) needed for reaching success is reported in Table III. Furthermore, in Appendix B we report the statistics on the error values, according to the CEC 2006 format [34].

We compared mViE's SR and median NFES needed to solve the CEC 2006 problems against representative algorithms from the state-of-the-art in constrained optimization that do



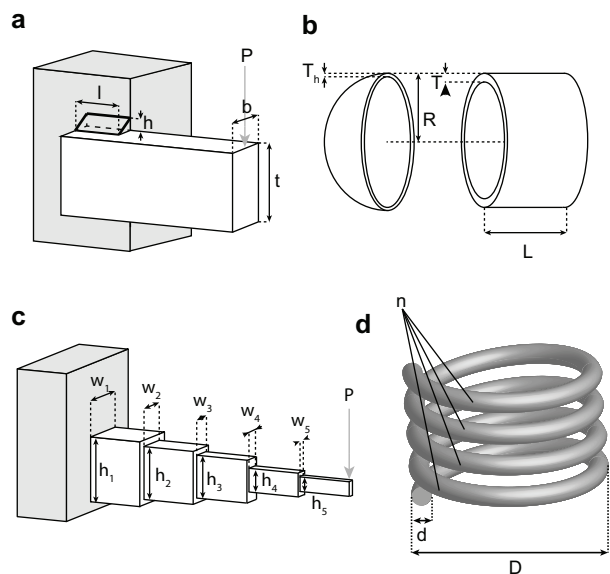


Fig. 3. Engineering problem benchmarks [3], [50]. **a)** Welded beam design optimization to minimize fabrication cost. The beam is fabricated out of carbon steel and welded on a rigid supporting structure. The shear force  $P$  is loading the free tip of the beam. The dimensions of the beam (width  $t$  and thickness  $b$ ) and the width  $h$  and length  $l$  of the welded joint have to be optimized subject to constraints on shear stress, bending stress, buckling load on the bar, deflection and geometric constraints. **b)** Minimization of fabrication cost of a pressure vessel. The thickness of the spherical head  $T_h$ , the thickness of the spherical skin  $T_s$ , and the inner radius of the vessel have to be designed to comply with constraints derived from the ASME (American Society of Mechanical Engineers) standards on pressure vessels. **c)** Minimization of volume of a stepped cantilever. The cantilever is composed of five segments having variable cross-section, defined by the design variables  $w_i$  and  $h_i$ . The design is subject to constraints limiting the bending stress and aspect ratio of each beam segment, and the total deflection of the cantilever at the tip. **d)** Design of a tension compression spring. The weight of the spring must be minimized optimizing mean coil diameter  $D$ , wire diameter  $d$  and number of active coils  $n$ , subject to constraints on deflection, shear stress, surge frequency, and maximum size on the outside diameter.

TABLE III

BEST, MEDIAN, WORST, MEAN AND STD. DEV. OF NFES TO ACHIEVE THE FIXED ACCURACY LEVEL ( $(f(\mathbf{x}) - f(\mathbf{x}^*)) \leq 0.0001$ ), AND SUCCESS RATE OVER 25 RUNS OF mViE ON THE SELECTED CEC 2006 PROBLEMS.

Prob.	Best	Median	Worst	Mean	Std	SR
g01	15032	20304	26301	20645.8	2757.39	100%
g02	42462	61072	200394	67972.4	30770.3	100%
g04	3089	3945	12461	4540.88	2154.32	100%
g06	1072	1901	7222	2782.84	1884.95	100%
g07	5954	7281	44035	10511.9	8732.84	100%
g08	185	482	812	504.2	183.513	100%
g09	2586	3436	26837	5193.68	5064.41	100%
g10	10995	14734	99587	23884.3	22860.9	100%
g12	187	3809	12607	3967.32	2371.44	100%
g16	2415	3128	10214	3843.12	2100.47	100%
g18	4683	7272	73350	13916.9	17785.5	100%
g19	22658	25914	35753	26770.1	3029.09	100%
g24	492	718	2511	838.68	400.114	100%

not make use of traditional non-linear programming (NLP) techniques, see Table IV (top). For convenience, we grouped the algorithms under comparison based on their underlying meta-heuristic: DE, CMA-ES, PSO, and algorithms which

are based on other evolutionary paradigms (GA, ES, and hybrid variants of ES/DE). The latter group is labeled as “others”. DE-based and CMA-ES-based algorithms have been considered for comparison as DE and CMA-ES represent the building blocks of the proposed mViE. Therefore, it is interesting to compare our method with other algorithms using similar logics. The two other groups (PSO-based methods and “others”) were included to show how our method compares with other successful examples from the state-of-the-art that are inspired by different computational paradigms. Regardless of the underlying meta-heuristic, it is worth noting how each of the compared algorithms ranks feasible/infeasible solutions: with respect to this aspect, we can see from Table IV that the majority of the algorithms in the state-of-the-art use  $\epsilon$ -ranking, three feasibility rules or (adaptive) penalty functions. Few methods use instead algorithm-specific methods, e.g. based on surrogate models or learning of the feasibility structure.

Table V shows the comparison with all the aforementioned algorithms. As an additional comparison, the table also includes the aggregate best results obtained on each problem by *all* the algorithms presented at CEC 2006. For the reader’s convenience, we report again (separately) the NFES and SR of mViE. For each problem, we also report the relative difference of NFES (labeled as  $\Delta\text{NFES}\%$ ) between mViE and the best algorithm (underlined) from each group of algorithms, i.e.:

$$\Delta\text{NFES}\% = \frac{\text{NFES}_{\text{mViE}} - \text{NFES}_{\text{bestGroupAlg}}}{\text{NFES}_{\text{bestGroupAlg}}} \times 100.$$

We calculate in a similar way the  $\Delta\text{NFES}\%$  w.r.t. the best results from CEC 2006. In the table, the symbol “-” indicates that the result of that algorithm for that problem is not available.

A closer examination at the comparison between mViE and the group of algorithms derived from DE reveals a strong performance advantage of mViE in finding optimal solutions. In only one problem out of the thirteen considered,  $g_{12}$ , mViE needs about 3.17 times more function evaluations to reach the optimum. In all the other problems, mViE needs between 14.1% ( $g_{02}$ ) and 87.24% ( $g_{07}$ ) less evaluations than the best DE-based algorithm (but, it should also be noted that on  $g_{02}$  the SR of the best algorithm, namely MDE [57], is 16%, while for mViE it is 100%).

Comparing mViE with the algorithms derived from CMA-ES highlights an even stronger performance gain. mViE is slower (in terms of NFES needed to reach the optimum) only in three problems  $g_{06}$ ,  $g_{08}$  and  $g_{10}$ . It must be noted however that for  $g_{06}$  and  $g_{10}$  (both unimodal problems) the fastest algorithm is (1+1)-ViE, i.e. the same employed by our local search units, presented in our previous work [45]. It is therefore to be expected a somehow higher number of function evaluations, as in the method proposed here the global search component is also at work and may slow down the search for the optimum when the landscape is unimodal. On  $g_{08}$ , mViE is also 1.38 times slower than APM-CMA-ES [39], but given the already low NFES needed to solve this problem the performance decay in this case is not particularly relevant. In all other cases, mViE needs less evaluations (up to 95.4% less for  $g_{02}$ ) than the best algorithm in the group.

Notably, mViE outperforms, on all problems, also all the

TABLE IV

MViE WAS COMPARED AGAINST MULTIPLE ALGORITHMS SELECTED FROM THE LITERATURE: ALGORITHMS NOT USING NLP TECHNIQUES (TOP); MEMETIC ALGORITHMS USING NLP TECHNIQUES (BOTTOM). FOR EACH ALGORITHM WE SUMMARIZE THE MAIN FEATURES, TOGETHER WITH THE METHOD USED BY THE ALGORITHM TO COMPARE SOLUTIONS AND HANDLE CONSTRAINTS.

	Algorithm	Ranking method	Notes
DE	Eps-DE [56]	$\varepsilon$ -ranking	Gradient-based mutations.
	Eps-RDE [69]	$\varepsilon$ -ranking	Surrogate function is used in $\varepsilon$ -ranking comparisons.
	MPDE [98]	Penalty Function (Near Feasibility Threshold [99])	Uses multiple sub-populations to maintain diversity.
	GDE [100]	Non dominated sorting	Problem reformulated as multi-objective problem using sum of constraint violations as objective.
	MDE [57]	Three feasibility rules <sup>A</sup>	Modified DE mutation operator that considers the best and three other randomly selected individuals. Uses a diversity procedure based on stochastic ranking with probability modified during search.
	JDE-2 [101]	Three feasibility rules <sup>B</sup>	Adapts the F and CR parameters of DE during search.
CMA-ES	(1+1)-aCMA [43]	If offspring violates constraints, adapts covariance matrix, otherwise substitutes parent if better fitness.	
	(1+1)-ViE [45]	As in (1+1)-aCMA	Includes a mechanism for adapting the step-size based on information gathered on constraint violations. Relaxes constraints and uses them to drive the search towards feasible area.
	APM-CMA-ES [39]	Adaptive penalty function	Adapts tolerances on equality constraints during the search.
	AP-CMA-ES [41]	Adaptive penalty function	Aggregates the violation of all solutions in the current population.
PSO	PSO [102]	Three feasibility rules <sup>A</sup>	
	COPSO [7]	Three feasibility rules <sup>A</sup>	Adapts tolerances on equality constraints during the search. Maintains an archive of solutions that are estimated to be close to constraint boundary. Applies local perturbation of best solutions found.
	PES0+ [103]	Three feasibility rules <sup>A</sup>	Adapts tolerances on equality constraints during the search. Maintains an archive of solutions close to constraint boundary. Applies local perturbation of best solutions found.
Others	ASRES [104]	Stochastic ranking performed using surrogate models of constraints and objectives.	ES
	$(\mu+\lambda)$ -CDE [90]	Adaptive trade-off model to select offspring for next generation that depends on population composition (only feasible, only infeasible or mixed).	$(\mu+\lambda)$ -DE. Selects adaptively among various DE mutation strategies.
	ICDE [89]	As in $(\mu+\lambda)$ -CDE	$(\mu+\lambda)$ -DE. Uses an archiving strategy in the infeasible population case.
	PCX [105]	Three feasibility rules <sup>C</sup>	Steady-state GA with PCX recombination operator. Adapts tolerances on equality constraints during the search.
<sup>A</sup> the error for infeasible solutions is computed as the sum of constraint violations <sup>B</sup> the error for infeasible solutions is computed as the mean of the constraint violations <sup>C</sup> the error for infeasible solutions is computed as the sum of constraint violations, the fitness is re-scaled according to special rules			

	Algorithm	Ranking method	Notes
MA with NLP Techniques	SADE [58]	Three feasibility rules <sup>A</sup>	DE + SQP. Selects probabilistically among various DE mutation strategies.
	DMS-PSO [106]	Uses one constraint (adaptively chosen) for each sub-population as objective.	PSO + SQP. Maintains multiple sub-populations.
	GB-MA [91]	Over-penalize approach [107]: ranks first feasible solutions by objective, then infeasible ones by sum of constraint violations.	EDA + DONLP2
	FSM [92]	Neighborhood-modulated selection	GA + SQP. Maintains a database of evaluated solutions. Uses Support Vector Machine to model the structure of the feasible region and choose which solutions are significant for local refinements.
<sup>A</sup> the error for infeasible solutions is computed as the mean of constraint violations, normalized by the maximum violation observed for each constraint			

TABLE V

MEDIAN NFES TO ACHIEVE THE FIXED ACCURACY LEVEL ( $(f(\mathbf{x}) - f(\mathbf{x}^*)) \leq 0.0001$ ) AND SUCCESS RATE FOR THE SELECTED CEC 2006 PROBLEMS. ALL THE COMPARED RESULTS ARE OBTAINED FROM THE CORRESPONDING PAPERS. PLEASE NOTE THAT RESULTS REPORTED FOR EPS-RDE [69] AND PCX [105] ARE, RESPECTIVELY, MEAN AND 25<sup>th</sup> PERCENTILE, RATHER THAN MEDIAN NFES. BEST CEC 2006 AGGREGATES THE BEST RESULTS OBTAINED ON EACH PROBLEM BY ALL THE ALGORITHMS PRESENTED AT CEC 2006.  $\Delta$ NFES% INDICATES FOR EACH PROBLEM THE RELATIVE DIFFERENCE OF NFES BETWEEN THE PROPOSED ALGORITHM mViE AND THE BEST ALGORITHM IN EACH GROUP (UNDERLINED).

This work			DE												
Prob.	mViE		Eps-DE [56]		Eps-RDE [69]		MPDE [98]		GDE [100]		MDE [57]		jDE-2 [101]		$\Delta$ NFES%
	NFES	SR	NFES	SR	NFES	SR	NFES	SR	NFES	SR	NFES	SR	NFES	SR	
g01	20304	100%	59345	100%	56508	100%	43794	100%	<u>40200</u>	100%	75000	100%	50354	100%	-49.49%
g02	61072	100%	146911	100%	99742	100%	280272	100%	106332	72%	<u>71100</u>	16%	138102	92%	-14.10%
g04	3945	100%	26098	100%	51614	100%	20823	100%	<u>15157</u>	100%	39300	100%	40958	100%	-73.97%
g06	1901	100%	7316	100%	10152	100%	10550	100%	6431	100%	5250	100%	29844	100%	-63.79%
g07	7281	100%	74476	100%	99830	100%	<u>57079</u>	100%	112969	100%	176400	100%	126637	100%	-87.24%
g08	482	100%	1182	100%	4063	100%	1632	100%	1486	100%	<u>900</u>	100%	3564	100%	-46.44%
g09	3436	100%	23172	100%	42266	100%	20814	100%	30784	100%	<u>15000</u>	100%	55515	100%	-77.09%
g10	14734	100%	105799	100%	99820	100%	<u>48508</u>	100%	81827	100%	163500	100%	144247	100%	-69.62%
g12	3809	100%	4155	100%	7873	100%	4227	100%	3016	100%	<u>1200</u>	100%	6684	100%	+217.41%
g16	3128	100%	13001	100%	-	-	13135	100%	13307	100%	<u>8700</u>	100%	261549	100%	-64.04%
g18	7272	100%	59232	100%	-	-	42550	100%	377732	76%	118050	100%	449306	100%	-82.90%
g19	25914	100%	354060	100%	-	-	115054	100%	206556	88%	-	-	<u>101076</u>	100%	-74.36%
g24	718	100%	2928	100%	-	-	4371	100%	3059	100%	<u>1650</u>	100%	319611	100%	-56.48%

CMA-ES									
Prob.	(1+1)-aCMA [43]		(1+1)-ViE [45]		APM-CMA-ES [39]		AP-CMA-ES [41]		$\Delta$ NFES%
	NFES	SR	NFES	SR	NFES	SR	NFES	SR	
g01	-	-	-	-	<u>51400</u>	100%	184778	52%	-60.50%
g02	-	-	-	-	<u>1328100</u>	30%	-	-	-95.40%
g04	-	-	-	-	25700	100%	4896	100%	-19.42%
g06	1060	100%	<u>900</u>	100%	7300	100%	2424	100%	+111.22%
g07	11283	100%	<u>7545</u>	100%	116800	100%	14420	100%	-3.50%
g08	-	-	-	-	1500	100%	<u>348</u>	100%	+38.50%
g09	4106	100%	<u>3660</u>	100%	77400	100%	5346	100%	-6.12%
g10	18781	100%	<u>8295</u>	100%	407400	100%	23780	100%	+77.62%
g12	-	-	-	-	<u>7500</u>	100%	26278	100%	-49.21%
g16	-	-	-	-	-	-	<u>5648</u>	100%	-44.62%
g18	-	-	-	-	-	-	<u>57430</u>	100%	-87.34%
g19	-	-	-	-	-	-	<u>74472</u>	100%	-65.20%
g24	-	-	-	-	-	-	<u>996</u>	100%	-27.91%

PSO									
Prob.	PSO [102]		COPSO [7]		PES0+ [103]		$\Delta$ NFES%		
	NFES	SR	NFES	SR	NFES	SR			
g01	46405	52%	95000	30%	102100	100%	-56.24%		
g02	-	-	<u>175800</u>	22%	219400	56%	-65.26%		
g04	19681	100%	65100	30%	79300	100%	-79.95%		
g06	<u>20007</u>	100%	54200	30%	56800	100%	-90.50%		
g07	327283	8%	<u>227600</u>	30%	358600	96%	-96.80%		
g08	<u>2311</u>	100%	6850	30%	6100	100%	-79.14%		
g09	57690	100%	78500	30%	96400	100%	-94.04%		
g10	461422	32%	<u>221300</u>	30%	468350	16%	-93.34%		
g12	-	-	<u>3933</u>	100%	6900	30%	8100	100%	-3.15%
g16	<u>33021</u>	100%	41000	30%	48700	100%	-90.52%		
g18	177989	80%	<u>153600</u>	27%	211800	92%	-95.26%		
g19	365284	8%	<u>259650</u>	14%	-	-	-90.02%		
g24	<u>7487</u>	100%	19350	30%	19900	100%	-90.41%		

Others									
Prob.	ASRES [104]		$(\mu+\lambda)$ -CDE [90]		ICDE [89]		PCX [105]		$\Delta$ NFES%
	NFES	SR	NFES	SR	NFES	SR	NFES	SR	
g01	62800	100%	89320	100%	106540	100%	<u>62026</u>	100%	-67.26%
g02	321200	12%	<u>272860</u>	100%	281470	100%	500000	64%	-77.62%
g04	57600	100%	<u>30130</u>	100%	36820	100%	40140	100%	-86.90%
g06	48400	100%	<u>11200</u>	100%	12880	100%	36180	100%	-83.02%
g07	<u>135600</u>	100%	139720	100%	135730	100%	258840	100%	-94.63%
g08	4800	100%	2170	100%	<u>1960</u>	100%	3510	100%	-75.41%
g09	72000	100%	39550	100%	<u>37870</u>	100%	58700	100%	-90.92%
g10	276000	100%	188860	100%	325570	100%	<u>109970</u>	100%	-86.60%
g12	15600	100%	<u>5110</u>	100%	6580	100%	11940	100%	-25.46%
g16	39200	100%	<u>18970</u>	100%	25060	100%	36790	100%	-83.51%
g18	119600	96%	218050	100%	134680	100%	<u>96180</u>	100%	-92.44%
g19	212000	92%	265930	100%	297640	100%	<u>187734</u>	100%	-86.19%
g24	14000	100%	<u>5110</u>	100%	5740	100%	13690	100%	-85.95%

CEC 2006			
Prob.	Best		$\Delta$ NFES%
	NFES	SR	
g01	25115	100%	-19.15%
g02	96222	100%	-36.53%
g04	15281	100%	-74.18%
g06	5202	100%	-63.45%
g07	26578	100%	-72.60%
g08	918	100%	-47.49%
g09	16152	100%	-78.72%
g10	25520	100%	-42.26%
g12	1308	100%	+191.20%
g16	8730	100%	-64.17%
g18	28261	100%	-74.27%
g19	21830	100%	+18.71%
g24	1794	100%	-59.98%

PSO-based methods, as well as other algorithms not classifiable in any of the aforementioned groups (PCX [105], ASRES [104], and two DE-based memetic algorithms using an adaptive policy to rank solutions depending on the population composition, namely  $(\mu+\lambda)$ -DE [90] and ICDE [89]). In most cases, mViE needs around 10-20% (or less) NFES w.r.t. the

best algorithm in the two groups. This result is confirmed when mViE is compared against the best aggregate results from CEC 2006: except g12 and g19, mViE is always able to solve the problem in less NFES.

Finally, to broaden our comparative analysis we considered some modern memetic algorithms that *do* make use of tra-

TABLE VI

MEDIAN NFES TO ACHIEVE THE FIXED ACCURACY LEVEL  $((f(\mathbf{x}) - f(\mathbf{x}^*)) \leq 0.0001)$  AND SUCCESS RATE FOR THE SELECTED CEC 2006 PROBLEMS (CONTINUED FROM TABLE V). ALL THE COMPARED RESULTS ARE OBTAINED FROM THE CORRESPONDING PAPERS. RESULTS REPORTED FOR GB-MA [91] SHOW MEAN, RATHER THAN MEDIAN NFES.  $\Delta$ NFES% INDICATES FOR EACH PROBLEM THE RELATIVE DIFFERENCE OF NFES BETWEEN mViE AND THE BEST ALGORITHM (UNDERLINED).

Prob.	MA with NLP Techniques								$\Delta$ NFES%
	SADE [58]		DMS-PSO [106]		GB-MA [91]		FSM [92]		
	NFES	SR	NFES	SR	NFES	SR	NFES	SR	
g01	25115	100%	25816	100%	7859	100%	<u>294</u>	100%	+6806.12%
g02	128970	84%	87107	84%	<u>45555</u>	100%	91324	100%	+34.06%
g04	25107	100%	25443	100%	1201	100%	<u>269</u>	100%	+1366.54%
g06	14404	100%	27636	100%	489	100%	<u>110</u>	100%	+1628.18%
g07	101240	100%	26685	100%	3588	100%	<u>2225</u>	100%	+227.24%
g08	1272	100%	3892	100%	1068	100%	448	100%	+7.59%
g09	16787	100%	29410	100%	<u>1632</u>	100%	6586	100%	+110.54%
g10	52000	100%	25500	100%	17319	100%	<u>1712</u>	100%	+760.63%
g12	1717	100%	6826	100%	348	100%	<u>176</u>	100%	+2064.20%
g16	14433	100%	28433	100%	7092	100%	<u>325</u>	100%	+862.461%
g18	26000	92%	28000	100%	11095	100%	<u>1875</u>	100%	+287.84%
g19	51588	100%	21587	100%	13355	100%	<u>1370</u>	100%	+1791.53%
g24	4843	100%	18729	100%	425	100%	<u>380</u>	100%	+88.94%

ditional NLP techniques, see Table IV (bottom) for details. These are methods that combine a population-based algorithm (DE, PSO, EDA or GA) with programming techniques, e.g. SQP. As such, these algorithms i) assume that the objective and constraint functions are differentiable and ii) compute and use first (and second) order information on constraints or objectives. Although evolutionary algorithms (such as our proposed mViE) cannot be directly compared with these techniques, as they assume that no gradient information is available and are typically derivative-free, we deemed interesting to relate our method also with this part of the literature. Results of these comparisons can be found in Table VI. As expected, the methods that make use of gradient information have a clear advantage in terms of NFES needed to converge to the optimum: this is particularly evident when mViE is compared against FSM [92]: the comparison shows that FSM converges in a very limited NFES, as it benefits from the use of second-order information in the SQP included in its memetic structure. On the other hand, it is remarkable that mViE, despite the lack of any information about the gradient, outperforms SADE [58] and DMS-PSO [106] (which both include SQP) on all problems but two (g12 and g19, respectively) and can exceed the performance of another powerful NLP-based memetic algorithm, GB-MA [91], in four problems, namely g08, g10, g16, and g18.

### B. Engineering problems

Given the particularly favorable results obtained by mViE on the CEC 2006 benchmark, we decided to test the method also on a group of well-known engineering problems. Specifically, we tested mViE on a welded beam design problem, the optimization of a pressure vessel, the design of a stepped cantilever (reference formulations for these three problems can

be found in [3]), and the design of a tension compression spring [50].

For each engineering problem, we compared the median number of function evaluations needed by mViE to reach the optimum solution against the results reported in literature, as shown in Table VII (again, “-” indicates that the result is not available). We selected algorithms from the literature considering only those reporting the values for the best solution found. Furthermore, algorithms whose performance is completely dominated by other reported ones were ignored. Notably, the number of function evaluations needed to reach on average the best solution fitness value is normally not reported in papers dealing with engineering optimization. Therefore, in those cases we report the full budget of function evaluations given to the algorithm. On two out of four problems, mViE is capable of discovering the best known solution in the lowest number of function evaluations. Interestingly, in one problem, namely the design of a cantilever beam, mViE discovered a solution which is better than the known optimum reported in the literature. On the other hand, on the spring design problem mViE converges to the known optimal solution but is almost 2.8 times slower than the fastest algorithm, MBA [110].

For completeness, we report in Table VIII for each problem the fitness value, the constraint violation, and the values of the design variables of the best solutions found, together with the median NFES to reach them (over 25 runs).

### C. Sample algorithm runs

Overall, mViE compared very favorably on both the CEC 2006 benchmark problems and the four selected engineering problems. To provide an idea of the algorithm dynamics, we show in Fig. 4 the sample execution of our method on three selected problems, respectively g01, g02 and g10, where the latter one is unimodal. The convergence plots (Fig. 4a) and the variables used by the scheduler (Fig. 4b-c) to make informed decisions on the frequency of selection of local versus global search are shown. We also display the actual probability of executing local/global search (Fig. 4d) and the total number of function evaluations performed by local and global search (Fig. 4e). It is noticeable how on the unimodal problem (g10) a higher number of function evaluations are allocated to the local search units. Furthermore, problems characterized by the presence of distinct local optima (g02) present periods in which mViE tries to locally optimize a local optimum intertwined with phases of global exploration during which the algorithm is capable of escaping from local optima. In all the three cases, we should note that the limits imposed on the frequency of selection of the global/local steps are necessary to avoid the algorithm falling in a phase where only global or local search is used, without the possibility to switch.

### D. Performance dissection

Finally, we conclude this section by showing the contribution of the algorithmic components (local search, global search and scheduler) to the overall performance of mViE. To assess this, we performed three separated experiments. First, in the condition mViE-L we enabled only the local



TABLE VII

MEDIAN NFES AND BEST FITNESS ACHIEVED FOR THE ENGINEERING PROBLEMS. ALL THE COMPARED RESULTS ARE OBTAINED FROM THE CORRESPONDING PAPERS. PLEASE NOTE THAT IF THE MEDIAN NFES TO REACH THE OPTIMUM IS NOT REPORTED WE SHOW THE FULL BUDGET.  $\Delta$ NFES% INDICATES THE RELATIVE DIFFERENCE OF NFES BETWEEN THE PROPOSED ALGORITHM mViE AND THE BEST ALGORITHM THAT OBTAINS THE SAME BEST FITNESS (UNDERLINED). FOR THE CANTILEVER PROBLEM  $\Delta$ NFES% IS NOT DEFINED AS mViE OBTAINS A LOWER FITNESS VALUE THAN FA [108].

Prob.	mViE		DETPS [109]		PSO-DE [88]		MBA [110]		COPSO [7]		SiC-PSO [9]		FA [108]		$\Delta$ NFES%
	NFES	Fitness	NFES	Fitness	NFES	Fitness	NFES	Fitness	NFES	Fitness	NFES	Fitness	NFES	Fitness	
W. Beam	6568	1.724852	<u>10000</u>	1.724852	66600	1.724852	47340	1.724853	30000	1.724852	24000	1.724852	50000	1.731210	-34.32%
Vessel	14087	5850.38306	10000	5885.3336	42100	6059.714335	70650	5889.3216	30000	6059.714335	24000	6059.714335	<u>25000</u>	5850.38306	-43.65%
Spring	21413	0.012665	10000	0.012665	42100	0.012665	<u>7650</u>	0.012665	30000	0.012665	24000	0.012665	-	-	+179.90%
Cantilever	66894	63893.490839	-	-	-	-	-	-	-	-	-	-	50000	63893.52	-

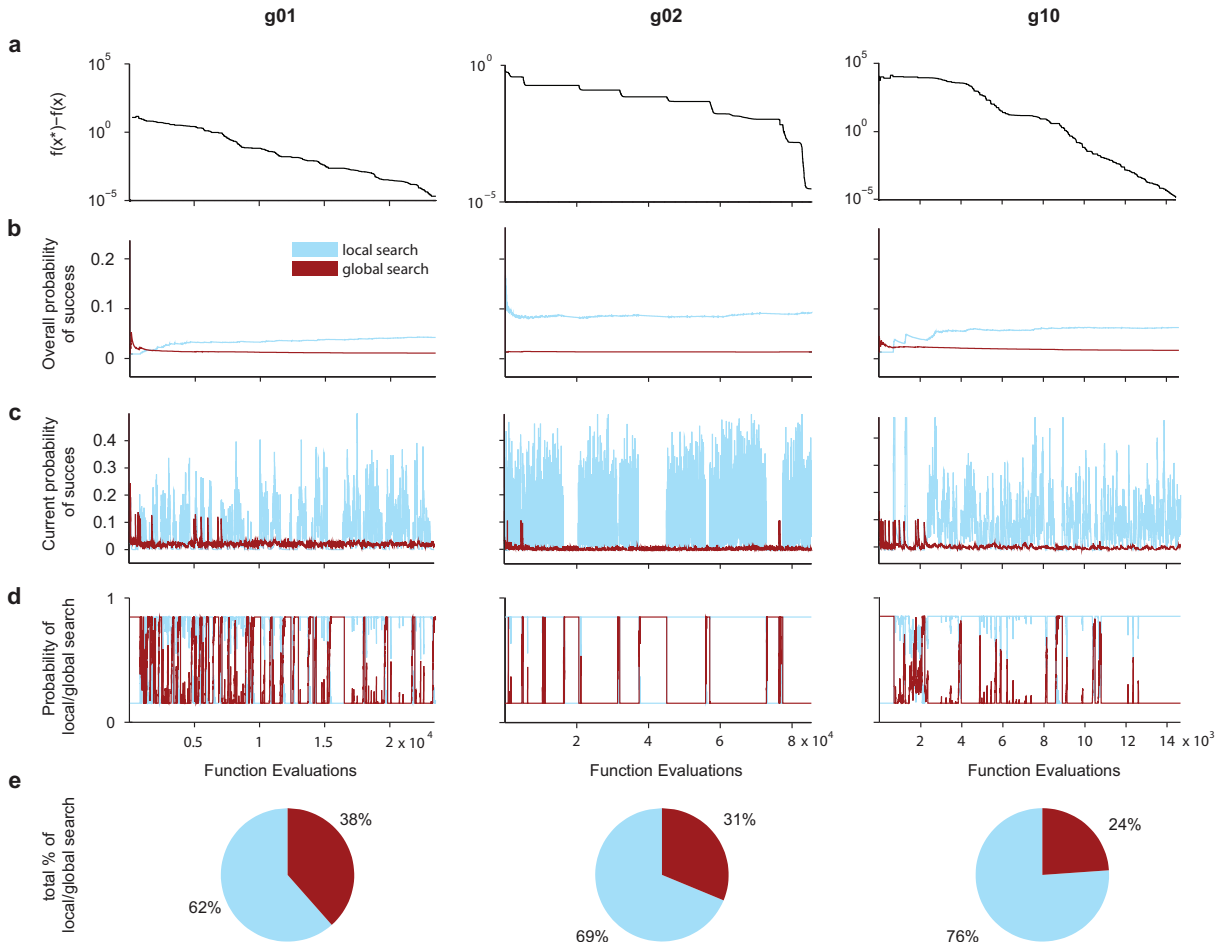


Fig. 4. Sample execution of the method on three functions taken from the CEC 2006 benchmark, namely  $g_{01}$ ,  $g_{02}$ , and  $g_{10}$ . We show **a)** the distance from the optimum, **b)** the overall probability of success of local and global search, **c)** the moving average of the probability of success of the two components, and **d)** their resulting frequency of execution. Finally, panel **e)** shows the resulting allocation of function evaluations to local and global search.

search component. Multiple units (with restart) were still used, but without applying global search. Second, in the condition mViE-G no local search unit was stepped, and only the global search component (DE operators) was used to explore the search space. Finally, in the third condition mViE-R both local and global components were used, coordinated by a trivial scheduler that randomly chooses the execution of global or local search at each iteration. We executed the algorithms corresponding to the three conditions for 25 runs on the selected CEC 2006 benchmark problems. Results are reported

in Fig. 5, that shows the comparison of the three conditions against the fully-featured mViE.

A rather evident result is that the success rates of the local and global search component alone are lower than those of the algorithms using the two components together, as expected. Furthermore, the amount of function evaluations used to discover the optimum is very high in the mViE-L condition due to the high number of restarts used by the algorithm before discovering a successful solution. Remarkably, the use of the two components allows the discovery of the optimum most of

TABLE VIII  
 FITNESS, CONSTRAINT VIOLATION (E), AND DESIGN VARIABLES ( $\mathbf{x}^*$ ) OF  
 THE BEST SOLUTIONS DISCOVERED FOR THE ENGINEERING PROBLEMS.

Problem	Best Fitness (Median NFES)	E	$\mathbf{x}^*$
Welded Beam	1.724852 (6568)	0	$h = 0.205729627974134$ $l = 3.470488964774360$ $t = 9.036623829898325$ $b = 0.205729643534243$
Pressure Vessel	5850.383060 (14087)	0	$T_s = 0.75$ $T_h = 0.375$ $R = 38.8601036269430$ $L = 221.3654713560083$
Helical Spring	0.012665 (21413)	0	$d = 0.051699916331388$ $D = 0.356978944672547$ $n = 11.273668588601133$
Stepped Cantilever	63893.490839 (66894)	0	$w_1 = 3$ $h_1 = 60$ $w_2 = 3.1$ $h_2 = 55$ $w_3 = 2.6$ $h_3 = 50$ $w_4 = 2.204553242032800$ $h_4 = 44.091064840656017$ $w_5 = 1.749768676906988$ $h_5 = 34.995373538139674$

the time, even using a random scheduler (mViE-R). However, the introduction of the adaptive scheduler allows a further  $\sim 25\%$  reduction in the number of function evaluations used.

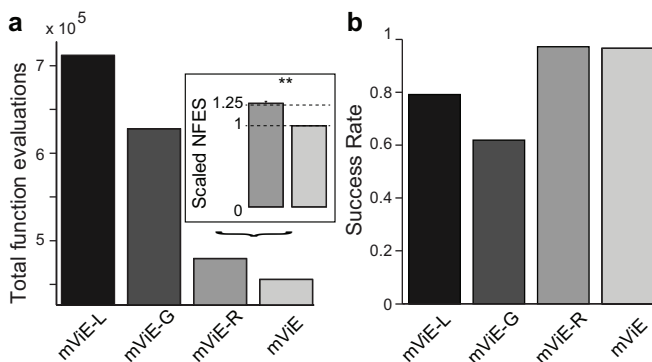


Fig. 5. Performance dissection over 25 runs of each of the selected CEC 2006 problems for the four experimental conditions. **a)** Total number of function evaluations (aggregate over all runs and problems) to reach for each problem the target fitness difference from the optimum ( $10^{-4}$ ). If the target is not reached, the maximum budget (250000 NFES) is added. The inset shows the scaled NFES, calculated as mean (over all problems) of the median NFES (calculated independently for each problem, considering only successful runs), scaled by the corresponding median NFES of mViE. mViE results  $\sim 25\%$  more efficient than mViE-R (Wilcoxon Rank-Sum test,  $\alpha = 0.05$ ,  $p = 0.0015$ ). **b)** Average success rate over all runs and problems.

## VII. DISCUSSION AND CONCLUSIONS

In this paper we presented mViE, a novel memetic computing approach for constrained optimization based on Viability Evolution. The proposed method is composed of multiple (1+1)-CMA-ES acting as local optimizers, that are combined with a Differential Evolution scheme to perform global search.

In numerical experiments, our method displayed a particularly robust performance, solving a broad number of test

functions (Tables III, V) and engineering design problems (Tables VII, VIII) more efficiently than the state-of-the-art compared methods, both in terms of success rate and number of function evaluations needed to reach the optimum. mViE outperformed the DE-based methods in all the comparisons (except for one case,  $g_{12}$ ), as well as the CMA-ES-based algorithms (except for three cases,  $g_{06}$ ,  $g_{08}$  and  $g_{10}$ ), the variants of PSO (except in one case,  $g_{19}$ ) and all the other algorithms under consideration.

Interestingly, the few cases in which mViE was outperformed correspond to problems where either strongly explorative algorithms (favoring global search only) or strongly exploitative ones (favoring local search) perform well enough. For example, the comparison with the algorithm we proposed in our previous work [45] revealed that on fitness landscapes such as  $g_{06}$  and  $g_{10}$  a single local search unit performs more efficiently than multiple ones coordinated by DE (as in mViE). We believe that in these cases the additional overhead introduced by the initial learning phase is the cause of this performance decay. On the other hand, under the general assumption that there is no prior knowledge on the features of the landscape, a learning phase might be necessary. Also, most of the problems are actually characterized by fitness landscapes where a proper trade-off between global and local search has to be found, and these are the cases where mViE excel.

In our opinion, the reason for the success of mViE is twofold: first, in the local search units the modelling of constraints as viability boundaries allows the search to be driven towards the feasible space and its most promising areas; secondly, the adaptive scheduler coordinates both the local search units and global recombination, thus enabling a synergistic exploration of the constrained search space. We discuss in detail these two aspects below.

1) *Modelling of constraints as viability boundaries*: as the local search units define viability boundaries separately on every constraint, they can collect additional information on each constraint. This information is therefore beneficial for a more fruitful and faster search, for example by adapting step-size and covariance matrix. On the contrary, approaches that combine constraints into the objective function or as a single constraint violation measure lose a potential source of information.

Dealing with constrained optimization problems without aggregating the constraints in the fitness function or in an aggregate constraint violation function unlocks additional information that is readily available to an evolutionary algorithm. Still, the large majority of approaches in the literature make use of some form of aggregation of constraints<sup>5</sup>. We therefore deemed useful to ask the question whether or not the current abstraction under which evolutionary algorithms operate is the most appropriate for optimization problems different from unconstrained ones. The logic behind most of the available constraint handling techniques, such as penalty functions, is in our view symptomatic of how the current evolutionary

<sup>5</sup>We also aggregate the constraints into a single constraint violation function when applying global search operators and when comparing against the global best solution in the main algorithmic loop. In a future version of mViE it may be beneficial to remove as well this form of aggregation.

algorithms paradigm may be misleading in the design of novel algorithms. Under this traditional paradigm, algorithms are designed for having solutions in competition with each other based on their fitness function values. Solutions are therefore ranked and compared uniquely using this single value. Intuitively modelled on a very high-level abstraction of natural evolution, this paradigm may hinder the development of the field towards more comprehensive paradigms and thus more powerful algorithms. In fact, the current abstraction forces researchers and adopters into thinking an evolutionary process as naturally modelled using a single fitness function.

On the other hand, Viability Evolution models an evolutionary process as elimination of solutions that do not comply with certain viability criteria, defined on both problem objectives and constraints. By adapting these criteria during the search, it is possible to drive the solutions towards desired areas of search space, typically the global optimum or feasible regions.

A first direct implication of the Viability Evolution paradigm is that constraints and objectives are kept implicitly separated<sup>6</sup>. Also, the “fitness” of individuals is in this case a property measured *a-posteriori* and not defined *a-priori* as done by using a fitness function in the classic sense. Furthermore, the availability of statistics on the viability of solutions made possible by this different abstraction, e.g. the number of individuals satisfying specific viability criteria or the number of viable/non-viable individuals, leads to increased information available for evolutionary methods. Third, more emphasis on elimination of non-viable solutions rather than competition of solutions by a unique fitness score may lead to enhanced diversity in the evolving population [47].

Here, we showed that by following the design principles of Viability Evolution it is possible to derive a very efficient method for constrained optimization. Overall our method models constraints separately and more importantly uses information about constraint violations (non-viability of individuals) for adapting the algorithm parameters during the search, without requiring a user to aggregate them.

2) *Global recombination and memetic adaptation*: a key element of mViE is the scheduler, that adaptively activates local or global search, providing a proper balance between the two regimes. An important advantage of this scheme is the fact that it has *direct* control over the budget assigned to both the global and local search operators. This feature makes our scheduler different from most of the existing credit assignment mechanisms used in memetic algorithms, which in general balance the budget assigned to different local search methods (memes), while they do not regulate directly the budget assigned to the global search operators, see e.g. [73]. One exception is the method presented in [75], which uses a measure called *local search intensity* for allocating the budget assigned to each meme in a pool of local search methods. However, also in this case the control over the global search budget is implicit rather than explicit,

i.e. global and local search are always executed at each step and only the number of function evaluations allotted to local search is adjusted. We believe, instead, that a direct control of both global and local search function evaluation budgets is of fundamental importance especially in constrained optimization scenarios, where it may be needed to adjust dynamically the exploration pressure. Overall, our algorithm provides two levels of adaptation: the first level is represented by our scheduler, whereas the second level consists of the use of self-adapting local search units, that learn their parameters during the search. Following the taxonomy provided in [74], our method can be therefore classified as a *self-adaptive* algorithm (because multiple (1+1)-ViE units coevolve, adapting independently their parameters), with adaptation at both *local* and *global* level: our method employs in fact a mix of recent information on the performance of local and global search (the moving averages of the probabilities of success) and complete historical information (the absolute number of successes of each operator).

To conclude, the present work contributes to the field of constrained optimization and suggests a wide spectrum of possible research lines that are worth following, going in the direction of: 1) extending the viability concept to different classes of problems, such as large-scale, multi-objective, and dynamic optimization; 2) testing alternative recombination schemes, based for instance of swarm intelligence, to coordinate the multiple local search operators; and 3) applying the proposed method to real-world applications where a resource-efficient constrained optimization solver might be needed, for instance in various domains of engineering or computational biology.

## APPENDIX A PARAMETER ANALYSIS

Given the impossibility of performing a full combinatorial exploration of the parameter space, we executed a preliminary tuning of the main parameters by following an iterative procedure. First, we set the initial parameters by empirically experimenting with the algorithm. We then analyzed the influence on the algorithmic performance varying each parameter, fixing the identified optimal parameter values in several steps. We investigated independently  $c_\alpha$  and  $\beta_R$ , secondly  $L$ ,  $F$  and  $CR$ , and finally  $pop_{size}$ .

We evaluated the algorithm's performance by testing it on the full CEC 2006 benchmark (with inequalities only) for 25 runs. We allowed the algorithm to run for 150000 function evaluations. We measured the success rate (SR) for each function and the factor of number of function evaluations (NFES) with respect to the ones achieved by the best algorithm of the CEC 2006 competition to reach the optimum at the accuracy of  $10^{-4}$ . To obtain a more robust evaluation of the success rate, we tested each parameter combination 5 times, for a total of 5 repetitions  $\times$  25 runs  $\times$  13 benchmark problems. We aggregated SR and NFES factor across all problems: we summed the rank of a parameter configuration calculated considering the success rate (higher SR are assigned better ranks) and the NFES factor (lower NFES factors are assigned

<sup>6</sup>A similar shift in paradigm was observed in multi-objective evolutionary algorithms (MOEAs), when classic aggregation methods such as weighted summing of objectives were replaced by the use of Pareto optimality concepts. Such a shift led to radically novel MOEAs, eventually obtaining dramatic performance improvements.

better ranks). To compute a single score for each parameter configuration, we summed the aggregate rank obtained by each parameter setting on all the tested problems.

The results of this parameter tuning procedure are shown in Fig. 6. Success rate, mean factor of NFES and aggregate rank of each parameter combination are shown when varying  $c_\alpha$  and  $\beta_R$  (Fig. 6a), the minimum relative execution frequency limit  $L$  (Fig. 6b),  $F$  and  $CR$  (Fig. 6c), and the population size (Fig. 6d). The final identified parameter values are  $c_\alpha = 0.1$ ,  $\beta_R = 0.05$ ,  $L = 0.18$ ,  $F = 0.5$ ,  $CR = 0.9$ ,  $P_{size} = 40$ . Although this may be a locally optimal combination of parameters and better parameter tuning may be achieved by a more thorough parameter exploration, we considered this procedure sufficient for our purposes. Also, we obtained an idea of how the method's performance is affected by changing its parameters.

More specifically, the first two parameters ( $c_\alpha$  and  $\beta_R$ ), which regulate the way the success probability of global and local search is updated (which, in turn, affect the information used by the scheduler to allocate function evaluations to the two components), seem to have a large impact on the performance. We can see from Fig. 6a that the highest values of SR are obtained when at least one of the two parameters has a very low value, namely  $c_\alpha \in [0, 0.1]$  or  $\beta_R \in [0, 0.1]$ . Values lying at the intersection of these two regions, e.g.  $c_\alpha = 0.1$  and  $\beta_R = 0.05$  (the values we chose in our experimentation), provide then among the highest SR. Interestingly, these values provide also the lowest NFES factor, and therefore the best rank. We further verified this setting looking at the performance on each single problem obtained with different values of  $c_\alpha$  and  $\beta_R$  (plots are not reported here for brevity). Interestingly, the SR landscape has a completely flat surface for g1, g4, g6, g8, g12, g16 and g24. In the other cases, it has a plateau with maximum values on the borders and a pronounced valley towards the region  $c_\alpha > 0.3$  and  $\beta_R > 1$ . For almost all problems (apart from g19), the NFES factor landscape has a large plateau with maxima (for g1, g2 and g12) or minima (all other cases) at the border regions. Overall, the setting corresponding to values in  $[0.05, 0.1]$  for both  $c_\alpha$  and  $\beta_R$  is considered the most robust and should be used in practical applications. Higher values for both parameters make the previous information on the success probability fade faster (see Equations 12-15), thus reducing the effect of the learning for the scheduler and producing a less efficient switch between global and local search.

As seen in Fig. 6b, the limit  $L$  on the frequency of activation of global/local search has a very limited impact on the performance. For  $L \in [0, 0.4]$ , the success rate is always between 0.98 and 0.99, while the NFES factor seems to be a bit lower for  $L \geq 0.15$ . The combined ranking information does not show a clear pattern in this case. We chose the value  $L = 0.18$  because it seems to provide the best trade-off SR-NFES. In general, we recommend using the setting  $L \in [0.15, 0.4]$ , which should be relatively robust on different classes of problems. Values outside this range make the frequency of activation of either local or global search too low ( $L < 0.15$ ) or too high ( $L > 0.4$ ) thus unbalancing the NFES allocated to the two search regimes.

Considering  $F$  and  $CR$ , i.e. the two parameters involved

in the DE-based global search operators, we know from the literature [97] that a generally recommended setting is  $F = 0.5$  and  $CR = 0.9$ . Nevertheless, we verified this setting experimentally as seen in Fig. 6c. From the figure, we can observe that the SR landscape is characterized by a peak in the region  $F \in [0.4, 0.7]$  and  $CR \in [0.8, 1.0]$ . The NFES factor landscape has instead a large plateau with a small peak corresponding to  $F \in [0.2, 0.3]$  and  $CR \in [0.9, 1.0]$ . Finally, the rank landscape (aggregating the info on SR and NFES) suggests that the minimum rank (i.e., the best rank) is in the region around  $CR = 0.9$  and  $F \in [0.5, 0.6]$ . Overall, our experiments confirm that the most robust setting corresponds to  $F = 0.5$  and  $CR = 0.9$ . We believe that this setting should be used in all applications of our method. Using different values for either  $F$  or  $CR$  has the effect of making mutation and crossover too disruptive (or not effective enough).

Finally, as for the population size (i.e., the number of local search units), we can see from Fig. 6d that the parameter value chosen,  $pop_{size} = 40$ , guarantees the best trade-off SR-NFES for all problems. Indeed, the SR hits the maximum at  $pop_{size} = 40$ , while the NFES factor has a minimum for  $pop_{size} \in [30, 40]$ . Overall, we suggest using values in this range, which seem robust enough to handle all the problems. Intuitively, smaller populations have a more limited exploration and may fail on highly multi-modal landscapes. Also too large populations have in our case a limited advantage, as the initial probability of selecting one local search unit decreases with the number of units in the population, and in later stages it is likely that only few successful units are selected.

## APPENDIX B

### CEC 2006 PROBLEM RESULTS - ERROR VALUES ACHIEVED AT DIFFERENT LEVEL OF NFES

Table IX reports the best, median, worst and mean error values, i.e. the difference between the known optimum fitness and the best fitness achieved by mViE, at different NFES. Furthermore, the number of violated constraints  $c$  and the average sum of constraint violation  $\bar{v}$  at the median solution (at the end of each NFES level) are shown.

## ACKNOWLEDGMENT

We thank Pavan Ramdya for useful comments on the manuscript. The computations were performed on the EPFL HPC Cluster "Aries" (<http://scitas.epfl.ch>). This research has been supported by the Swiss National Science Foundation, grant number 141063.

## REFERENCES

- [1] Z. Michalewicz, D. Dasgupta, R. L. Riche, and M. Schoenauer, "Evolutionary algorithms for constrained engineering problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 851–870, Sept. 1996.
- [2] K. Rasheed, "An adaptive penalty approach for constrained genetic-algorithm optimization," in *Proc. 3rd Annu. Genetic Programming Conf.*, 1998, pp. 584–590.
- [3] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Comput. Ind.*, vol. 41, no. 2, pp. 113–127, Mar. 2000.
- [4] C. A. Coello Coello and E. Mezura-Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Adv. Eng. Inform.*, vol. 16, no. 3, pp. 193–203, July 2002.



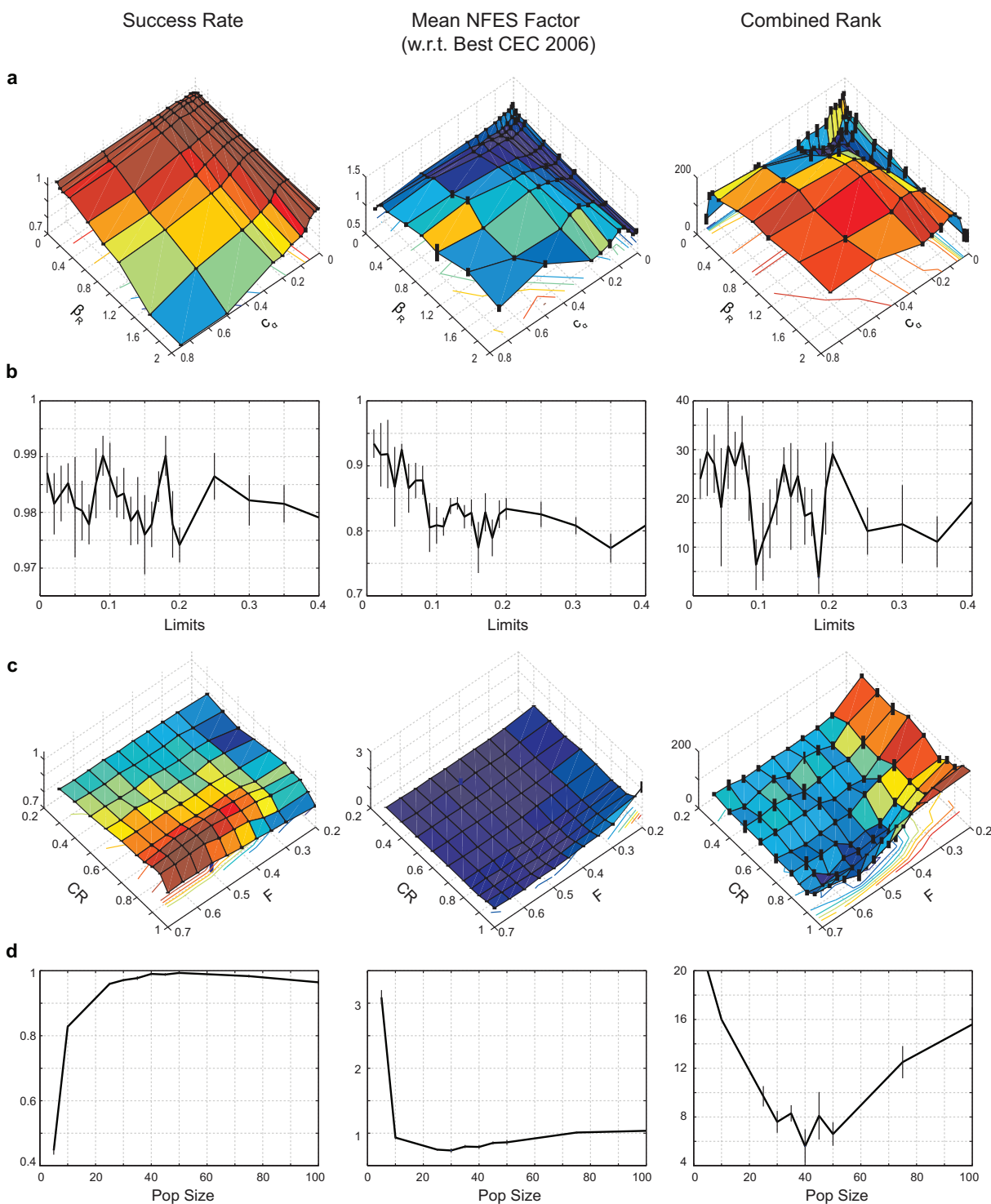


Fig. 6. Parameter analysis: each reported result is the aggregation obtained by repeating 25 times each function (with inequalities only) of the CEC 2006 benchmark for 5 independent repetitions (i.e. 13 benchmark functions  $\times$  25 runs  $\times$  5 repetitions). The standard deviation across the 5 repetitions is reported as a black bar for each parameter combination. We show mean success rate across the 13 benchmark functions and mean factor of function evaluations computed with respect to the best algorithm in the CEC 2006 competition. To select the best parameter combination for each of the parameters, we independently rank each parameter combination by success rate and mean factor of function evaluations. In the third column, we show the combined (summation) rank for each parameter combination. We report four parameter studies for **a**) parameters regulating the update rules of the success probability,  $c_\alpha$  and  $\beta_R$ ; **b**) minimum global/local search execution frequency limit  $L$ ; **c**) DE operators' parameters  $F$  and  $CR$ ; and **d**) number of local search units, i.e. population size  $pop_{size}$ .

TABLE IX

ERROR VALUES (DIFFERENCE BETWEEN THE KNOWN OPTIMUM FITNESS AND THE BEST FITNESS VALUE) ACHIEVED WHEN NFES=  $5 \times 10^3$ , NFES=  $5 \times 10^4$ , AND NFES=  $5 \times 10^5$  FOR THE SELECTED CEC 2006 PROBLEMS. BEST, MEDIAN, WORST AND MEAN ERROR VALUES ARE REPORTED IN THE TABLE, TOGETHER WITH THE NUMBER OF VIOLATED CONSTRAINTS  $c$  AND THE AVERAGE SUM OF CONSTRAINT VIOLATION  $\bar{v}$  OF THE MEDIAN SOLUTION.

NFES		g01	g02	g04	g06	g07
$5 \times 10^3$	Best	0.695563 (0)	0.132752 (0)	0 (0)	0 (0)	0.001159 (0)
	Median	1.91028 (0)	0.302738 (0)	1e-06 (0)	0 (0)	0.015729 (0)
	Worst	3.7779 (0)	0.418978 (0)	0.202275 (0)	0.02595 (0)	1.47459 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	2.429400e-02	0	0	6.130000e-04	2.052570e-01
	Mean	2.1593	0.287957	0.00825104	0.00141272	0.110063
	Std	0.822658	0.0663392	0.0404259	0.00529882	0.307058
$5 \times 10^4$	Best	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0.011011 (0)	0 (0)	0 (0)	0 (0)
	Worst	7.6e-05 (0)	0.132974 (0)	0 (0)	0 (0)	0.0001 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	4.013410e-01	0	0	1.375290e-01	3.400200e-01
	Mean	1.968e-05	0.0228126	0	0	4e-06
	Std	2.7308e-05	0.0296265	0	0	2e-05
$5 \times 10^5$	Best	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Worst	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	4.964673e+00	0	6.900000e-05	8.971150e+00	4.285058e+00
	Mean	0	0	0	0	0
	Std	0	0	0	0	0

NFES		g08	g09	g10	g12	g16
$5 \times 10^3$	Best	0 (0)	0 (0)	10.0031 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	216.449 (0)	0 (0)	0 (0)
	Worst	0 (0)	0.410366 (0)	4434.49 (0)	0.005625 (0)	0.000864 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	9.640000e-04	0	1.444769e+03	0	1.902000e-03
	Mean	0	0.0224805	922.665	0.00116472	8.572e-05
	Std	0	0.0833781	1447.02	0.00227881	0.000239219
$5 \times 10^4$	Best	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	3e-06 (0)	0 (0)	0 (0)
	Worst	0 (0)	0 (0)	0.265618 (0)	0 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	1.057000e-03	0	2.776000e-03	2.800000e-05	1.864644e+01
	Mean	0	0	0.01235	0	0
	Std	0	0	0.0530731	0	0
$5 \times 10^5$	Best	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	Worst	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	1.057000e-03	9.073334e+00	3.134842e+03	2.800000e-05	1.082927e+02
	Mean	0	0	0	0	0
	Std	0	0	0	0	0

NFES		g18	g19	g24
$5 \times 10^3$	Best	5e-05 (0)	13.4913 (0)	0 (0)
	Median	0.010392 (0)	43.6379 (0)	0 (0)
	Worst	0.595968 (3)	107.837 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	4.908650e-01	0	0
	Mean	0.095619	46.6881	0
	Std	0.154321	19.8156	0
$5 \times 10^4$	Best	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	0 (0)
	Worst	0.191044 (0)	2e-06 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	4.561900e-01	0	1.494000e-03
	Mean	0.0152835	8e-08	0
	Std	0.0528977	4e-07	0
$5 \times 10^5$	Best	0 (0)	0 (0)	0 (0)
	Median	0 (0)	0 (0)	0 (0)
	Worst	0 (0)	0 (0)	0 (0)
	$c$	0, 0, 0	0, 0, 0	0, 0, 0
	$\bar{v}$	7.948723e+00	0	4.546000e-03
	Mean	0	0	0
	Std	0	0	0

- [5] E. Mezura-Montes and C. A. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 1–17, Feb. 2005.
- [6] X. Hu, R. C. Eberhart, and Y. Shi, "Engineering optimization with particle swarm," in *Proc. IEEE Swarm Intell. Symp.*, Apr. 2003, pp. 53–57.
- [7] A. Aguirre, A. Muñoz Zavala, E. Villa Diharce, and S. Botello Rionda, "COPSO: Constrained optimization via PSO algorithm," Center of Research in Mathematics (CIMAT), Guanajuato, México, Technical Report I-07-04, 2007.
- [8] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Eng. Appl. Artif. Intel.*, vol. 20, no. 1, pp. 89–99, Feb. 2007.
- [9] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello, "Solving engineering optimization problems with the simple constrained particle swarm optimizer," *Informatica*, vol. 32, no. 3, pp. 319–326, 2008.
- [10] M. Pant, R. Thangaraj, and A. Abraha, "Low discrepancy initialized particle swarm optimization for solving constrained optimization problems," *Fund. Inform.*, vol. 95, no. 4, pp. 511–531, Dec. 2009.
- [11] L. d. S. Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1676–1683, Mar. 2010.
- [12] G. Leguizamón and C. A. Coello Coello, "Boundary search for constrained numerical optimization problems with an algorithm inspired on the ant colony metaphor," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 350–368, Apr. 2009.
- [13] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *Int. J. of Math. Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, Oct. 2010.
- [14] I. Brajevic, M. Tuba, and M. Subotic, "Performance of the improved artificial bee colony algorithm on standard engineering constrained problems," *Math. Comput. Simulat.*, vol. 5, no. 1, pp. 135–143, 2011.
- [15] E. Cuevas and M. Cienfuegos, "A new algorithm inspired in the behavior of the social-spider for constrained optimization," *Expert Syst. Appl.*, vol. 41, no. 2, pp. 412–425, 2014.
- [16] W. Zhang, G. G. Yen, and Z. He, "Constrained optimization via artificial immune system," *IEEE Trans. Cybern.*, vol. 44, no. 2, pp. 185–198, Feb. 2014.
- [17] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," in *Proc. 4th Annu. Conf. Evolutionary Programming*, Cambridge, MA, USA, 1995, pp. 135–155.
- [18] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Comput. Method Appl. M.*, vol. 191, no. 11–12, pp. 1245–1287, Jan. 2002.
- [19] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evol. Comput.*, vol. 1, no. 4, pp. 173–194, Dec. 2011.
- [20] B. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Trans. Syst., Man Cybern. A, Syst. Humans*, vol. 39, no. 3, pp. 565–578, May 2009.
- [21] O. Kramer, U. Schlachter, and V. Spreckels, "An adaptive penalty function with meta-modeling for constrained problems," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1350–1354.
- [22] S. Salcedo-Sanz, "A survey of repair methods used as constraint handling techniques in evolutionary algorithms," *Computer Science Review*, vol. 3, no. 3, pp. 175–192, Aug. 2009.
- [23] S. Wessing, "Repair methods for box constraints revisited," in *Proc. 16th European Conf. Appl. Evol. Computation*, Berlin, Heidelberg, 2013, pp. 469–478.
- [24] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sept. 2000.
- [25] T. Takahama and S. Sakai, "Constrained optimization by the  $\epsilon$  constrained differential evolution with an archive and gradient-based mutation," in *Proc. IEEE Congr. Evol. Comput.*, July 2010, pp. 1–9.
- [26] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. methods in applied mechanics and engineering*, vol. 186, no. 2–4, pp. 311–338, June 2000.
- [27] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evol. Comput.*, vol. 1, no. 2, pp. 61–70, June 2011.
- [28] R. L. While and P. Hingston, "Usefulness of infeasible solutions in evolutionary search: An empirical and mathematical study," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1363–1370.
- [29] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 80–92, Feb. 2008.
- [30] A. Angantyr, J. Andersson, and J.-O. Aidanpaa, "Constrained optimization based on a multiobjective evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, Dec. 2003, pp. 1560–1567.
- [31] E. Mezura-Montes and C. A. C. Coello, "Constrained optimization via multiobjective evolutionary algorithms," in *Multiobjective Problem Solving from Nature*, ser. Natural Computing Series, J. Knowles, D. Corne, K. Deb, and D. Chair, Eds. Springer Berlin Heidelberg, 2008, pp. 53–75.
- [32] L. Clevenger, L. Ferguson, and W. E. Hart, "A filter-based evolutionary algorithm for constrained optimization," *Evol. Comput.*, vol. 13, no. 3, pp. 329–352, Sep. 2005.
- [33] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 658–675, Dec. 2006.
- [34] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization." Nanyang Technological University, Singapore, Tech. Rep., 2006.
- [35] P. S. R. Mallipeddi, "Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2010.
- [36] N. Hansen, S. D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, 2003.
- [37] O. Kramer, A. Barthelme, and G. Rudolph, "Surrogate constraint functions for CMA evolution strategies," in *KI 2009: Advances in Artificial Intell.*, vol. 5803, 2009, pp. 169–176.
- [38] G. Collange, N. Delattre, N. Hansen, I. Quinquis, and M. Schoenauer, "Multidisciplinary optimization in the design of future space launchers," in *Multidisciplinary Design Optimization in Computational Mechanics*. John Wiley & Sons, Inc., 2010, pp. 459–468.
- [39] A. O. Kusacki and M. Can, "An adaptive penalty based covariance matrix adaptation-evolution strategy," *Comput. Oper. Res.*, vol. 40, no. 10, pp. 2398–2417, Oct. 2013.
- [40] F. Gieseke and O. Kramer, "Towards non-linear constraint estimation for expensive optimization," in *Applications of Evolutionary Computation*, A. Esparcia-Alcázar, Ed. Springer Berlin Heidelberg, 2013, pp. 459–468.
- [41] V. V. de Melo and G. Iacca, "A modified covariance matrix adaptation evolution strategy with adaptive penalty function and restart for constrained optimization," *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7077–7094, Nov. 2014.
- [42] H.-G. Beyer and S. Finck, "On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 578–596, Aug. 2012.
- [43] D. V. Arnold and N. Hansen, "A (1+1)-CMA-ES for constrained optimisation," in *Proc. 14th Annu. Conf. Genetic and Evol. Computation*, New York, NY, USA, 2012, pp. 297–304.
- [44] C. Igel, T. Suttorp, and N. Hansen, "A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies," in *Proc. 8th Annu. Conf. Genetic and Evol. Computation*, New York, NY, USA, 2006, pp. 453–460.
- [45] A. Maesani and D. Floreano, "Viability principles for constrained optimization using a (1+1)-CMA-ES," in *Parallel Problem Solving from Nature*, vol. 8672, 2014, pp. 272–281.
- [46] C. Mattiussi and D. Floreano, "Viability evolution: Elimination and extinction in evolutionary computation," EPFL, Tech. Rep., 2003, Available from: <http://infoscience.epfl.ch/record/177577>.
- [47] A. Maesani, P. R. Fernando, and D. Floreano, "Artificial evolution by viability rather than competition," *PLoS ONE*, vol. 9, no. 1, p. e86831, Jan. 2014.
- [48] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
- [49] E. Mezura-Montes and B. C. Lopez-Ramirez, "Comparing bio-inspired algorithms in constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 662–669.
- [50] A. H. Gandomi and X.-S. Yang, "Benchmark problems in structural optimization," in *Computational Optimization, Methods and Algorithms*, S. Koziel and X.-S. Yang, Eds. Berlin, Germany: Springer, 2011, ch. 12, pp. 259–281.

- [51] S. B. Hamida and M. Schoenauer, "ASCEA: New Results Using Adaptive Segregational Constraint Handling," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, 2002, pp. 884–889.
- [52] A. O. Kusakci and M. Can, "A novel evolution strategy for constrained optimization in engineering design," in *XXIV Int. Symp. Inform., Commun. and Automation Technologies (ICAT)*, 2013, pp. 1–6.
- [53] E. Mezura-Montes, C. Coello Coello, and E. Tun-Morales, "Simple feasibility rules and differential evolution for constrained optimization," in *MICAI 2004: Advances in Artificial Intell.*, ser. Lecture Notes in Comput. Science, R. Monroy, G. Arroyo-Figueroa, L. Sucar, and H. Sossa, Eds. Springer Berlin Heidelberg, 2004, vol. 2972, pp. 707–716.
- [54] E. Mezura-Montes, C. A. Coello Coello, and J. Velázquez-Reyes, "Increasing successful offspring and diversity in differential evolution for engineering design," in *Proc. 7th Int. Conf. Adaptive Computing in Design and Manufacture*, 2006, pp. 131–139.
- [55] R. L. Becerra and C. A. Coello Coello, "Cultured differential evolution for constrained optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33-36, pp. 4303–4322, July 2006.
- [56] T. Takahama and S. Sakai, "Constrained optimization by the  $\epsilon$  constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 1–8.
- [57] E. Mezura-Montes, H. Velazquez-Reyes, and C. A. Coello Coello, "Modified differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 25–32.
- [58] V. Huang, A. Qin, and P. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 17–24.
- [59] E. Mezura-Montes and A. G. Palomeque-Ortiz, "Parameter control in differential evolution for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1375–1382.
- [60] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gomez-Ramon, "Differential evolution in constrained numerical optimization: An empirical study," *Inform. Sciences*, vol. 180, no. 22, pp. 4223–4262, Nov. 2010.
- [61] E. Mezura-Montes and A. Palomeque-Ortiz, "Self-adaptive and deterministic parameter control in differential evolution for constrained optimization," in *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence, E. Mezura-Montes, Ed. Springer Berlin Heidelberg, 2009, vol. 198, pp. 95–120.
- [62] D. Zou, H. Liu, L. Gao, and S. Li, "A novel modified differential evolution algorithm for constrained optimization problems," *Comput. Math. Appl.*, vol. 61, no. 6, pp. 1608–1623, Mar. 2011.
- [63] A. W. Mohamed and H. Z. Sabry, "Constrained optimization based on modified differential evolution algorithm," *Inform. Sciences*, vol. 194, pp. 171–208, July 2012.
- [64] X. Kong, H. Ouyang, and X. Piao, "A prediction-based adaptive grouping differential evolution algorithm for constrained numerical optimization," *Soft Comput.*, vol. 17, no. 12, pp. 2293–2309, Dec. 2013.
- [65] V. V. de Melo and G. L. C. Carosio, "Evaluating differential evolution with penalty function to solve constrained engineering problems," *Expert Syst. Appl.*, vol. 39, no. 9, pp. 7860–7863, July 2012.
- [66] M. Ali and W. Zhu, "A penalty function-based differential evolution algorithm for constrained global optimization," *Comput. Optim. Appl.*, vol. 54, no. 3, pp. 707–739, Apr. 2013.
- [67] W. Long, X. Liang, Y. Huang, and Y. Chen, "A hybrid differential evolution augmented lagrangian method for constrained numerical and engineering optimization," *Comput. Aided Design*, vol. 45, no. 12, pp. 1562–1574, Dec. 2013.
- [68] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 561–579, Aug. 2010.
- [69] T. Takahama and S. Sakai, "Efficient constrained optimization by the  $\epsilon$  constrained rank-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.
- [70] —, "Efficient constrained optimization by the  $\epsilon$  constrained differential evolution with rough approximation using kernel regression," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1334–1341.
- [71] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.
- [72] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation - past, present & future," *IEEE IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.
- [73] Y.-S. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, April 2004.
- [74] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb 2006.
- [75] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, June 2009.
- [76] J. Smith, "Coevolving memetic algorithms: A review and progress report," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 6–17, Feb 2007.
- [77] Z. Zhou, Y.-S. Ong, P. Nair, A. Keane, and K.-Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 1, pp. 66–76, Jan 2007.
- [78] G. Iacca, F. Neri, E. Mininno, Y.-S. Ong, and M.-H. Lim, "Ockham's razor in memetic computing: three stage optimal memetic exploration," *Inform. Sciences*, vol. 188, pp. 17–43, 2012.
- [79] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct 2011.
- [80] F. Neri, C. Cotta, and P. Moscato, Eds., *Handbook of Memetic Algorithms*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 379.
- [81] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, 2003.
- [82] A. S. S. M. B. Ullah, R. Sarker, D. Cornforth, and C. Lokan, "AMA: a new approach for solving constrained real-valued optimization problems," *Soft Comput.*, vol. 13, no. 8–9, pp. 741–762, July 2009.
- [83] A. Barkat Ullah, R. Sarker, and C. Lokan, "Handling equality constraints with agent-based memetic algorithms," *Memetic Computing*, vol. 3, no. 1, pp. 51–72, Mar. 2011.
- [84] M. Pescador Rojas and C. A. Coello Coello, "A memetic algorithm with simplex crossover for solving constrained optimization problems," in *World Automation Congr.*, 2012, pp. 1–6.
- [85] H. Bernardino, H. Barbosa, and A. Lemong, "A hybrid genetic algorithm for constrained optimization problems in mechanical engineering," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 646–653.
- [86] Q. He and L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," *Appl. Math. Comput.*, vol. 186, no. 2, pp. 1407–1422, Mar. 2007.
- [87] T. Takahama, S. Sakai, and N. Iwano, "Constrained optimization by the  $\epsilon$  constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *AI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Comput. Science, S. Zhang and R. Jarvis, Eds. Springer Berlin Heidelberg, 2005, vol. 3809, pp. 389–400.
- [88] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Appl. Soft Comput.*, vol. 10, no. 2, pp. 629–640, Mar. 2010.
- [89] G. Jia, Y. Wang, Z. Cai, and Y. Jin, "An improved  $(\mu + \lambda)$ -constrained differential evolution for constrained optimization," *Inform. Sciences*, vol. 222, pp. 302–322, Feb. 2013.
- [90] Y. Wang and Z. Cai, "Constrained evolutionary optimization by means of  $(\mu + \lambda)$ -differential evolution and improved adaptive trade-off model," *Evol. Comput.*, vol. 19, no. 2, pp. 249–285, 2011.
- [91] J. Sun and J. M. Garibaldi, "A novel memetic algorithm for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, July 2010, pp. 549–556.
- [92] S. Handoko, C. K. Kwok, and Y.-S. Ong, "Feasibility structure modeling: An effective chaperone for constrained memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 740–758, Oct 2010.
- [93] N. M. Hamza, R. A. Sarker, D. L. Essam, K. Deb, and S. M. Elsayed, "A constraint consensus memetic algorithm for solving constrained optimization problems," *Eng. Optimiz.*, vol. 46, no. 11, pp. 1447–1464, Nov. 2014.
- [94] M. Bonyadi, X. Li, and Z. Michalewicz, "A hybrid particle swarm with velocity mutation for constraint optimization problems," in *Proc. 15th Annu. Conf. Genetic and Evol. Comput.*, 2013, pp. 1–8.



- [95] D. V. Arnold and N. Hansen, "Active covariance matrix adaptation for the (1+1)-CMA-ES," in *Proc. 12th Annu. Conf. Genetic and Evol. Computation*, New York, NY, USA, 2010, pp. 385–392.
- [96] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 22–34, Apr. 1999.
- [97] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer Berlin Heidelberg, 2005.
- [98] M. Tasgetiren and P. Suganthan, "A multi-populated differential evolution algorithm for solving constrained optimization problem," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 33–40.
- [99] A. E. Smith and D. Tate, "Genetic optimization using a penalty function," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 499–505.
- [100] S. Kukkonen and J. Lampinen, "Constrained real-parameter optimization with generalized differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, July 2006, pp. 911–918.
- [101] J. Brest, V. Zumer, and M. Maucec, "Self-adaptive differential evolution algorithm in constrained real-parameter optimization," in *2006 IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 215–222.
- [102] K. Zielinski and R. Laur, "Constrained single-objective optimization using particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., 2006, pp. 443–450.
- [103] A. Munoz-Zavala, A. Hernandez-Aguirre, E. Villa-Diharce, and S. Botello-Rionda, "PESO+ for constrained optimization," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B.-T. Zhang, C. A. Coello Coello, and T. P. Runarsson, Eds., 2006, pp. 231–238.
- [104] T. Runarsson, "Approximate evolution strategy using stochastic ranking," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 745–752.
- [105] A. Sinha, A. Srinivasan, and K. Deb, "A population-based, parent centric procedure for constrained real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 239–245.
- [106] J. Liang and P. Suganthan, "Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism," in *Proc. IEEE Congr. Evol. Comput.*, G. G. Yen, S. M. Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. A. Coello Coello, and T. D. Runarsson, Eds., 2006, pp. 9–16.
- [107] T. P. Runarsson and X. Y., "Search biases in constrained evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 233–243, May 2005.
- [108] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Comput. Struct.*, vol. 89, no. 23, pp. 2325–2336, Dec. 2011.
- [109] G. Zhang, J. Cheng, M. Gheorghe, and Q. Meng, "A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1528–1542, Mar. 2013.
- [110] A. Sadollah, A. Bahreinejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2592–2612, May 2013.



**Andrea Maesani** holds a Ph.D. in Computer, Communication and Information Sciences from the Swiss Federal Institute of Technology, Lausanne (EPFL) and a M.Sc. in Computer Engineering from Politecnico di Milano, Italy. During his doctoral studies, his research interests included stochastic optimization methods and their application to biological and computational neuroscience problems.



**Giovanni Iacca** (M'11) received the M.Sc. (2006) in Computer Engineering from Politecnico di Bari, Italy. In 2011, he earned a Ph.D. in Computational Intelligence from the University of Jyväskylä, Finland. Since 2012, he has been a scientific researcher at INCAS<sup>3</sup> in Assen, The Netherlands. Currently, he holds a joint postdoctoral position with the University of Lausanne and EPFL. His research interests include evolutionary computation, stochastic optimization, and robotics.



**Dario Floreano** (SM'05) is Director of the Laboratory of Intelligent Systems at EPFL Switzerland and Director of the Swiss National Center of Robotics. His research focuses on the convergence of biology, artificial intelligence, and robotics. He has published more than 300 peer-reviewed papers and four books on the topics of evolutionary robotics, bio-inspired artificial intelligence, and bio-mimetic flying robots with MIT Press and Springer Verlag.