A Surrogate-assisted Partial Optimization for Expensive Constrained Optimization Problems

Kei Nishihara^{1}[0009−0006−2610−9276] and Masaya Nakata^{1}[0000−0003−3428−7890]

Yokohama National University, Yokohama, 2408501 Kanagawa, Japan
nishihara-kei-jv@ynu.jp, nakata-masaya-tb@ynu.ac.jp

Abstract. Surrogate-assisted evolutionary algorithms (SAEAs) are gradually gaining attention as a method for solving expensive optimization problems with inequality constraints. Most SAEAs construct a surrogate model for each objective/constraint function and then aggregate approximation functions of constraints to estimate the feasibility of unevaluated solutions. However, because of the aggregation, the differences in the scales among constraints are ignored. Constraints with smaller scales do not benefit from constraint handling techniques as much as larger constraints, while the effects of handling constraints with larger scales scatter to the other many constraints. This results in an inefficient constraint optimization. Accordingly, this work proposes a new SAEA that partially optimizes each objective/constraint, namely surrogate-assisted partial optimization (SAPO). Solutions with better values of objective/constraint are selected from the evaluated solutions as the parent solutions and a focused objective/constraint is independently optimized using surrogate models one by one. Experimental results reveal the superiority of SAPO compared to the state-of-the-art SAEAs on a single-objective optimization problem suite with inequality constraints under an expensive optimization scenario.

Keywords: Surrogate-assisted Evolutionary Algorithm · Constrained Optimization Problem · Expensive Optimization Problem · Radial Basis Function Network · Differential Evolution.

1 Introduction

Such as wind turbine structure optimization [19] and aerospace and automotive design [17], expensive constrained optimization problems (ECOPs) can often be seen in the real world. In ECOPs, the function evaluations (FEs) are computationally or financially expensive because time-consuming simulations or expensive physical experiments are used as FEs [11]. Hence, finding an optimal and feasible solution within a limited number of FEs is required when solving ECOPs. Expensive single-objective optimization problems with inequality constraints are focused in this work.

The formulation of an ECOP is as follows;

$$\begin{aligned} \text{Minimize} \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & g_m(\boldsymbol{x}) \leq 0, \; m = 1, 2, \ldots, M \;, \\ & x_j^l \leq \boldsymbol{x} \leq x_j^u, \; j = 1, 2, \ldots, D \;, \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a solution, $f : \mathbb{R}^D \to \mathbb{R}$ is the expensive objective function, D is the number of decision variables, $g_m : \mathbb{R}^D \to \mathbb{R}$ is the m-th expensive constraint of M constraints, and x_j^l and x_j^u are the lower and upper values for the j-th decision variable, respectively. The feasibility of a solution is judged by the degree of constraint violation given by;

$$G(\boldsymbol{x}) = \sum_{m=1}^{M} \max\left(g_m(\boldsymbol{x}), 0\right). \tag{2}$$

The solution $\boldsymbol{x}$ is feasible when $G(\boldsymbol{x}) = 0$, otherwise $\boldsymbol{x}$ is infeasible.

Surrogate-assisted evolutionary algorithms (SAEAs) have begun to be applied to ECOPs [10]. SAEAs can save the consumption number of FEs by prescreening candidate solutions to be evaluated using surrogate models of objective/constraints made by machine learning (ML) techniques [22].

Most SAEAs construct surrogate models for the objective and each constraint in Eq. (1) and form a response surface set (RSS), $\left\{\hat{f}(\boldsymbol{x}), \hat{g}_1(\boldsymbol{x}), \hat{g}_2(\boldsymbol{x}), \ldots, \hat{g}_M(\boldsymbol{x})\right\}$ [30]. An RSS can determine the structure of each constraint and capture the characteristics of the feasible region boundaries [28]. In the early study on SAEAs with an RSS, Regis *et al.* [23,20,21] found the radial basis function network (RBFN) [18] can accurately approximate each in an RSS to some extent and work well with various optimizers. To obtain a more accurate RSS, ASAGA [24] and SACOBRA-MQcubic [1] adaptively select ML model types, e.g., RBFN and Kriging [15], and radial basis function (RBF) types in RBFN, respectively. Miranda-Varela and Mezura-Montes conducted an empirical study on constraint handling techniques using the original SA-DECV framework [16] with a k Nearest Neighbor [4,9]-based RSS. SACCDE [31] divides its population into two by the feasibility rule [5] as a constraint handling technique and selects solutions for the mutation strategy of differential evolution (DE) [25] from both groups to generate a variety of offspring solutions. An RSS is used to estimate the feasibility of offspring solutions. GLoSADE [28] constructs global and local surrogate models with RSSs and optimizes them with DE and the interior-point method, respectively. This global-local structure is also adopted in FMSADE [3] and SA-TSDE [14]. FMSADE employs the offspring generation method of SACCDE in the global search while SA-TSDE introduces a surrogate-based repair strategy to obtain a feasible solution using an RSS. MPMLS [12] constructs multiple local surrogate models with different penalty coefficients for the approximation of the degree of constraint violation calculated by an RSS. This mechanism contributes to maintaining a good population diversity [12].

Although these SAEAs create RSSs, prescreening of solutions is performed based on the approximation of the degree of constraint violation $\hat{G}(\boldsymbol{x})$, computed in the manner of Eq. (2) as follows;

$$\hat{G}(\boldsymbol{x}) = \sum_{m=1}^{M} \max\left(\hat{g}_m(\boldsymbol{x}), 0\right). \tag{3}$$

Considering, however, $\hat{G}(\boldsymbol{x})$ is an aggregation of each $\hat{g}_m(\boldsymbol{x})$, the errors between $\hat{g}_m(\boldsymbol{x})$ and its true constraint $g_m(\boldsymbol{x})$ accumulate in $\hat{G}(\boldsymbol{x})$. What is worse, failing to account for the differences in scales between constraints hinders an effective handling of constraints. In other words, relying on only $\hat{G}(\boldsymbol{x})$ to estimate the feasibility of $\boldsymbol{x}$ lacks efficiency and robustness because the approximation accuracy and the effect of constraint handling techniques are highly affected by the scales of $\hat{g}_m(\boldsymbol{x})$ among different m. For example, the improvement of unfulfilled constraints with small scales is prevented by other constraints with large scales because optimization effects of $\hat{G}(\boldsymbol{x})$ less contribute to small-scaled constraints. On the other hand, constraints with larger scales converge slowly as the optimization effect scatters to the other many constraints. One way to tackle this challenge is the normalization of constraints like in SACOBRA [2]. However, even if the normalization is performed, handling only $\hat{G}(\boldsymbol{x})$ works well only when constraints are correlated with each other. Furthermore, in general, obtaining feasible solutions becomes more difficult as the problem dimension D increases [27]. Thereby, optimizing each $\hat{g}_m(\boldsymbol{x})$ with a small number of FEs becomes more important with the increase of D.

Accordingly, this work proposes an SAEA named surrogate-assisted partial optimization (SAPO), which optimizes each objective/constraint in turn. In other words, SAPO partially (independently) optimizes each objective/constraint while putting the other objective/constraints on hold. The RBFN is used to construct an RSS. Unlike existing SAEAs, SAPO prescreens candidate solutions in terms of each approximated constraint $\hat{g}_m(\boldsymbol{x})$ or objective $\hat{f}(\boldsymbol{x})$ in an RSS in place of $\hat{G}(\boldsymbol{x})$. SAPO can obtain solutions specialized for each $\hat{g}_m(\boldsymbol{x})$ or $\hat{f}(\boldsymbol{x})$ and thus each $g_m(\boldsymbol{x})$ or $\hat{f}(\boldsymbol{x})$ can be optimized effectively. To promote the entire optimization of the objective and constraints, SAPO selects solutions with better objective and constraint values to form parent solutions of DE. Thus, the solution diversity is kept high even if SAPO focuses on a certain objective/constraint and improves it. To the best of our knowledge, this is the first attempt to directly utilize the approximated constraint $\hat{g}_m(\boldsymbol{x})$ as the criterion of solution prescreening. The partial optimization of the objective and constraints can be a new constraint handling technique. This work empirically demonstrates that this new technique improves the performance of SAEAs on ECOPs because more feasible solutions are found with fewer FEs.

The remainder of this work is organized as follows. Section 2 introduces DE and RBFN as the component techniques of SAPO. Section 3 provides the design concept and the detailed algorithm of SAPO. Section 4 compares the performance of SAPO with state-of-the-art SAEAs on the CEC 2017 constrained real-parameter optimization benchmark suite [29] within an expensive scenario. In Section 5, we show the effectiveness of our partial optimization as an ablation study. Lastly, Section 6 concludes our work and discusses future directions.

2 Preliminary

This section introduces DE as the optimizer of the objective and constraints. Successively, RBFN as an ML technique for surrogate models is explained.

2.1 DE: Differential Evolution

DE is an evolutionary algorithm originally proposed for real-parameter bound-constrained single-objective optimization problems. The optimization process of DE consists of initialization, mutation, crossover, and solution selection, where procedures from mutation to solution selection are repeated till the termination.

DE initializes its population $\mathcal{P} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, where N is the population size. Specifically, DE samples each solution $\boldsymbol{x}_i = [x_{i,1}, \ldots, x_{i,D}]^\top$ using a uniform distribution within $x_{i,j} \in [x_j^l, x_j^u]$. The definition of x_j^l and x_j^u follows Eq. (1).

In the mutation procedure, a mutant solution $\boldsymbol{v}_i = [v_{i,1}, \cdots, v_{i,D}]^\top$ of each $\boldsymbol{x}_i$, i.e., a parent solution, is produced using an employed mutation strategy. To generate a variety of solutions, this work adopts two mutation strategies. The *rand/1* and *best/1* strategies contribute to exploration and exploitation, respectively. Definitions of these strategies are as follows;

$$\textit{rand/1}: \quad \boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}), \tag{4}$$

$$\textit{best/1}: \quad \boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}), \tag{5}$$

where $\boldsymbol{x}_{r_1}$, $\boldsymbol{x}_{r_2}$, and $\boldsymbol{x}_{r_3}$ are mutually exclusive solutions randomly selected from $\mathcal{P}$, also different from $\boldsymbol{x}_i$. The best solution $\boldsymbol{x}_{best}$ is a solution in $\mathcal{P}$ having the best fitness value. A scaling factor $F \in [0, 1]$ controls the contribution of differential vectors $(\boldsymbol{x}_r - \boldsymbol{x}_{r'})$.

Next, DE generates a trial vector $\boldsymbol{u}_i = [u_{i,1}, \cdots, u_{i,D}]^\top$ as an offspring solution via a crossover strategy and crossover rate $CR \in [0, 1]$. We use the most popular *binomial* crossover strategy given by;

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{6}$$

where j_{rand} is an integer randomly selected from $[1, D]$ and $rand(0, 1)$ is a uniformly sampled real random value in $(0, 1)$.

Finally, $\boldsymbol{u}_i$ is evaluated and the solution for the next generation is selected, i.e., $\boldsymbol{x}_i$ is replaced with $\boldsymbol{u}_i$ if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ for minimization problems.

2.2 RBFN: Radial Basis Function Network

RBFN is a feed-forward neural network with a three-layer structure. We employ the RBFN as an ML technique for the surrogate model because its construction or prediction time is relatively short and the model accuracy is scalable to the increase of problem dimension [7]. Let a training dataset (size n), vector of function values, and an RBF be $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{n}$, $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)]^\top$, and $\phi(r)$, respectively. The *cubic* RBF $\phi(r) = r^3$ is used as it can establish a fitness landscape that is more stable and exhibits improved convergence, thanks to its ability to avoid ill-conditioning [26]. Note that RBFN can also be used to approximate constraints by replacing $f(\boldsymbol{x})$ with $g_m(\boldsymbol{x})$.

The approximation of $f(\boldsymbol{x})$ for $\boldsymbol{x}$ can be formulated as follows;

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right) + p(\boldsymbol{x}), \tag{7}$$

where $p(\boldsymbol{x}) = \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} + c_0$ ($\boldsymbol{c} \in \mathbb{R}^D, c_0 \in \mathbb{R}$) is regularization terms using a linear polynomial function, and $\lambda_i \in \mathbb{R}$ is the i-th element of the weight vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_n]^{\mathsf{T}}$.

Three parameters $\boldsymbol{\lambda}$, $\boldsymbol{c}$, and c_0 in Eq. (7) are obtained by solving the following equation;

$$\begin{bmatrix} \mathit{\Phi} & P \\ P^{\mathsf{T}} & 0_{\mathrm{m}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c}' \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}, \tag{8}$$

where $\mathit{\Phi}$ is the $n \times n$ matrix with $\phi_{ij} = \phi\left(|\boldsymbol{x}_i - \boldsymbol{x}_j|\right)$, $P \in \mathbb{R}^{n\times(D+1)}$ is a matrix whose i-th row is $\left[1,\ \boldsymbol{x}_i^{\mathsf{T}}\right]$, 0_{m} is a $(D{+}1)\times(D{+}1)$ matrix of zeros, $\boldsymbol{c}' = \left[\boldsymbol{c}^{\mathsf{T}},\ c_0\right]^{\mathsf{T}}$, and $\boldsymbol{0}$ is a column vector of zeros whose length is $(D + 1)$.

3 SAPO: Surrogate-assisted Partial Optimization

This section starts by sharing the concept of SAPO and then explicates its algorithm.

3.1 Concept

Different from existing SAEAs for ECOPs where all constraints are dealt with in one bundle as the approximation of the degree of constraint violation, SAPO addresses an objective/constraint one by one, namely a partial optimization. This idea is inspired by the partial differential equation (PDE). Focusing on one element in PDE improves the efficiency of structure analysis or optimization [8,13]. That is, complex systems must be disassembled, and processed one by one.

Going back to ECOPs, the partial optimization has advantages below;

- Solutions suitable to an objective/constraint function can be screened by surrogate models. Compared to only using the approximation of the degree of constraint violation, this mechanism has a clearer intent to improve each function. Thus, the optimization efficiency for each function is enhanced.
- Solution diversity is kept high until the end of the search because SAPO has multiple criteria to prescreen offspring solutions.
- The partial optimization can handle the case where the scales among constraints are highly different. Unsatisfied constraints with relatively small scales are also focused while they are likely to be ignored when only the degree of constraint violation is used. Constraints with relatively large scales are also improved with fewer FEs because they are exclusively considered.

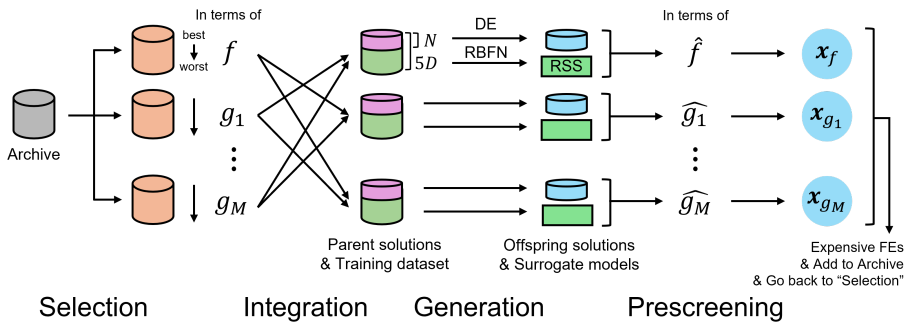

Fig. 1. A diagram of SAPO.

Algorithm 1 SAPO

```
1:  Initialize the archive $\mathcal{A} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i), \{g_m(\boldsymbol{x})\}_{m=1}^{M})\}_{i=1}^{N_{init}}$, $FE = N_{init}$
2:  while $FE < FE_{\max}$ do
3:    for all Target Function $\leftarrow \{f, g_1, g_2, \ldots, g_M\}$ do
4:      for all Selection Criterion $\leftarrow \{f, g_1, g_2, \ldots, g_M\} \setminus$ Target Function do
5:        $\mathcal{S} \leftarrow$ Algorithm 2($\mathcal{A}$, Selection Criterion, Target Function)   // Selection
6:      end for
7:      $\mathcal{P}, \mathcal{D} \leftarrow N, 5D$ solutions from $\mathcal{S}$, respectively   // Integration
8:      $\mathcal{U} \leftarrow$ Generate offspring from $\mathcal{P}$ with DE Eqs. (4-6)   // Generation
9:      $\mathcal{R} \leftarrow$ Generate RSS using $\mathcal{D}$ with RBFN Eq. (7)   // Generation
10:     $\boldsymbol{x}^* \leftarrow$ Algorithm 3($\mathcal{U}$, $\mathcal{R}$, Target Function)   // Prescreening
11:     Evaluate $\boldsymbol{x}^*$ and add it to $\mathcal{A}$, $FE = FE + 1$
12:   end for
13: end while
```

To fully utilize the partial optimization idea and the solution diversity mentioned above, parent solutions are selected from top solutions in terms of the objective/constraints that are not focused on now. Thus, SAPO can generate and prescreen offspring solutions that improve a focused objective/constraint from parent solutions with good values of the other objective/constraints.

3.2 Mechanism

Algorithm 1 provides the complete pseudocode of SAPO. The algorithm consists of three procedures; 1) Initialization, 2) Selection and Integration of evaluated solutions to obtain parent solutions and training dataset, and 3) Generation of offspring solutions and an RSS and Prescreening of offspring solutions. After initialization, SAPO repeats 2) and 3) until the termination criteria are met. Fig. 1 shows the diagram of 2) and 3). For each generation, SAPO sets a target objective/constraint to be optimized and this sequentially changes to the next one, i.e., in the order of $f, g_1, g_2, \ldots, g_M, f, g_1, \ldots$ and so on.

Algorithm 2 Selection

```
Input: Archive $\mathcal{A}$, Objective/constraint as the selection criterion $f$ or $g_{m'}$,
       Target objective/constraint to be optimized $f$ or $g_m$
Output: Selected and sorted set $\mathcal{S}$
 1: if $f$ is the target to be optimized then
 2:     $\mathcal{T}_1 \leftarrow$ Sort feasible solutions in $\mathcal{A}$ in ascending order of $f$
 3:     $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \land g_{m'}(\boldsymbol{x}) \leq 0 \land f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_{m'}$
 4:     $\mathcal{T}_3 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \land g_{m'}(\boldsymbol{x}) \leq 0 \land f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of $f$
 5:     $\mathcal{T}_4 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \land g_{m'}(\boldsymbol{x}) > 0 \land f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_{m'}$
 6:     $\mathcal{T}_5 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \land g_{m'}(\boldsymbol{x}) > 0 \land f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of $f$
 7: else if $g_m$ is the target to be optimized then
 8:     $\mathcal{G} \leftarrow \{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \land g_m(\boldsymbol{x}) > 0\}$    // $\boldsymbol{x}$ that satisfy $g_m(\boldsymbol{x})$ need not be optimized
 9:     if $f$ is the selection criterion then
10:         $\mathcal{T}_1 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \land f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_m$
11:         $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \land f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of $f$
12:     else if $g_{m'}$ is the selection criterion then
13:         $\mathcal{T}_1 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \land g_{m'}(\boldsymbol{x}) \leq 0 \land f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_m$
14:         $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \land g_{m'}(\boldsymbol{x}) \leq 0 \land f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of $f$
15:         $\mathcal{T}_3 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \land g_{m'}(\boldsymbol{x}) > 0\}$ in ascending order of $g_{m'}$
16:     end if
17: end if
18: $\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{last}]$
```

Initialization Latin hypercube sampling is employed to sample N_{init} points. These points are evaluated with the objective and constraint functions. SAPO creates an archive $\mathcal{A} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i), \{g_m(\boldsymbol{x}_i)\}_{m=1}^{M})\}_{i=1}^{N_{init}}$ to store all evaluated solutions and their objective/constraint values.

Selection and Integration This phase intends to extract good solutions from the archive to prepare parent solutions and a training dataset for an RSS. Here, we define f^*_{fea} as the best fitness value among feasible solutions in $\mathcal{A}$. If there are no feasible solutions in $\mathcal{A}$, f^*_{fea} is set to ∞. Solutions satisfying $f(\boldsymbol{x}) < f^*_{fea}$ are infeasible but may be useful in the optimization if their constraints are improved. Thus these solutions are utilized below.

First, SAPO makes M sets of selected and sorted solutions, where M is the number of constraints. Each set is generated corresponding to each objective/constraint except for the target objective/constraint. Note that each solution in $\mathcal{A}$ can be selected multiple times. The selecting and sorting method is summarized in Algorithm 2.

– When f is the target to be optimized, SAPO selects and sorts solutions for each $g_{m'}$ as Lines 1-6 of Algorithm 2. Let $g_{m'}$ be a focused constraint as the selection criterion in this phase. First, feasible solutions are selected from $\mathcal{A}$ and sorted in ascending order of f and form $\mathcal{T}_1$. Next, $\boldsymbol{x}$s that satisfy $g_{m'}$ are corrected. Among these solutions, ones that give $f(\boldsymbol{x}) < f^*_{fea}$ are preferred as mentioned before. As $g_{m'}$ is the criterion now, they are sorted with the

Algorithm 3 Prescreening

```
Input: Offspring solutions $\mathcal{U} = \{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{U}|}$, RSS $\mathcal{R}$, Target objective/constraint $f$ or $g_m$
Output: Selected solution $\boldsymbol{x}^*$
1: if $f$ is the target then
2:     Calculate $\hat{G}(\boldsymbol{x})$ values of $\forall \boldsymbol{x} \in \mathcal{U}$ by Eq. (3) using $\mathcal{R}$
3:     $\mathcal{F} \leftarrow \{\boldsymbol{x} |\ \boldsymbol{x} \in \mathcal{U} \land \hat{G}(\boldsymbol{x}) = 0\}$        // $\boldsymbol{x}$ estimated to be feasible
4:     $\boldsymbol{x}^* = \begin{cases} \arg\min_{\boldsymbol{x}\in\mathcal{F}} \hat{f}(\boldsymbol{x}), & \mathcal{F} \neq \emptyset \\ \arg\min_{\boldsymbol{x}\in\mathcal{U}} \hat{G}(\boldsymbol{x}), & \text{otherwise} \end{cases}$
5: else if $g_m$ is the target then
6:     $\mathcal{G} \leftarrow \{\boldsymbol{x} |\ \boldsymbol{x} \in \mathcal{U} \land \hat{g}_m(\boldsymbol{x}) \leq 0\}$        // $\boldsymbol{x}$ that satisfy $\hat{g}_m(\boldsymbol{x})$
7:     $\boldsymbol{x}^* = \begin{cases} \arg\min_{\boldsymbol{x}\in\mathcal{G}} \hat{f}(\boldsymbol{x}), & \mathcal{G} \neq \emptyset \\ \arg\min_{\boldsymbol{x}\in\mathcal{U}} \hat{g}_m(\boldsymbol{x}), & \text{otherwise} \end{cases}$
8: end if
```

$g_{m'}$ values, forming $\mathcal{T}_2$. Then, $\boldsymbol{x}$s that satisfy $g_{m'}$ but give $f(\boldsymbol{x}) \geq f^*_{fea}$ are used. As they are inferior to $\boldsymbol{x} \in \mathcal{T}_2$, the sorting order of $\mathcal{T}_3$ follows f values. Similar procedures are performed to make $\mathcal{T}_4$ and $\mathcal{T}_5$ for $\boldsymbol{x}$s that violate $g_{m'}$. Finally, SAPO returns a set $\mathcal{S}$ by joining $\mathcal{T}_1, \mathcal{T}_2, \dots,$ and $\mathcal{T}_5$ in this order.

– When g_m is the target to be optimized, solutions that already satisfy g_m are ignored as they need not be optimized anymore, as shown in Line 8. Solutions that violate g_m are selected from $\mathcal{A}$ and consist of $\mathcal{G}$. The remained procedures are as Lines 9-15. If f is the selection criterion, solutions in $\mathcal{G}$ and $f(\boldsymbol{x}) < f^*_{fea}$ are selected to utilize the possibility of improvement, forming $\mathcal{T}_1$. However, sorting order follows the ascending order of the target constraint g_m as solutions in $\mathcal{T}_1$ lack feasibility. Next, solutions with $f(\boldsymbol{x}) \geq f^*_{fea}$ form $\mathcal{T}_2$ where solutions are ordered with f values. An output set is $\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2]$. On the other hand, $g_{m'}$ is taken into account when $g_{m'}$ is the selection criterion. The fulfillment of $g_{m'}$ has the first priority. However, as $g_{m'} \leq 0$ is enough, the next priority becomes comparison with f^*_{fea}. Similarly, $\mathcal{T}_1$ and $\mathcal{T}_2$ are obtained. Finally, solutions that violate $g_{m'}$ are selected and sorted In increasing order of the degree of violation of $g_{m'}$, forming $\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]$.

After gaining M sets of sorted solutions, they are integrated as follows. The elements of each set are taken in order from the first one, let this be the parent solution set $\mathcal{P}$ when there are N elements, and let this be the training dataset $\mathcal{D}$ for constructing an RSS when its size becomes $5D$.

Generation and Prescreening This phase begins by generating offspring solutions and constructing an RSS. For the evolution of $\mathcal{P}$, the mutation strategies shown in Eqs. (4-5) and the crossover strategy in Eq. (6) are applied to $\mathcal{P}$. Note that the size of the offspring solution set $\mathcal{U}$ becomes double that of $\mathcal{P}$, i.e., $2N$. To generate a variety of offspring solutions, $\mathcal{P}$ is copied, and two mutation strategies are independently used for each $\mathcal{P}$. Successively, an RSS is constructed using RBFNs and $\mathcal{D}$ as presented in Eq. (7).

Finally, SAPO prescreens $\mathcal{U}$ in terms of the target objective/constraint. The detailed procedures are indicated in Algorithm 3. When f is the target, the feasibility rule is employed. Specifically, $\hat{G}(\boldsymbol{x})$ calculated by Eq. (3) using an RSS estimates the feasibility of solutions in $\mathcal{U}$. If there are expected to be feasible solutions, the solution having the minimum $\hat{f}(\boldsymbol{x})$ is selected. Otherwise, the solution having the minimum $\hat{G}(\boldsymbol{x})$ is chosen. When g_m is the target, however, solutions that satisfy $\hat{g}_m(\boldsymbol{x})$ are esteemed. Among them, $\boldsymbol{x}$ having the minimum $\hat{f}(\boldsymbol{x})$ is selected, denoting $\boldsymbol{x}^*$. If no solution fulfills $\hat{g}_m(\boldsymbol{x})$, the solution with the least violation of $\hat{g}_m(\boldsymbol{x})$ is selected. The selected solution is evaluated with the expensive function, and it and its objective/constraint values are added to $\mathcal{A}$.

4 Experiment

Through a comparison of performances between SAPO and SAEAs for ECOPs, we evaluate the effectiveness of SAPO.

4.1 Experimental Design

We use the IEEE CEC 2017 constrained real-parameter optimization benchmark suite [29]. Nine problems with inequality constraints are selected from the suite, i.e., F1, F2, F4, F5, F12, F13, F20, F21, and F22. Note that F19 and F28 are excluded although they are problems with inequality constraints as they have no feasible solutions according to the problem definition [29]. The problem dimensions are set to $D \in \{30, 50, 100\}$ to evaluate the scalability of the performance of SAPO against the increase of D. The other settings follow the regulation of competition [29]. All experiments are done with Intel(R) Core(TM) i7-10700 (2.90 GHz) CPU and 16 GB RAM.

Four state-of-the-art SAEAs, GLoSADE [28], FMSADE [3], MPMLS [12], and SA-TSDE [14], are employed for the compared algorithms. All algorithms use DE and RBFN, so we can fairly compare the performances. Note that GLoSADE, FMSADE, and SA-TSDE adopt the interior point algorithm for the local search. While these three SAEAs utilize global and local searches, MPMLS decomposes the approximation of the degree of constraint violation $\hat{G}(\boldsymbol{x})$ using penalty coefficients. MPMLS and SAPO construct only local surrogate models. Consequently, this work can investigate the impact of differences among global/local structures and the deals of approximated constraints on the performance. Hyperparameter settings of the compared algorithms follow the original papers [28,3,12,14]. For SAPO, we set to $N_{init} = 100$ for $D \in \{30, 50\}$ and 200 for $D = 100$ so that $N_{init} > D$. We also use $N = 100$, $F = 0.5$, and $CR = 0.9$, following the original paper of DE [25], and $kernel = cubic$, one of the most popular RBFN kernels.

Following the original papers [28,3,12,14], the maximum number of FEs is set to $3,000$. The performance is evaluated with the average fitness values of feasible solutions obtained in 31 independent runs for each problem and average ranks over nine problems. We apply the Wilcoxon rank-sum test with a significance level of 0.05 to check statistical significance. When reporting statistical results, we use "+", "−", or "∼", indicating the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot decide that there is a significant difference, respectively. In case no feasible solution is obtained within the observed number of FEs, a certain enough large fitness value (1E+20) is assigned to the corresponding runs when computing statistical test results and average ranks.

4.2 Result

Table 1 summarizes the average fitness values obtained at $3,000$ FEs for $D \in \{30, 50, 100\}$. SAPO derived six, six, and four best performances out of nine problems in the order of $D = 30$, 50, and 100 and no worst performance. This shows the robustness of SAPO over different problems, except for those on which all algorithms failed in finding feasible solutions as shown in gray. This means the partial optimization methodology proposed in this work contributed to steadily improving the objective value while satisfying constraints on many types of problems. The average ranks shown at the bottom of Table 1 also demonstrate the superiority of SAPO. Although the average rank slightly degrades as D increases, each average rank is the best among the five algorithms for all D. Thus, the performance of SAPO scales to the increase of problem dimension.

From a statistical point of view, the number of "+" indicating the superiority of the compared algorithms is zero or one in all comparison pairs for all D. The number of "−" indicating the inferiority of the compared algorithms is four and seven at least and at most, respectively. Accordingly, the number of "−" is much larger than that of "+" in all comparisons, where the maximum difference is seven out of nine problems. The total results of 108 comparisons across the three types of dimensions and the four compared algorithms are $+/-/\sim = 3/67/38$. These results clearly indicate the outstanding performance of SAPO.

It is worth noting that SAPO derived a much larger number of successful runs. For example, SAPO succeeded in finding feasible solutions in all runs on F12 ($D \in \{30, 50\}$) and F21 ($D = 30$) although some or all of the other algorithms failed. Even if D increased and the difficulty of finding feasible solutions became higher, SAPO obtained multiple successful runs on the same problems, i.e., F12 ($D = 100$) and F21 ($D = 50$), while the other algorithms could not succeed in any run. MPMLS is the best algorithm except SAPO in terms of finding feasible solutions with better objective values, e.g., on F12 and F20. This may be because MPMLS employs a decomposition strategy of $\hat{G}(\boldsymbol{x})$, unlike other SAEAs. However, SAPO contributed to obtaining more feasible solutions with better objective value on problems not only with one constraint but also with multiple constraints. These results demonstrate the effectiveness of the partial optimization methodology of SAPO, where each objective and constraint were optimized directly and thus improved effectively. This methodology was also useful for non-separable and rotated constraints like F2 and F5 as SAPO derived

Table 1. The average fitness values at $3,000$ function evaluations for $D \in \{30, 50, 100\}$. The integers next to the problem name in brackets indicate the number of constraints. The best and worst results among the five algorithms are highlighted in green bold and pink italic, respectively. When some of the 31 runs failed in obtaining feasible solutions, the number of successful runs, i.e., cases where an algorithm found at least one feasible solution, is noted in brackets. When all algorithms could not find feasible solutions, corresponding cells are highlighted in gray. Statistical test result "+", "−", or "∼" indicates the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot see a significant difference, respectively.

a) $D = 30$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	9.557e+03 −	*4.164e+04* −	2.499e+04 −	7.389e+03 −	**5.400e+03**
F2 (1)	4.032e+03 −	*(1)* −	5.468e+03 −	3.070e+03 −	**2.110e+03**
F4 (2)	3.883e+02 −	*4.060e+02* −	1.973e+02 ∼	2.148e+02 −	**1.888e+02**
F5 (2)	4.178e+01 −	*3.183e+02* −	**2.895e+01** ∼	5.707e+01 −	3.251e+01
F12 (2)	1.517e+02 −	*(0)* −	1.559e+01 −	(30) −	**1.261e+01**
F13 (3)	*(0)* −	*(0)* −	*(0)* −	**(7)** ∼	**(7)**
F20 (2)	9.801e+00 ∼	*1.002e+01* ∼	**9.589e+00** +	9.920e+00 ∼	9.877e+00
F21 (2)	*(0)* −	*(0)* −	(29) −	*(0)* −	**1.075e+01**
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	0/**7**/2	0/**7**/2	1/**5**/3	0/**6**/3	-
Ave. Rank	3.222	*4.556*	2.556	3.056	**1.611**

b) $D = 50$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	3.038e+04 −	*1.064e+05* −	6.341e+04 −	3.852e+04 −	**2.547e+04**
F2 (1)	1.707e+04 −	*(0)* −	2.087e+04 −	1.487e+04 −	**1.056e+04**
F4 (2)	*6.819e+02* −	6.686e+02 −	4.574e+02 −	3.507e+02 −	**3.100e+02**
F5 (2)	2.444e+03 −	*(27)* −	1.210e+02 −	1.879e+03 −	**9.113e+01**
F12 (2)	*(0)* −	*(0)* −	1.052e+02 −	*(0)* −	**1.429e+01**
F13 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F20 (2)	1.824e+01 ∼	*1.882e+01* ∼	**1.803e+01** ∼	1.860e+01 ∼	1.838e+01
F21 (2)	*(0)* −	*(0)* −	*(0)* −	*(0)* −	**(30)**
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	0/**6**/3	0/**6**/3	0/**6**/3	0/**6**/3	-
Ave. Rank	3.278	*4.167*	2.833	3.056	**1.667**

c) $D = 100$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	1.644e+05 ∼	*4.964e+05* −	2.352e+05 −	**1.597e+05** ∼	1.624e+05
F2 (1)	1.314e+05 −	*(0)* −	9.185e+04 −	9.066e+04 −	**7.763e+04**
F4 (2)	1.491e+03 −	*1.523e+03* −	1.257e+03 −	8.659e+02 −	**7.750e+02**
F5 (2)	4.271e+04 −	*(25)* −	2.094e+03 −	2.385e+04 −	**1.384e+03**
F12 (2)	*(0)* −	*(0)* −	*(0)* −	*(0)* −	**(19)**
F13 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F20 (2)	4.007e+01 +	4.067e+01 ∼	**3.969e+01** +	*4.113e+01* ∼	4.093e+01
F21 (2)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	1/**4**/4	0/**5**/4	1/**5**/3	0/**4**/5	-
Ave. Rank	3.278	*3.944*	2.833	2.833	**2.111**

good performances on them. Even if decision variables cannot be separated, the objective and constraints can be optimized independently.

Table 2. Significant differences regarding findings for "+/ − / ∼" between SAPO and state-of-the-art SAEAs. Statistical test result "+", "−", or "∼" indicates the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot see a significant difference, respectively. In the comparisons between the numbers of "+" and "−", the larger numbers are highlighted in bold.

D	FE	GLoSADE	FMSADE	MPMLS	SA-TSDE	D	FE	GLoSADE	FMSADE	MPMLS	SA-TSDE
	300	0/**3**/6	0/**4**/5	0/**3**/6	**1**/**1**/7		300	0/ **2**/7	0/ **3**/6	0/ **1**/ 8	**2**/ **2**/ 5
	500	0/**5**/4	0/**5**/4	0/**3**/6	**2**/1/6		500	1/ **2**/6	0/ **3**/6	1/ **2**/ 6	**2**/ **2**/ 5
30	1,000	0/**6**/3	0/**6**/3	0/**5**/4	1/**3**/5	100	1,000	1/ **3**/5	0/ **4**/5	1/ **2**/ 6	**2**/ **2**/ 5
	2,000	0/**7**/2	0/**7**/2	1/**7**/1	0/**5**/4		2,000	0/ **3**/6	0/ **4**/5	1/ **3**/ 5	1/ **2**/ 6
	3,000	0/**7**/2	0/**7**/2	1/**5**/3	0/**6**/3		3,000	1/ **4**/4	0/ **5**/4	1/ **5**/ 3	0/ **4**/ 5
	300	0/**3**/6	1/**3**/5	0/**3**/6	**2**/**2**/5		300	0/ **8**/19	1/**10**/16	0/ **7**/20	**5**/ **5**/17
	500	0/**3**/6	0/**4**/5	0/**3**/6	**2**/1/6		500	1/**10**/16	0/**12**/15	1/ **8**/18	**6**/ 4/17
50	1,000	0/**5**/4	0/**5**/4	0/**5**/4	**2**/**2**/5	Total	1,000	1/**14**/12	0/**15**/12	1/**12**/14	5/ **7**/15
	2,000	0/**6**/3	0/**7**/2	0/**6**/3	0/**5**/4		2,000	0/**16**/11	0/**18**/ 9	2/**16**/ 9	1/**12**/14
	3,000	0/**6**/3	0/**6**/3	0/**6**/3	0/**6**/3		3,000	1/**17**/ 9	0/**18**/ 9	2/**16**/ 9	0/**16**/11

Furthermore, we show the results of Wilcoxon rank-sum tests at 300, 500, $1,000$, and $2,000$ FEs in addition to $3,000$ (baseline) FEs for $D \in \{30, 50, 100\}$ and their total number in Table 2 to evaluate the convergence performance of SAPO. SAPO outperformed GLoSADE, FMSADE, and MPMLS on every number of FEs in the table, where the number of "+" indicating the superiority of the compared SAEAs is at most one. Although SAPO was competitive with SA-TSDE before $1,000$ FEs, "−" outnumbers "+" after $2,000$ FEs, indicating that SAPO kept improving the objective/constraint towards $3,000$ FEs. This tendency can be observed in every D and their total. Therefore, we can confirm the scalability of the performance of SAPO to the increase in the number of FEs.

5 Discussion

5.1 Impact of the Partial Optimization

Unlike existing SAEAs where constraints are treated as only an aggregation of constraints, SAPO partially optimizes each objective/constraint. This subsection investigates the effectiveness of the partial optimization proposed in this work. Here, we prepared three variants of SAPO.

1. **Variant using Aggregation (VUA)** This variant is the most similar to the mechanism of existing SAEAs, which employs the feasibility rule [5] using aggregation of constraints, instead of partial optimization. The feasibility rule is the representative constraint handling technique [12]. Specifically, this variant selects and sorts solutions for parent solutions and the training dataset of an RSS by Eq. (2) every time. Feasible solutions are preferentially selected and sorted in ascending order of $f(\boldsymbol{x})$. Then, infeasible solutions are sorted in ascending order of $G(\boldsymbol{x})$. DE offspring solutions and an RSS are generated in the same manner as those of Section 3.2. Offspring solutions are prescreened as Lines 2-4 in Algorithm 3. Comparison with this variant reveals the impact of partial optimization.

2. **Variant Targeting Objective (VTO)** This variant sets only the objective function as the target. Thus, solutions for DE parent solutions and the training dataset are selected and sorted from good constraints. Specifically, only Lines 1-6 and Lines 1-4 are conducted in Algorithm 2 and Algorithm 3, respectively.
3. **Variant Targeting Constraints (VTC)** This variant sets only the constraints as the target. Hence, M constraints are partially optimized, but solutions are not screened to improve $f(\boldsymbol{x})$. Only Lines 9-15 and Lines 5-7 are conducted in Algorithm 2 and Algorithm 3, respectively. VTO and VTC are prepared to evaluate whether both the objective and constraints are needed or not as the targets of the partial optimization.

The experimental design is similar to Section 4.1. Table 3 summarizes the results of the Wilcoxon rank-sum test. Statistical sign "+", "−", or "∼" indicates the variant significantly outperformed SAPO, significantly underperformed SAPO, or we cannot say that there is a significant difference, respectively. In comparison with VUA, SAPO outperformed VUA at $D = 30$. Specifically, the number of "+", indicating the superiority of VUA, is larger than that of "−", indicating the inferiority of VUA, with 300 FEs. As the feasibility rule devotes many resources to feasible solutions [12], the objective value is improved in the very early phase of the search once feasible solutions are obtained. However, in problems where obtaining feasible solutions is difficult, the use of the constraint aggregation $G(\boldsymbol{x})$ or its approximation $\hat{G}(\boldsymbol{x})$ requires more FEs to improve constraint violation in the feasibility rule. Thus, the performance of VUA stagnated and the number of "−" outnumbers that of "+" after $1,000$ FEs. The same tendency is observed for $D \in \{50, 100\}$. Although the difference in the number of "−" and "+" decreases as D increases, SAPO outperformed VUA in total of all D. Thus, the effectiveness of the partial optimization is confirmed.

In the comparison between SAPO and VTO, the performance of SAPO is slightly better than that of VTO. A similar trend to the comparison between SAPO and VUA is detected; the performance of VTO stagnated. Thus, partial optimization of constraints is needed to keep improving the performance. On the other hand, SAPO clearly outperformed VTC for all D. This indicates the improvement of the objective values is necessary although the partial optimization of constraints contributes to the improvement of each constraint. From these two comparisons, we identified that both the objective and constraints should be dealt with as the targets of the partial optimization.

5.2 Impact of the Parallel Use of DE Mutation Strategies

SAPO uses both *rand/1* and *best/1* mutation strategies to produce a variety of offspring solutions. This subsection evaluates the effectiveness of the parallel use of two DE mutation strategies. We prepared two variants of SAPO; one uses only *rand/1* and the other uses only *best/1*. Note that these variants generate $2N$ offspring for fair comparison. Again, the experimental design is similar to Section 4.1. Table 4 summarizes the results of the Wilcoxon rank-sum test. Statistical test sign "+", "−", or "∼" denotes the variant significantly outperformed

Table 3. Significant differences regarding findings for "+/ − / ∼" between SAPO and its variants for an ablation study on partial optimization.

	$D = 30$			$D = 50$			$D = 100$			Total		
FE	VUA	VTO	VTC	VUA	VTO	VTC	VUA	VTO	VTC	VUA	VTO	VTC
300	**3**/1/5	**3**/0/6	1/**4**/4	**2**/0/7	**3**/0/6	0/**3**/6	**2**/0/7	**4**/0/5	0/**4**/5	**7**/ 1/19	**10**/0/17	1/**11**/15
500	1/**3**/5	**3**/0/6	0/**8**/1	**5**/0/4	**5**/0/4	1/**4**/4	**4**/0/5	**4**/0/5	0/**4**/5	**10**/ 3/14	**12**/0/15	1/**16**/10
1,000	0/**5**/4	0/**5**/4	0/**8**/1	**4**/0/5	**4**/0/5	0/**6**/3	**6**/0/3	**4**/0/5	0/**6**/3	**10**/ 5/12	**8**/5/14	0/**20**/ 7
2,000	0/**5**/4	0/**4**/5	0/**8**/1	**2**/1/6	**2**/**2**/5	0/**7**/2	**2**/1/6	**2**/1/6	0/**7**/2	4/ **7**/16	4/**7**/16	0/**22**/ 5
3,000	0/**5**/4	0/**3**/6	0/**8**/1	0/**3**/6	0/**3**/6	0/**6**/3	1/**3**/5	1/**2**/6	0/**6**/3	1/**11**/15	1/**8**/18	0/**20**/ 7

Table 4. Significant differences regarding findings for "+/ − / ∼" between SAPO and its variants for an ablation study on the DE mutation strategies.

	$D = 30$		$D = 50$		$D = 100$		Total	
FE	*rand/1*	*best/1*	*rand/1*	*best/1*	*rand/1*	*best/1*	*rand/1*	*best/1*
300	0/**4**/5	0/**1**/8	0/**3**/6	**2**/0/7	0/**2**/7	**1**/0/8	0/ **9**/18	**3**/1/23
500	0/**5**/4	0/**2**/7	0/**3**/6	**2**/1/6	0/**3**/6	**0**/**0**/9	0/**11**/16	**2**/3/22
1,000	0/**6**/3	**1**/**1**/7	0/**5**/4	**2**/1/6	0/**4**/5	0/**2**/7	0/**15**/12	**3**/4/20
2,000	0/**7**/2	0/**1**/8	0/**6**/3	**1**/**1**/7	0/**4**/5	0/**2**/7	0/**17**/10	1/**4**/22
3,000	0/**6**/3	0/**3**/6	0/**6**/3	1/**3**/5	0/**5**/4	1/**3**/5	0/**17**/10	2/**9**/16

SAPO, significantly underperformed SAPO, or we cannot determine that there is a significant difference, respectively. From the table, SAPO outperformed the variant with *rand/1* as no "+". indicating the superiority of the variant, are observed. The single-use of *rand/1* lacks the exploitation ability. The variant with *best/1* derived slightly better performance than SAPO in the early stage of optimization while SAPO became slightly better at the end of the search. This indicates the strong exploitation ability of *best/1* but the diversity of offspring solution should be maintained by adding *rand/1*.

6 Conclusion

This work proposed an SAEA named surrogate-assisted partial optimization (SAPO). SAPO selects and sorts solutions with good objective/constraint values to form parent solutions and then independently optimizes each objective/constraint one by one. In the experiment, SAPO derived significantly better performance than existing SAEAs as SAPO dealt with each constraint effectively. In the discussion, we showed our partial optimization methodology can find feasible solutions with better objective values within a smaller number of FEs than using only an approximation of constraint violation made by aggregation of constraint approximations, which is commonly used in existing SAEAs.

Future work includes an adaptive selection of the objective/constraint to be optimized to improve the optimization efficiency. We are motivated to solve the entire problem set of the CEC 2017 benchmark suite, including problems with equality constraints. We will also extend SAPO for multi-objective ECOPs, where surrogate models are constructed for independent objectives/constraints or an aggregated function of all objectives/constraints [6].

Acknowledgments. This work was supported by JSPS KAKENHI under Grant No. 22KJ1409.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bagheri, S., Konen, W., Bäck, T.: Online selection of surrogate models for constrained black-box optimization. In: IEEE Symp. Ser. Comput. Intell. (SSCI). pp. 1–8. ieeexplore.ieee.org (Dec 2016). https://doi.org/10.1109/SSCI.2016.7850206
2. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Appl. Soft Comput. **61**, 377–393 (Dec 2017). https://doi.org/10.1016/j.asoc.2017.07.060
3. Chu, S., Yang, Z., Xiao, M., Qiu, H., Gao, K., Gao, L.: Explicit topology optimization of novel polyline-based core sandwich structures using surrogate-assisted evolutionary algorithm. Comput. Methods Appl. Mech. Eng. **369**, 113215 (Sep 2020). https://doi.org/10.1016/j.cma.2020.113215
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (Jan 1967). https://doi.org/10.1109/TIT.1967.1053964
5. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2), 311–338 (Jun 2000). https://doi.org/10.1016/S0045-7825(99)00389-8
6. Deb, K., Roy, P.C., Hussein, R.: Surrogate Modeling Approaches for Multiobjective Optimization: Methods, Taxonomy, and Results. Math. Comput. Appl. **26**(1), 5 (Dec 2020). https://doi.org/10.3390/mca26010005
7. Díaz-Manríquez, A., Toscano, G., Coello Coello, C.A.: Comparison of metamodeling techniques in evolutionary algorithms. Soft Comput. **21**(19), 5647–5663 (Oct 2017). https://doi.org/10.1007/s00500-016-2140-z
8. Evans, L.C.: Partial Differential Equations. American Mathematical Society (Mar 2022)
9. Fix, E., Hodges, J.L.: Discriminatory Analysis - Nonparametric Discrimination: Small Sample Performance. Tech. Rep. ADA800391, University of California, Berkeley (1952)
10. He, C., Zhang, Y., Gong, D., Ji, X.: A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. Expert Syst. Appl. **217**, 119495 (May 2023). https://doi.org/10.1016/j.eswa.2022.119495
11. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm Evol. Comput. **1**(2), 61–70 (Jun 2011). https://doi.org/10.1016/j.swevo.2011.05.001
12. Li, G., Zhang, Q.: Multiple Penalties and Multiple Local Surrogates for Expensive Constrained Optimization. IEEE Trans. Evol. Comput. **25**(4), 769–778 (Aug 2021). https://doi.org/10.1109/TEVC.2021.3066606
13. Liu, R., Bianco, M.J., Gerstoft, P.: Automated partial differential equation identification. J. Acoust. Soc. Am. **150**(4), 2364 (Oct 2021). https://doi.org/10.1121/10.0006444
14. Liu, Y., Liu, J., Jin, Y., Li, F., Zheng, T.: A Surrogate-Assisted Two-Stage Differential Evolution for Expensive Constrained Optimization. IEEE Trans. Emerg. Topics Comput. **7**(3), 715–730 (Jun 2023). https://doi.org/10.1109/TETCI.2023.3240221

15. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: DACE: A MATLAB Kriging Toolbox. Tech. Rep. IMM-REP-2002-12, Informatics and Mathematical Modelling, DTU (Aug 2002)
16. Miranda-Varela, M.E., Mezura-Montes, E.: Constraint-handling techniques in surrogate-assisted evolutionary optimization. An empirical study. Appl. Soft Comput. **73**, 215–229 (Dec 2018). https://doi.org/10.1016/j.asoc.2018.08.016
17. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. AIAA J. **41**(4), 687–696 (Apr 2003). https://doi.org/10.2514/2.1999
18. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. **3**(2), 246–257 (1991). https://doi.org/10.1162/neco.1991.3.2.246
19. Preen, R.J., Bull, L.: Toward the Coevolution of Novel Vertical-Axis Wind Turbines. IEEE Trans. Evol. Comput. **19**(2), 284–294 (Apr 2015). https://doi.org/10.1109/TEVC.2014.2316199
20. Regis, R.G.: Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Comput. Oper. Res. (2011). https://doi.org/10.1016/j.cor.2010.09.013
21. Regis, R.G.: Evolutionary Programming for High-Dimensional Constrained Expensive Black-Box Optimization Using Radial Basis Functions. IEEE Trans. Evol. Comput. **18**(3), 326–347 (Jun 2014). https://doi.org/10.1109/TEVC.2013.2262111
22. Regis, R.G.: A Survey of Surrogate Approaches for Expensive Constrained Black-Box Optimization. In: World Congr. Global Optimiz. (WCGO). pp. 37–47. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-21803-4_4
23. Regis, R.G., Shoemaker, C.A.: Constrained Global Optimization of Expensive Black Box Functions Using Radial Basis Functions. J. Global Optimiz. **31**(1), 153–171 (Jan 2005). https://doi.org/10.1007/s10898-004-0570-0
24. Shi, L., Rasheed, K.: ASAGA: an adaptive surrogate-assisted genetic algorithm. In: Annu. Conf. Genet. Evol. Comput. (GECCO). pp. 1049–1056. GECCO '08, Association for Computing Machinery, New York, NY, USA (Jul 2008). https://doi.org/10.1145/1389095.1389289
25. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. Global Optimiz. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
26. Wang, W., Liu, H.L., Tan, K.C.: A Surrogate-Assisted Differential Evolution Algorithm for High-Dimensional Expensive Optimization Problems. IEEE Trans Cybern **53**(4), 2685–2697 (Apr 2023). https://doi.org/10.1109/TCYB.2022.3175533
27. Wang, Y., Li, J.P., Xue, X., Wang, B.C.: Utilizing the Correlation Between Constraints and Objective Function for Constrained Evolutionary Optimization. IEEE Trans. Evol. Comput. **24**(1), 29–43 (Feb 2020). https://doi.org/10.1109/TEVC.2019.2904900
28. Wang, Y., Yin, D.Q., Yang, S., Sun, G.: Global and Local Surrogate-Assisted Differential Evolution for Expensive Constrained Optimization Problems With Inequality Constraints. IEEE Trans Cybern **49**(5), 1642–1656 (May 2019). https://doi.org/10.1109/TCYB.2018.2809430
29. Wu, G., Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization.

Tech. rep., Nat. Univ. Defense Tech., China, Kyungpook Nat. Univ., South Korea, and Nanyang Tech. Univ., Singapore (2017)

30. Wu, Y., Yin, Q., Jie, H., Wang, B., Zhao, J.: A RBF-based constrained global optimization algorithm for problems with computationally expensive objective and constraints. Struct. Multidiscip. Optim. **58**(4), 1633–1655 (Oct 2018). https://doi.org/10.1007/s00158-018-1987-2
31. Yang, Z., Qiu, H., Gao, L., Cai, X., Jiang, C., Chen, L.: Surrogate-assisted classification-collaboration differential evolution for expensive constrained optimization problems. Inf. Sci. **508**, 50–63 (Jan 2020). https://doi.org/10.1016/j.ins.2019.08.054