Optimization and Engineering
https://doi.org/10.1007/s11081-025-09975-y

RESEARCH ARTICLE

Enhanced hybrid algorithm based on Bayesian optimization and Interior Point OPTimizer for constrained optimization

Loukas Kyriakidis[1] · **Martin Bähr**[1] · **Miguel Alfonso Mendez**[2]

Received: 12 November 2024 / Revised: 6 March 2025 / Accepted: 16 March 2025

Abstract
Optimization methods are essential for solving problems in fields like engineering, operations research, machine learning, and data science. In real-world applications, most optimization problems involve numerous constraints and nonlinear relationships, such as nonlinear balance equations. As a result, solving these constrained nonlinear optimization problems is challenging, with the global optimum often hidden behind multiple local ones. To address these challenges, recent work introduced the BO-IPOPT method, which combines Bayesian Optimization (BO) with the Interior Point OPTimizer (IPOPT) to leverage the global search of BO and the local search of IPOPT. To enhance the performance of the optimizer, this work explores aspects of the BO component, such as candidate selection for the Gaussian process model and constraint handling. First, we propose an effective strategy to provide IPOPT with good starting points without significantly increasing the computational costs of BO-IPOPT. Second, we introduce a novel approach using a single surrogate model for handling equality and inequality constraints in the BO part through the augmented Lagrangian framework with slack variables. This approach supports the global role of BO while reducing computational time and minimizing the number of hyperparameters defined by users. Third, a parameter study is conducted to optimize the hyperparameters in BO-IPOPT, minimizing user-defined inputs. This enhances the global guidance provided by the BO component and reduces the computational resources required by the hybrid method, improving its performance especially in problems of higher dimensions. Last, we evaluate BO-IPOPT against state-of-the-art methods using test and real-world problems with varying sizes, multimodality, and nonlinearity. Results show that BO-IPOPT achieves high accuracy, robustness, and efficiency, especially in high-dimensional, complex cases where other methods struggle.

✉ Loukas Kyriakidis
loukas.kyriakidis@dlr.de

1 Simulation and Virtual Design Department, German Aerospace Center, Institute of Low-Carbon Industrial Processes, Walther-Pauer-Straße 5, 03046 Cottbus, Germany

2 Environmental and Applied Fluid Dynamics Department, von Karman Institute for Fluid Dynamics, Waterloosesteenweg 72, 1640 Brussels, Belgium

Keywords Nonlinear constrained optimization · Bayesian optimization · IPOPT · Hybrid method · Augmented Lagrangian

1 Introduction

Numerical optimization is crucial in numerous areas, including engineering, data science, economics, finance, etc. (Pintér 2006; Andrei 2013; Kumar et al. 2020). The goal is to determine the values of decision variables that minimize an objective function while satisfying certain constraints that represent the limitations or requirements of the problem considered. The general problem can be written as:

$$\min_{\boldsymbol{x}\in\Omega} \left\{ f(\boldsymbol{x}) \;\text{ s. t. }\; \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}, \boldsymbol{x}_{\mathrm{L}} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\mathrm{U}} \right\} \tag{1}$$

with $\boldsymbol{x} \in \Omega \subseteq \mathbb{R}^n$ the n-dimensional decision variable contained in the set Ω, $\boldsymbol{x}_{\mathrm{L}} \in [-\infty, \infty)^n$ and $\boldsymbol{x}_{\mathrm{U}} \in (-\infty, \infty]^n$, with $x_{\mathrm{L}}^{(i)} \leq x_{\mathrm{U}}^{(i)}$ for $i = 1, \ldots, n$ the lower and upper bounds of the variables $\boldsymbol{x}$, $f : \mathbb{R}^n \to \mathbb{R}$ the objective function to be minimized, $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^p$ the set of equality constraints, and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^q$ the set of inequality constraints. A key challenge in numerical optimization is the increasing complexity of the search space as the number of variables and nonlinear, nonconvex constraints grow (Cao et al. 2018; Zhan et al. 2022), making it difficult to find the global optimum. In this work, we focus on problems that fall between two extremes: they are computationally expensive enough to make evolutionary algorithms less appealing, yet their dimensionality is large enough that optimizing the acquisition function over a continuous space in surrogate-based approaches like Bayesian optimization is non-negligible compared to evaluating the objective function and constraints. To address the challenges in numerical optimization, a variety of methods have been developed, broadly classified into global and local approaches (Nocedal and Wright 2006; Hendrix and Tóth 2010).

Global methods can be further subdivided into deterministic and stochastic methods. Deterministic global methods such as those used in state-of-the-art solvers such as BARON (Tawarmalani and Sahinidis 2002) rely on enumeration, cutting planes and bounding regions to exclude areas that cannot contain optimal solutions. However, these methods require significant computational efforts, especially for nonlinear and nonconvex problems, limiting them to problems with around 100 variables. A way to circumvent these limitations is to focus on linear (Parisio et al. 2014; Ma et al. 2017; Dowling et al. 2017) or linearized (Bischi et al. 2014; Pan et al. 2020; Xu et al. 2020) formulations of the optimization problems. Although this allows for faster solutions using linear optimizers, it often sacrifices realism and demands a careful balance between accuracy and computational cost in the linearization.

Stochastic global optimization, on the other hand, introduces randomness to explore the solution space. Depending on the criteria used to update the solution approximation, these methods can be further categorized into metaheuristics and surrogate-based techniques. Metaheuristic methods (e.g., genetic algorithms, simulated annealing, particle swarms) (Long and Wu 2014; Wang et al. 2020; Ngo et al. 2016; Zapata et al.

2019) mimic natural processes. Surrogate-based approaches like Bayesian optimization (Jeong and Obayashi 2005; Lan et al. 2022; Berk et al. 2019; Brochu et al. 2010; Roberts et al. 2024) use metamodels to iteratively approximate the objective function and constraints. Stochastic methods are less vulnerable to local minima, but typically require more function evaluations (total number of evaluations for f, $\boldsymbol{g}$ and $\boldsymbol{h}$ together). Competitive swarm optimizers (CSO), derived from particle swarm optimization, have been developed to enhance stochastic and metaheuristic methods for large-scale optimization. Improvements include competitive mechanisms for faster convergence, two-stage particle updates, and strategies to avoid local optima (Cheng and Jin 2015; Zhang et al. 2018; Tian et al. 2020; He et al. 2021; Ming et al. 2023). Recent work introduced fast-converging competitive and cooperative swarm optimizers (F-CCSO) for constrained multi-objective optimization, leveraging competitive–cooperative swarm interactions to approximate the Pareto front more efficiently (Zhou et al. 2024). However, more enhancements are needed to address complex problems by refining these interactions.

Local methods, or local search algorithms, aim to find the optimal solution near a given starting point. These methods can be classified as gradient-free (Larson et al. 2019) (e.g., Nelder–Mead) and gradient-based (Andrei 2017) (e.g., quasi-Newton methods), depending on whether the iterate is based on the gradient information or not (Nocedal and Wright 2006). Gradient-free methods are suitable for nonsmooth or noisy functions where derivatives are uninformative, while gradient-based methods are typically applied to well-conditioned, high-dimensional problems. Compared to global methods, local searches converge much faster, requiring fewer function evaluations, especially when the starting point is well-chosen or the objective function is locally convex. However, their main drawback is the risk of getting trapped in local minima, making them less effective for multi-modal or nonlinear nonconvex optimization problems.

A conceptual globalization of local methods are the multi-start (MS) algorithms (Martí et al. 2018), where the local search is performed from multiple starting points and the local solution with the best objective function value is selected as the best estimate for the global optimum. The main disadvantage of MS strategies is that many starting points may converge to the same local minimum, wasting computational effort without yielding new information. To mitigate this risk, techniques such as tunneling, clustering, scatter search, and memory structure(Arora et al. 1995; Martí et al. 2019; Lasdon and Plummer 2008; Ugray et al. 2007) are used to improve the selection of starting points. However, most heuristic MS methods are challenging for high-dimensional problems, so random starting points are often used in practice.

Another promising strategy is to combine global stochastic methods with local optimization techniques (Long and Wu 2014; Asim et al. 2018; Kelner et al. 2008; Chelouah and Siarry 2003; Ulder et al. 1990), taking advantage of the strengths of both approaches. These hybrid approaches are generally classified into sequential and integrative types (Cherki et al. 2019), although this classification varies in the literature. In sequential methods (Attaviriyanupap et al. 2002; Sivasubramani and Swarup 2010), a stochastic method is used initially and its approximate solution is used as a starting point for the local search. This approach often requires a large number of function evaluations during the stochastic phase to get a solution close to the global optimum. In contrast, integrative methods (Long and Wu 2014; Kelner et al. 2008; Ulder et al. 1990) have both stochastic and local techniques that work simultaneously in an iterative manner, allowing them to complement each other more effectively. As a result, integrative approaches generally offer higher sample efficiency and better performance. The most popular hybrid methods in this category combine metaheuristics with local techniques (Long and Wu 2014; Chelouah and Siarry 2003; Ulder et al. 1990), but more recent work has focused on integrating surrogate-based methods with gradient-based optimization (Luo et al. 2022; Gao et al. 2020; Kyriakidis et al. 2024). Surrogate-based hybrid models provide a more efficient alternative by progressively approximating objective functions, reducing computational costs compared to metaheuristic-based hybrids. However, their application to high-dimensional problems remains challenging because of the increasing complexity with more samples. Recently, several strategies within high-dimensional Bayesian optimization (HDBO) have been developed to handle large search spaces more efficiently.

An approach to improve HDBO efficiency was proposed by Wang et al. (2016), which projected the high-dimensional space into a lower-dimensional subspace using random linear embeddings (REMBO). Another approach called HESBO (Nayebi et al. 2019) focuses on optimizing functions in lower-dimensional spaces identified through random embeddings and structure learning. Eriksson et al. (2019) introduced TuRBO (Trust-Region Bayesian Optimization), a scalable global optimization method that divides the search space into smaller, more manageable regions, enabling efficient optimization in high dimensions, while Binois et al. (2020) further explored the idea of dimensionality reduction by discussing the importance of selecting the appropriate low-dimensional domain for global optimization via random embeddings. In more recent work, Priem et al. (2023) combined random and supervised embeddings in their high-dimensional efficient global optimization method "EGORSE". This approach takes advantage of both random embeddings and supervised learning to identify promising regions in the search space, enhancing the efficiency of the optimization process in high-dimensional settings. Nevertheless, all the aforementioned BO methods have been developed for unconstrained optimization.

Incorporating constraints in high-dimensional settings is challenging but essential in many real-world applications. Constraints add complexity by requiring a balance between optimization and feasibility, which is difficult to model in high-dimensional spaces. While unconstrained methods provide a strong foundation, recent advances in Gaussian Process Regression (GPR) for constrained optimization offer promising extensions. Bouhlel et al. (2018) developed an approach (SEGOKPLS(+K)) that combines the SEGO algorithm (Super Efficient Global Optimization, Sasena et al. 2001) with kriging models and with partial least squares (PLS), addressing the challenges of constrained problems of high dimensions by effectively reducing the dimensionality. In addition, Eriksson and Poloczek (2021) extended the idea of TuRBO to handle constrained optimization problems, developing SCBO (Scalable Constrained Bayesian Optimization).

In Kyriakidis et al. (2024), we introduced BO-IPOPT, a hybrid method that combines surrogate-based Bayesian optimization (BO) with the Interior Point OPTimizer (IPOPT) for local search. BO-IPOPT leverages the global exploration capabilities of BO with the local efficiency and feasibility control of IPOPT. Unlike traditional BO approaches that aim to build precise surrogate models for both the objective function and constraints, our method uses a single surrogate model to provide a coarse approximation of the cost function. The integration of BO with IPOPT enables the hybrid method to handle constraints by incorporating feasibility checks directly into the optimization workflow. During each iteration, IPOPT refines candidate solutions proposed by the BO-guided surrogate model, ensuring that they satisfy constraints to the extent possible. This capability distinguishes BO-IPOPT from non-constrained optimization methods such as BOwLS (Gao et al. 2020), which focus solely on improving the accuracy of the objective function without addressing the handling of constraints. By extending BOwLS to constrained problems, BO-IPOPT provides a structured framework for balancing exploration and exploitation while adhering to the feasibility requirements inherent in constrained optimization tasks. Our approach aims to reduce both computational time and the number of function evaluations needed to reach the global minimum, especially in complex, high-dimensional problems, compared to state-of-the-art methods. Although BO-IPOPT has shown promising results in solving non-linear constrained optimization problems, several technical aspects within the BO framework still need further investigation to improve its performance. A key area of improvement includes enhancing the speed at which the global optimum is achieved in high-dimensional problems. This requires improving hyperparameter selection in the BO component to provide effective global guidance for IPOPT, especially in multimodal functions of higher dimensions, optimizing the selection of sample points, and investigating better strategies for handling constraints to ensure that BO provides high-quality candidates to IPOPT.

This work aims to refine BO-IPOPT to improve its performance in constrained optimization problems, with a focus on enhancing the BO component. Although BO is typically used as a standalone optimizer for black-box problems, its methods for sampling point selection, constraint handling, and hyperparameter settings need adjustment when guiding IPOPT rather than working standalone.

First, we propose selecting candidate solutions by evaluating the acquisition function over a finite set of points rather than across the entire continuous space, as it is commonly done in the literature. This provides good starting points for IPOPT while avoiding the computational cost of an optimizer for the acquisition function. Second, we introduce an approach to handle constraints using, unlike traditional methods, a single surrogate model for both objective function and constraints, integrated with an augmented Lagrangian (AL) framework and combined with IPOPT. This simplifies the process, reduces the number of hyperparameters, avoids overfitting and efficiently manages high-dimensional problems with numerous constraints. Third, to improve the global guidance offered by BO to IPOPT compared to (Kyriakidis et al. 2024) and to avoid scalability issues in problems of high dimensions, we conduct a parameter study to identify optimal values that balance convergence speed with computational cost, thereby minimizing the need for user-defined inputs.

The results show that BO-IPOPT outperforms competing methods in efficiency, particularly in high-dimensional problems, and scales linearly with the problem dimensions, making it well suited for larger-scale optimization tasks. This finding highlights the practical advantage of BO-IPOPT for complex real-world problems.

This paper is organized as follows. Section 2 introduces the proposed hybrid BO-IPOPT method and Sect. 3 discusses some technical aspects related to the BO part of this method, which are addressed in this work. In Sect. 4, we first study open issues in detail using several benchmark problems for fine-tuning BO-IPOPT, and compare the performance and scalability of our proposed method with state-of-the-art approaches. Finally, this paper ends with a conclusion and provides an outlook for future work.

2 Optimization methods

In this work, we consider nonlinear nonconvex constrained optimization problems as defined by (1). The degree of complexity for real-world optimization problems depends on the known functions f, $\boldsymbol{h}$, $\boldsymbol{g}$, which are usually nonlinear and nonconvex. All the optimizers considered in the following provide a sequence of candidate solutions $\{\boldsymbol{x}_K\}$ that hopefully converge efficiently toward the global minimum $\boldsymbol{x}_K \rightarrow \boldsymbol{x}^\star$.

To better understand the interaction between the BO and IPOPT in the proposed hybrid BO-IPOPT solver, in the following we briefly review each approach in its standalone implementation in Sects. 2.1 and 2.2 before discussing their hybridization in Sect. 2.3.

Optimization algorithm performance is typically assessed using benchmark problems, classified by the number of variables. We arbitrarely define problems as small scale if $n \leq 100$, medium-scale if $100 < n \leq 5{,}000$, and large-scale $n > 5{,}000$. In this work, we focus on small and medium scale problems, for which the functions in (1) are known (i.e., their gradient can be efficiently evaluated). However, we consider constrained problems that are highly nonconvex and challenging for local search methods, aiming to improve sample efficiency compared to traditional global stochastic approaches.

2.1 Bayesian optimization

The BO framework is a global optimization method for expensive-to-evaluate black-box functions. It builds surrogate models using samples at candidate solutions $\{\boldsymbol{x}_K\}$, typically using GPR (Rasmussen and Williams 2005; Hebbal et al. 2023), which allows for the analytical calculation of both the cost function estimate and its uncertainty, enabling a balance between exploitation (i.e., sampling where the surrogate predicts the best values) and exploration (i.e., sampling where the uncertainties are the highest). The sampling is driven by an acquisition function.

Several variants have been proposed to account for constraints. An approach is to constrain the acquisition function with the probability of a violation of the constraint (Gelbart et al. 2014; Gardner et al. 2014). Another approach, valid for decoupled constraints, is the predictive entropy search (Hernández-Lobato et al. 2015) based on the expected information gain. Pelamatti et al. (2024) use a multi-output Gaussian process to optimally decide which constraint should be evaluated at each point while Lam and Willcox (2017) propose a "look-ahead approach" in which the next evaluation is selected to maximize the long term feasible reduction of the objective function. Ariafar et al. (2019) propose the alternating direction method of multipliers, augmenting the objective function with auxiliary variables while Gramacy et al. (2016), in the framework of Augmented Lagrangian (AL), transform the constrained optimization problem into a sequence of unconstrained problems that are iteratively optimized. An alternative approach is introduced in Picheny (2014), where a step-wise uncertainty reduction strategy is considered to address constrained optimization problems, offering an efficient trade-off between exploration and local search near the boundaries. All these methods handle only inequality constraints.

To deal with both equality and inequality constraints, a classic approach is to transform equality constraints $\boldsymbol{h}(\boldsymbol{x})$ into two inequality constraints of the type $\boldsymbol{h}(\boldsymbol{x}) \geq \mathbf{0}$ and $-\boldsymbol{h}(\boldsymbol{x}) \geq \mathbf{0}$. However, this increases the number of constraints and might lead to constraint qualification issues (Nocedal and Wright 2006). A state-of-the-art approach that addresses both equality and inequality constraints without any transformation is the ALBO (Picheny et al. 2016), which combines the unconstrained BO with the AL framework. In addition, Priem et al. (2020) extended the SEGO method, which builds a surrogate model for every inequality constraint, to handle both types of constraints. To address the issue of poorly modeled constraints obtained in SEGO, Priem et al. (2020) uses a fixed scalar upper trust bound introduced in Lam and Willcox (2017); Lam et al. (2015); Priem et al. (2019) to enhance the exploration of the design space. A version of the upper trust method was presented in Lu and Paulson (2022), where an exact penalty function is used on the lower confidence bound acquisition function to solve expensive constrained black-box optimization problems.

Existing methods for handling equality and inequality constraints, such as ALBO and SEGO, rely on multiple surrogate models for different constraints, resulting in high computational complexity, especially in high-dimensional problems with numerous constraints. This approach also increases the number of hyperparameters to tune and raises the risk of overfitting, as each surrogate model is trained independently. To address these issues, we propose a single surrogate model for both the objective function and constraints. While single-model approaches are often avoided due to limitations in prediction accuracy, our hybrid framework combines BO for global guidance with IPOPT, eliminating the need for precise predictions. This allows the surrogate model to focus on providing the global direction necessary for efficient optimization. To integrate equality and inequality constraints into a single objective function, we employ two approaches. First, the objective function is augmented with both equality and inequality constraints as penalty terms (Long and Wu 2014):

$$u(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\delta}^\top \boldsymbol{h}(\boldsymbol{x})^2 + \boldsymbol{\tau}^\top \max(\mathbf{0}, \boldsymbol{g}(\boldsymbol{x}))^2 \tag{2}$$

where $\boldsymbol{\delta}, \boldsymbol{\tau} \geq \mathbf{0}$ are penalty weight vectors associated with the magnitude of the constraint violation. This approach has been considered in Kyriakidis et al. (2024). Second, we follow the AL framework (Picheny et al. 2016) with slack variables based on the ALBO approach to transform inequalities into equality constraints and iteratively solve a sequence of simpler (unconstrained) problems. The AL is defined as follows:

$$u(\boldsymbol{x}, \boldsymbol{s}) = f(\boldsymbol{x})+\boldsymbol{\lambda}_{\boldsymbol{g}}^{\top}(\boldsymbol{g}(\boldsymbol{x})+\boldsymbol{s})+\boldsymbol{\lambda}_{\boldsymbol{h}}^{\top}\boldsymbol{h}(\boldsymbol{x})+\frac{1}{2\rho}\left[\sum_{w=1}^{q}(g_w(\boldsymbol{x})+s_w)^2+\sum_{r=1}^{p}h_r(\boldsymbol{x})^2\right] \quad (3)$$

where $\rho > 0$ is a penalty parameter on constraint violation, $\boldsymbol{\lambda}_{\boldsymbol{g}} \in \mathbb{R}_+^q$ and $\boldsymbol{\lambda}_{\boldsymbol{h}} \in \mathbb{R}_+^p$ denote the Lagrange multiplier vectors for the inequality and equality constraints, respectively, and $\boldsymbol{s}$ describes the slack variable vector expressed by the following equation:

$$\boldsymbol{s} = \max\{\boldsymbol{0}, -\boldsymbol{\lambda}_{\boldsymbol{g}}\rho - \boldsymbol{g}(\boldsymbol{x})\} \quad (4)$$

The Lagrange multipliers, the penalty parameter, and the slack variable vector are updated at each outer iteration K, building a dynamic process that is described in more detail in Appendix A. This approach has been selected because the concept of slack variables is similar to the one used in IPOPT, enabling a seamless integration between the two frameworks. In addition, the dynamic update of the parameters re reduces the number of hyperparameters that need to be manually tuned by the user, simplifying the optimization process. It is important to note that solving high-dimensional problems is challenging with this iterative process, as the AL is not solved analytically and must handle many constraints, especially with infeasible points. Nevertheless, in all the investigated test cases, the AL framework worked effectively in the BO-IPOPT formulation as further discussed in Sect. 4.

In the GPR, we use a standard Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-d(\boldsymbol{x}, \boldsymbol{x}')^2/2l^2)$, where l is the length scale of the kernel and $d(\cdot, \cdot)$ is the Euclidean distance between points in Ω. In the GPR, the kernel matrices are regularized using standard Tikhonov regularization, with a ridge parameter α. Therefore, the key hyperparameters in the process are the l, α.

For the acquisition function, there are some popular functions like the expected improvement (EI), WB2, WB2S, probability of improvement, and lower confidence bound, see e.g., Greenhill et al. (2020); Brochu et al. (2010); Watson and Barnes (1995); Bartoli et al. (2019). While the choice of acquisition function often depends on the specific requirements of the optimization problem, we consider EI in the current work due to its widespread use (one of the most widely used acquisition functions) in the field and its proven effectiveness, balancing exploration and exploitation. EI is defined for each new sample $\bar{\boldsymbol{x}}$ as follows:

$$\mathrm{EI}(\bar{\boldsymbol{x}}|\hat{u}) = \begin{cases} (\hat{u}(\boldsymbol{x}^+) - \mu(\bar{\boldsymbol{x}}) - \xi)\Phi(Z) + \sigma(\bar{\boldsymbol{x}})\phi(Z), & \text{if} \quad \sigma(\bar{\boldsymbol{x}}) > 0 \\ 0, & \text{if} \quad \sigma(\bar{\boldsymbol{x}}) = 0 \end{cases} \quad (5)$$

with

$$Z = \begin{cases} \frac{\hat{u}(\boldsymbol{x}^+) - \mu(\bar{\boldsymbol{x}}) - \xi}{\sigma(\bar{\boldsymbol{x}})}, & \text{if} \quad \sigma(\bar{\boldsymbol{x}}) > 0 \\ 0, & \text{if} \quad \sigma(\bar{\boldsymbol{x}}) = 0 \end{cases} \quad (6)$$

where $\mu(\bar{\boldsymbol{x}})$ and $\sigma(\bar{\boldsymbol{x}})$ are the mean and the standard deviation of the GPR predictions, respectively. Φ and ϕ denote the cumulative and probability distribution functions, respectively. The first term regulates the exploitation, while the second term controls

Algorithm 1 Classical BO Algorithm

```
Input: Length scale $l$; regularization term $\alpha$; exploration $\xi$; number of initialization points $N_0$; number of candidates $N_\mathrm{c}$ considered in EI; penalty weight vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$; Lagrange multiplier vectors $\boldsymbol{\lambda_g}$, $\boldsymbol{\lambda_h}$; number of outer iterations $K$
1: Generate an initial set of points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_0}\}$ in $\Omega$
2: Evaluate $y_{i_0} = u(\boldsymbol{x}_{i_0})$ or $y_{i_0} = u(\boldsymbol{x}_{i_0}, \boldsymbol{s}_{i_0})$ for $i_0 = 1{:}N_0$
3: Let $D_0 = \{(\boldsymbol{x}_{i_0}, y_{i_0})\}_{i_0=1}^{N_0}$
4: Construct a GP model $\hat{u}_0$ from $D_0$
5: $z = N_0$, $j = 0$, $FE = N_0$
6: while $j < K$ do
7:     Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{N_\mathrm{c}}\}$ in $\Omega$
8:     $\boldsymbol{x}_{z+1} \in \arg\max_{\boldsymbol{x} \in D_j} \mathrm{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_\mathrm{c}$
9:     $y_{z+1} = u(\boldsymbol{x}_{z+1})$
10:    $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}, y_{z+1})\}$
11:    Update GP model $\hat{u}_{j+1}$ from $D_{j+1}$
12:    $z := z + 1$, $j := j + 1$, $FE := FE + 1$
13: end while
14: return $y_\mathrm{min} = \min\{y_j\}_{j=1}^{K}$
```

the exploration. The parameter ξ defines a threshold for the minimum EI that justifies the exploration and is another hyperparameter of BO. The larger this value, the more BO explores the entire space. It should be noted that the generation of new candidate solutions is usually based on the optimization of EI in traditional BO approaches (Brochu et al. 2010; Lu and Paulson 2022; Malu et al. 2021). The classical BO algorithm starts with a set of randomly selected candidate solutions and iterates alternately, updating the GPR and maximizing the EI to define a new candidate, as presented in Algorithm 1. The variable FE denotes the function evaluations.

2.2 Interior Point OPTimizer

The IPOPT solver from COIN-OR is one of the most widely used open-source tools for solving large-scale nonlinear, nonconvex constrained optimization problems (1). Developed in Wächter and Biegler (2006), this algorithm uses a primal–dual interior-point method combined with a filtered line-search method to find a local solution. IPOPT requires all the functions in (1) to be at least once and ideally twice continuously differentiable.

IPOPT solves the optimization problem in (1) by approximating solutions for a sequence of barrier problems, iteratively approaching the optimum from within the feasible set. To handle inequality constraints, it introduces slack variables, transforming them into equalities $\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s} = \boldsymbol{0}$ with $\boldsymbol{s} \geq \boldsymbol{0}$. With $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{s})$ and $\boldsymbol{c}(\tilde{\boldsymbol{x}}) = (\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s})$, the optimization problem takes the following form:

$$\min_{\tilde{\boldsymbol{x}} \in \tilde{\Omega}} \left\{ f(\tilde{\boldsymbol{x}}) \ \text{ s. t. } \ \boldsymbol{c}(\tilde{\boldsymbol{x}}) = \boldsymbol{0}, \tilde{\boldsymbol{x}}_\mathrm{L} \leq \tilde{\boldsymbol{x}} \leq \tilde{\boldsymbol{x}}_\mathrm{U} \right\} \tag{7}$$

The latter formulation is modified by introducing a natural logarithmic barrier term into the objective function that handles the boundaries of the variable vector $\tilde{\boldsymbol{x}}$. In other words, IPOPT attempts to find a Karush–Kuhn–Tucker optimality point for (7) by solving a sequence of problems with a decreasing barrier parameter $\mu \in \mathbb{R}^+$ of the form:

$$\min_{\tilde{\boldsymbol{x}}(\mu)\in\tilde{\Omega}} \left\{ f(\tilde{\boldsymbol{x}}) - \mu \left[\sum_{i=1}^{n+q} \ln(\tilde{x}_i - \tilde{x}_{\mathrm{L}}) + \sum_{i=1}^{n+q} \ln(\tilde{x}_{\mathrm{U}} - \tilde{x}_i) \right] \text{ s. t. } \boldsymbol{c}(\tilde{\boldsymbol{x}}) = \boldsymbol{0} \right\} \tag{8}$$

The nonlinear primal–dual problems in (8) are solved for the corresponding barrier parameter such that for $\mu \to 0$ one has $\tilde{\boldsymbol{x}}^\star(\mu) \to \tilde{\boldsymbol{x}}^\star$, which then also solves (1). For each barrier subproblem with the previous approximated solution as the starting point, a Newton-type algorithm with line search is applied, including the solution of indefinite sparse symmetric linear systems. The overall performance (runtime, accuracy and robustness) strongly depends on the properties of the chosen sparse linear solver. The MA27 is used in the open-source version. It should be noted that IPOPT internally uses a special Hessian regularization to avoid maximizers and saddle points. For further details of the algorithm, we refer the reader to Wächter and Biegler (2006); Wächter (2002).

Overall, IPOPT is a highly efficient solver for finding a local minimum of (1), but the performances strongly depends on the starting point as for any local method.

2.3 Hybrid method BO-IPOPT

The proposed BO-IPOPT method, firstly introduced in Kyriakidis et al. (2024), is illustrated in Algorithm 2. The algorithm is divided into two blocks: an initialization step and a main part as a loop over the number of outer iterations K. Below, we describe the procedure in more detail.

In the initialization step (lines 1–4), a set of initialization points is generated by an appropriate method and then evaluated using the selected augmented objective function in order to generate an initial surrogate model via GPR. Unlike traditional BO, which builds separate GP models for the objective function and constraints, the role of BO in the hybrid BO-IPOPT approach is to offer global exploration, while IPOPT handles local refinement and ensures constraint satisfaction at each iteration. It should be noted that the input parameters l and α are considered in the GP model, while the exploration term ξ is included in EI. The input vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$, $\boldsymbol{\lambda_g}$ and $\boldsymbol{\lambda_h}$ are required for the objective function in line 2 of Algorithm 2, which is defined by (2) or (3).

The algorithm alternates the BO and IPOPT steps until K is reached (lines 6–13). Specifically, candidates with the highest EI values are selected to update the GPR by evaluating the acquisition function, either over a discrete set of candidates or the entire continuous space, similar to EGO. In either case, multiple candidate points are chosen for each outer iteration K as starting points for IPOPT, which then optimizes the real objective function and constraints. The number of function evaluations FE at each iteration consists of the number of best candidate points selected by the BO part plus the function evaluations performed by IPOPT for each candidate.

It is worth noticing that the current implementation risks selecting points from the same region, potentially converging to the same local optimium. This risk could be mitigated by using the multipoint expected improvement (q-EI, Wang et al. 2020), which optimizes the expected improvement across a batch of points and accounts for correlations between them. We plan to explore q-EI in the BO-IPOPT formulation in future works. Nevertheless, while the GPR is updated with the EI selected candidate solutions in BO, in the BO-IPOPT the IPOPT uses these candidates as starting points, and hence the GPR is updated from local optima that might significantly differ from those identified by the EI.

Finally, the algorithm returns (line 14) the minimum objective function value included in the set of points D without considering the initial set of points D_0, so that $y_{\min}$ always represents at least a local solution of (1) or better the global minimum and all the constraints are met.

The proposed algorithm only uses feasible solutions (line 9). If the local solution from IPOPT is not feasible, the next-best candidate is selected, and the IPOPT is restarted. This strategy has successfully led BO-IPOPT to feasible solutions in all cases in this work, and we have not encountered situations where the algorithm failed. For challenges with infeasibility due to a small feasible set volume in future work, methods such as repair functions or constraint relaxation (Samanipour and Jelovica 2020; Sankaran 1993) could be applied. Moreover, the hybrid method recommends selecting more than one best candidate at each iteration (line 8), which is usually not the case in the classical BO algorithm. In this way, we expect that each additional best candidate increases the probability that BO-IPOPT finds the global optimum faster. At the same time, the computational costs of the optimizer naturally increase as more IPOPT searches are performed sequentially.

This approach can become time-consuming for large, complex problems with high-dimensional search spaces. Parallelizing the evaluation of selected points can significantly reduce computation time by leveraging the independence of these evaluations. Using multiple processors allows simultaneous processing, speeding up the optimization process. Within the BO-IPOPT framework, we propose using Python's MULTIPROCESSING module (Foundation 2023), specifically the POOL function, to manage parallel execution. This function dynamically assigns tasks to worker processes and collects results upon completion, enabling efficient parallelization. The total CPU time is reduced to the average runtime of the local solver for a single point, plus communication and synchronization overhead. This approach is especially advantageous for large, computationally intensive problems, though its benefits diminish for smaller problems where solver runtimes are already minimal, and communication costs can outweigh the gains.

It should also be underlined that the current BO-IPOPT formulation handles only continuous variables, but future work could extend it to include both integer and continuous variables. For the BO, an adaptive dimension reduction process (Saves et al. 2023) can be used to address mixed-integer variables while for IPOPT, approaches such as relaxation, branch-and-bound, outer approximation, and the extended cutting plane method (Belotti et al. 2013) can be investigated to enable the local solver to handle mixed-integer variables.

Algorithm 2 Hybrid Method BO-IPOPT

```
Input: Length scale $l$; regularization term $\alpha$; exploration $\xi$; number of initialization points $N_0$; number of candidates $N_c$ considered in EI; number of best candidates $N_{bc}$; penalty weight vectors $\boldsymbol{\delta}, \boldsymbol{\tau}$; Lagrange multiplier vectors $\boldsymbol{\lambda_g}, \boldsymbol{\lambda_h}$; number of outer iterations $K$
1: Generate an initial set of points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_0}\}$ in $\Omega$
2: Evaluate $y_{i_0} = u(\boldsymbol{x}_{i_0})$ or $y_{i_0} = u(\boldsymbol{x}_{i_0}, \boldsymbol{s}_{i_0})$ for $i_0 = 1{:}N_0$
3: Let $D_0 = \{(\boldsymbol{x}_{i_0}, y_{i_0})\}_{i_0=1}^{N_0}$
4: Construct a GP model $\hat{u}_0$ from $D_0$
5: $z = N_0$, $j = 0$, $FE = N_0$
6: while $j < K$ do
7:     Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{N_c}\}$ in $\Omega$
8:     $\{\boldsymbol{x}_{z+1}, \dots, \boldsymbol{x}_{z+N_{bc}}\} \in \underset{\boldsymbol{x} \in D_j}{\arg\max}\, \mathrm{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_c$
9:     Solve $[y_k^\star, \boldsymbol{x}_k^\star] = \mathrm{IPOPT}(\boldsymbol{x}_k)$ for $k = z+1{:}z+N_{bc}$
10:    $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}^\star, y_{z+1}^\star), \dots, (\boldsymbol{x}_{z+N_{bc}}^\star, y_{z+N_{bc}}^\star)\}$
11:    Update GP model $\hat{u}_{j+1}$ from $D_{j+1}$
12:    $z := z + N_{bc}$, $j := j+1$, $FE := FE + N_{bc} + \sum_{k=z+1}^{z+N_{bc}} FE_{\mathrm{IPOPT}(\boldsymbol{x}_k)}$
13: end while
14: return $y_{\min} = \min\{y_j^\star\}_{j=1}^{KN_{bc}}$
```

The proposed BO-IPOPT method has two key advantages. First, IPOPT accelerates BO by moving candidate solutions toward optimal locations. If these are local optima, the EI evaluation ensures global exploration and improves regression when needed. If a global minimum is found, EI prioritizes its sampling in future iterations. Second, IPOPT benefits from the good starting points provided by BO. Notably, any local solver could replace IPOPT, which was chosen for its efficiency, robustness, and open-source availability.

A final point to discuss is how the BO part in our hybrid method deals with the constraints of the underlying optimization problem. The best candidates are evaluated in the classical BO based on the selected augmented objective function. By our construction, a local solution $\boldsymbol{x}^\star$ for (2) and (3) implies $u(\boldsymbol{x}_k^\star) \equiv f(\boldsymbol{x}_k^\star)$ and $u(\boldsymbol{x}_k^\star, \boldsymbol{s}(\boldsymbol{x}_k^\star)) \equiv f(\boldsymbol{x}_k^\star)$, respectively. This results directly from the fact that a local solution fulfills all constraints (rounding errors are neglected here); see also Appendix A for the AL framework. As a result, both techniques only directly influence the initialization points and the corresponding initial GPR built from these points. Nevertheless, the updated GP model is indirectly affected at each outer loop since D_0 is a subset of D_j considered in the acquisition function EI.

3 Specific contributions of this work

This work extends several aspects of the BO-IPOPT formulation presented in Kyriakidis et al. (2024). First, we explore effective methods for selecting new candidates for the GPR. While continuous optimization of acquisition functions is common in the literature, we show that evaluating EI over a discrete set, combined with IPOPT, offers an effective balance between convergence speed and computational efficiency in BO-IPOPT. This approach preserves BO's global guidance while addressing scalability challenges in high-dimensional problems. Additionally, we advance constraint handling in BO by using a single surrogate model within an AL framework integrated with IPOPT. Unlike methods requiring multiple surrogate models for individual constraints, our approach reduces computational complexity, minimizes overfitting risk, and streamlines the handling of equality and inequality constraints without excessive hyperparameter tuning. This method achieves a robust balance of efficiency and accuracy, making it well-suited for high-dimensional optimization.

Third, we analyze BO-IPOPT's hyperparameters to improve BO's global guidance while minimizing user input and avoiding the computational overhead of learning these parameters during optimization, a common issue in the literature. Fourth, we assess BO-IPOPT's performance and scalability both qualitatively and quantitatively, comparing it with traditional methods and other BO approaches for high-dimensional problems. Finally, we address other issues such as the choice of the local solver and the use of local searches instead of random points to initialize the GPR.

4 Numerical experiments

In this section, we analyze and evaluate the proposed BO-IPOPT method on the open issues presented in Sect. 3 at hand of four nonlinear constrained optimization problems selected with respect to the problem size (number of decision variables), the degree of multimodality (number of global/local optima) and the degree of complexity (number/type of nonlinear terms). We distinguish between two test problems and two real-world problems, more precisely, the well-known dynamic economic dispatch (DED) problem and the optimal operation of a renewable steam generation (RSG) system. In doing so, the test problems and the 5-unit DED problem are firstly used to investigate the acquisition function evaluation, constraint handling, parameter study and performance comparison of the fine-tuned BO-IPOPT with SCBO, SEGOKPLS and SEGOKPLS+K, while the 10-unit and 30-unit DED problem as well as the RSG system (along with the test problems) then serve to examine the solver performance of the fine-tuned BO-IPOPT method compared to the classical random-based MS-IPOPT and the hybrid BOwLS method in terms of accuracy, CPU time and robustness.

On the one hand, MS-IPOPT (cf. Algorithm 4 in Appendix B) is selected for solver comparison in this study because MS algorithms are widely used in practice, as already mentioned in Sect. 1. This study aims to investigate whether BO provides better starting points for IPOPT in our hybrid approach than the random-based starting points of MS-IPOPT, thus ensuring a higher probability of obtaining lower objective function values. On the other hand, the competing hybrid method BOwLS (cf. Algorithm 5 in Appendix C) is considered as it is similar in structure to BO-IPOPT, but has some technical differences described in Appendix C. SCBO, SEGOKPLS and SEGOKPLS+K are considered in this study, since they are effective BO approaches for constrained optimization problems of higher dimensions.

All algorithms are implemented in PYTHON 3.8 (Van Rossum and Drake 2009) and include GAMS (GAMS Development Corporation) or PYOMO (Hart et al. 2011) programs, which are used to deterministically solve the optimization problems with the existing solver package IPOPT (with default settings). The interface between PYTHON and GAMS is built by GAMS API (GAMS Development Corporation). Moreover, BO is implemented with the SKLEARN library (Pedregosa et al. 2011) for the GPR and the generation of random points (for the initialization and a new group of points at each outer iteration for the acquisition function) is implemented using the LATINHYPERCUBE package from SCIPY (Virtanen et al. 2020). Note that the PYOMO and GAMS environment are used for the test and real-world problems, respectively. Further details on using the different environments can be found in Sect. 4.2.5. All computations were carried out with an INTEL(R) CORE(TM) i7-8665U CPU.

In the following, we first present the optimization problems to be investigated and then give a detailed study of the proposed BO-IPOPT method.

4.1 Benchmark problems

Below, we briefly introduce the four constrained optimization problems (two test and two real-world problems) considered in this work; for more details we refer the reader to the original papers. It should be mentioned that the second and fourth benchmark problem were already used in our paper (Kyriakidis et al. 2024), but without a detailed qualitative analysis of BO-IPOPT, which is now addressed in this paper.

4.1.1 Simple test problem

The first artificial test problem is taken from Sakawa and Yauchi (2000) and has the following form:

$$\begin{aligned}
\min f(\boldsymbol{x}) = {} & x_1^3 + (x_2-5)^2 + 3(x_3-9)^2 - 12x_3 + 2x_4^3 + 4x_5^2 + (x_6-5)^2 - 6x_7^2 \\
& + 3(x_7-2)x_8^2 - x_9x_{10} + 4x_9^3 + 5x_1x_3 - 3x_1x_7 + 2x_8x_7 \\
& \text{subject to} && (9) \\
& -3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4 - 2x_5x_6x_8 + 120 \geq 0, && (10) \\
& -5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0, && (11) \\
& -x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 - 6x_5x_6 \geq 0, && (12) \\
& -0.5(x_1-8)^2 - 2(x_2-4)^2 - 3x_5^2 + x_5x_8 + 30 \geq 0, && (13) \\
& 3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0, && (14) \\
& 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 105, && (15) \\
& 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0, && (16) \\
& -8x_1 + 2x_2 + 5x_9 - 2x_{10} \leq 12, \quad \boldsymbol{x} \in [-5, 10]^{10} && (17)
\end{aligned}$$

The optimization problem (9)–(17) describes a small-scale nonlinear nonconvex optimization problem of dimension 10, which is constrained only by inequalities. The problem has multiple local minima and a global minimum $\boldsymbol{x}^\star$ with $f(\boldsymbol{x}^\star) \approx -216.65649$ quickly computed by the BARON solver. In addition, it should be mentioned that several inequalities are active in $\boldsymbol{x}^\star$.

4.1.2 Constrained Ackley function

The second test problem considered here is the well-known Ackley function, which is often used to demonstrate the performance of optimization algorithms. This function is nonlinear, nonconvex and highly multi-modal with several local minima and a global minimum. To set up a constrained optimization problem, the function (which was originally designed without constraints) is equipped with two inequality constraints as described in Eriksson and Poloczek (2021). The resulting n-dimensional constrained optimization problem reads as follows:

$$\begin{aligned}\min f(\boldsymbol{x}) = &-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) \\ &- \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \qquad (18)\end{aligned}$$

subject to

$$\sum_{i=1}^{n} x_i \le 0, \quad \|\boldsymbol{x}\|_2 - 5 \le 0, \quad \boldsymbol{x} \in [-5, 10]^n \qquad (19)$$

To consider a small-scale model (18)–(19) for the experimental study, we set $n = 100$. The optimization problem has a global minimum of $f(\boldsymbol{x}^\star) = 0$ at $\boldsymbol{x}^\star = \mathbf{0}$.

4.1.3 Dynamic economic dispatch

The DED problem has become one of the most important and frequently studied nonlinear constrained optimization problems in practice and serves as the third use case in this work. The general task is to determine the scheduling of committed generators over a certain operating period while ensuring the load demand under several technical constraints. In other words, the objective of DED is to minimize the total cost of electricity generation. The corresponding optimization problem with valve point effects (Attaviriyanupap et al. 2002; Ravikumar Pandi and Panigrahi 2011; Yadav and Deep 2014; Balamurugan and Subramanian 2007; Santra et al. 2020) can be formulated as follows:

$$\min f(\boldsymbol{P}) = \sum_{t=1}^{T}\sum_{i=1}^{N_\mathrm{G}} a_i P_{it}^2 + b_i P_{it} + c_i + \left| e_i \sin\left(f_i\left(P_{it}^{\min} - P_{it}\right)\right)\right| \qquad (20)$$

subject to

$$\sum_{i=1}^{N_\mathrm{G}} P_{it} = P_{\mathrm{D}t}, \quad \forall t \in \mathcal{T}, \qquad (21)$$

$$P_i^{\min} \le P_{it} \le P_i^{\max}, \quad \forall i \in \mathcal{I}, \quad \forall t \in \mathcal{T}, \qquad (22)$$

$$P_{it} - P_{i(t-1)} \le \mathrm{UR}_i, \quad \forall i \in \mathcal{I}, \quad P_{i(t-1)} - P_{it} \le \mathrm{DR}_i, \quad \forall i \in \mathcal{I} \qquad (23)$$

with $\mathcal{I} = \{1, \ldots, N_\mathrm{G}\}$ and $\mathcal{T} = \{1, \ldots, T\}$. The objective function (20) is represented by a nonsmooth formulation consisting of both a quadratic polynomial and an absolute sinusoidal term. Equalities and inequalities are specified by power balances (21), generator capacities (22) and unit ramp rate limits (23). We note that for the typical 24-h operation of a 5-unit, 10-unit and 30-unit system (i.e., $N_\mathrm{G} = 5, 10, 30$), the corresponding problem sizes are 120, 240 and 720, respectively. Therefore, all three DED problems are classified here as medium-scaled problems. At this point, we would like to mention that we use the 5-unit DED problem for analyzing the BO-IPOPT method and the 10-unit and 30-unit DED problem for the solver comparison, as the former one still has a reasonable problem size to investigate different parameter settings when using the hybrid method. The scenario input data for the load demand $P_{\mathrm{D}t}$ is given in Appendix D.

4.1.4 Renewable steam generation

The fourth use case is the operational optimization of an industrial energy conversion system for RSG, recently proposed in Walden et al. (2023). The resulting optimization problem aims to minimize the operational costs while taking fluctuating wind energy and electricity prices into account. The discretized nonlinear nonconvex constrained optimization problem is written as follows:

$$\min \; f(\boldsymbol{P}_\mathrm{grid}) = \sum_{k=1}^{n} P^k_\mathrm{grid} g^k_\mathrm{grid} \Delta t \tag{24}$$

subject to

$$P^k_\mathrm{grid} + P^k_\mathrm{WT} = 3F_\mathrm{HTHP}\big(T^k_\mathrm{I}, \dot{m}^k_\mathrm{I}, T^k_\mathrm{III}, R^k\big), \tag{25}$$

$$T^k_\mathrm{II} = F_\mathrm{HTHX}\big(T^k_\mathrm{I}, \dot{m}^k_\mathrm{I}, T^k_\mathrm{III}, R^k\big), \quad T^k_\mathrm{IV} = F_\mathrm{LTHX}\big(T^k_\mathrm{I}, \dot{m}^k_\mathrm{I}, T^k_\mathrm{III}, R^k\big), \tag{26}$$

$$T^k_2 = T^k_\mathrm{II}\beta^k_1 + T^k_1\big(1-\beta^k_1\big), \quad T^k_\mathrm{I} = T^k_3\beta^k_2 + T^k_4\big(1-\beta^k_2\big), \tag{27}$$

$$T^k_2 = 201.92 + \tfrac{1819.32}{3\dot{m}^k_\mathrm{II}}, \quad T^k_3 = 196.3 - \tfrac{188.4}{3\dot{m}^k_\mathrm{I}}, \tag{28}$$

$$\dot{Q}^k_\mathrm{s,ch} = 3\dot{m}^k_\mathrm{II}c_{p,\mathrm{f}}\big(T^k_\mathrm{II} - T^k_1\big)\big(1-\beta^k_1\big), \quad \dot{Q}^k_\mathrm{s,dch} = 3\dot{m}^k_\mathrm{I}c_{p,\mathrm{f}}\big(T^k_4 - T^k_3\big)\big(1-\beta^k_2\big), \tag{29}$$

$$\dot{Q}^k_\mathrm{s,ch}\dot{Q}^k_\mathrm{s,dch} \le \gamma, \tag{30}$$

$$T^k_1 = T^k_\mathrm{II} - \epsilon_\mathrm{ch}\big(T^k_\mathrm{II} - T^{k-1}_\mathrm{s}\big), \quad T^k_4 = T^k_3 - \epsilon_\mathrm{dch}\big(T^k_3 - T^{k-1}_\mathrm{s}\big), \tag{31}$$

$$T^k_\mathrm{s} = T^{k-1}_\mathrm{s} + \frac{\dot{Q}^k_\mathrm{s,ch} - \dot{Q}^k_\mathrm{s,dch}}{m_\mathrm{s}c_\mathrm{p,s}}\Delta t, \quad T^0_\mathrm{s} = \widetilde{T}_0, \quad T^n_\mathrm{s} = T^0_\mathrm{s}, \tag{32}$$

$$T^k_\mathrm{I} \in [177, 250], \quad \dot{m}^k_\mathrm{I} \in [5, 16], \quad T^k_\mathrm{III} \in [60, 100], \quad R^k \in [0.8, 1.53] \tag{33}$$

for $k = 1, \ldots, n$ and with time step size Δt. The objective function (24) is linear and depends on the grid power consumed by the high-temperature heat pump. Most of the constraints are nonlinear (cf. (25)–(30)) due to incorporated nonlinear component (surrogate) models, bypass modeling as well as charging and discharging heat flows. Linear constraints (31)–(32) belong to the thermal energy storage effectiveness model and the boundary conditions for the storage temperature. We emphasize that a typical 1-day system operation with an hourly time step size leads to a problem with a dimension of 408, which can thus be classified as a medium-scale problem. The input data (wind turbine energy P_{WT}^{k} and electricity price g_{grid}^{k}) considered for the scenario is presented in Appendix D.

4.2 Experimental results and analysis

The following study of the BO-IPOPT method is mainly divided into two parts. First, we examine the aspects relevant to the application of the proposed hybrid method: (i) selecting new candidates for the GPR, (ii) dealing with constraints in the BO part, and (iii) choosing optimal values for the hyperparameters involved. Second, we evaluate MS-IPOPT, BOwLS, and BO-IPOPT by comparing their accuracy, computational costs, scalability, and robustness at the hands of the aforementioned benchmark problems.

4.2.1 Evaluating EI: discrete set of candidates versus continuous space

As mentioned in Sect. 3, it is important to effectively select new candidates for the GPR at each outer iteration in the BO part of our proposed hybrid method. In particular, the next sampling point can be determined in two ways: by evaluating the EI-based acquisition function either over a discrete set of candidates or, as is often done in the literature, over the entire continuous space using a numerical optimization technique. For this reason, we compare both techniques and their impact on the accuracy and CPU time of BO-IPOPT. Although recent studies have demonstrated the potential of alternative formulations like LogEI (Ament et al. 2023) to enhance the performance of EI in high-dimensional and numerically challenging landscapes, as well as the promise of compositional acquisition functions (Grosnit et al. 2021) in improving optimization strategies, our focus remains on using the standard EI due to its proven effectiveness. It should be emphasized that the comparison is only performed for the simple test problem (9)–(17) because the problem size of the other benchmark problems is too large, which would lead to extremely high CPU times (when considering a continuous space).

For the experimental implementation, we use the following setup: the constraints in BO are handled via (2) and the hyperparameters are fixed to $N_0 = 10$, $N_{\mathrm{c}} = 500$, $N_{\mathrm{bc}} = 1$, $\xi = 0.01$,[1] $\alpha = 0.1$, $l = 100$, $\boldsymbol{\tau} = \mathbf{100}$ and $\boldsymbol{\delta} = \mathbf{0}$ (since no equality constraints are included). The selected values for α and l are larger than the default values[2] provided by the SKLEARN library, which therefore enables the BO's GP model to explore the entire input space on a global scale, taking into account smooth variations and ignoring details of the function to be optimized. Then, IPOPT focuses on the regions of interest provided by the BO part, ensuring that all constraints are satisfied at each outer iteration K. Although the benchmark problems considered in this work

[1] This value corresponds to the default value of BO provided as a package by the SCIKIT-OPTIMIZE library (The Scikit-Optimize Contributors 2020).

[2] The default values in the SKLEARN library are $\alpha = 10^{-10}$ and $l = 1$.

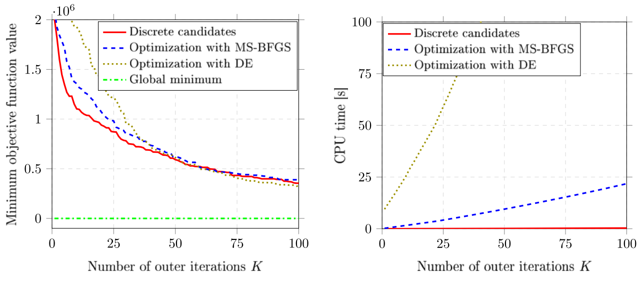

Fig. 1 Optimization results for the simple test problem (9)–(17) using the original BO: comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 100 trials by evaluating the EI over a discrete set of candidates and through a continuous optimization procedure (via MS-BFGS and DE) as a function of the number of outer iterations K. The global minimum $f(\boldsymbol{x}^{\star}) \approx -216.65649$ determined by the BARON solver is also visualized on the left. Particular caution must be taken when evaluating the visualization in the left figure, as the BO solutions have significantly larger minimum objective function values than the global minimum. Therefore, we only display the results up to $K = 100$

are not noisy, α helps ensure the covariance matrix is well-conditioned, making the matrix inversion more stable and preventing numerical issues such as singularity. In addition, this parameter helps avoid overfitting or underfitting by ensuring that the GP model remains sufficiently flexible. In cases with small number of data and high dimensions, α provides additional stability and robustness by preventing the covariance matrix from becoming ill-conditioned. In our study, we adopt a different approach compared to traditional BO methodologies. Specifically, while conventional BO often utilizes covariance kernels with length scales l that are optimized during the process, we employ isotropic covariance kernels with a fixed length scale. This choice focuses the optimization on providing a global direction by BO rather than delving into fine details. By fixing the length scale l, we avoid the additional computational cost of optimizing this parameter, thereby streamlining the BO process and enhancing its scalability. The selected numbers for N_0 and N_c are used to avoid unreasonable increases in CPU time due to the correlation matrix inversion and acquisition function evaluation/optimization at each outer iteration, respectively.

For the evaluation of the EI function over the GPR through a continuous optimization procedure, i.e., $\boldsymbol{x}_{z+1} = \arg\max_{\boldsymbol{x}\in D_j} \mathrm{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$, we use two common solvers, namely the MS-L-BFGS-B (multistart BFGS with limited memory and boundaries) algorithm and the differential evolution (DE) algorithm both from SciPy. L-BFGS-B is restarted 20 times to avoid local optima, and DE uses a population size of 100, a maximum number of 200 iterations, a relative tolerance for convergence of 0.01, a recombination of 0.7, and a mutation in the interval [0.5,1]. Below, we first examine the effect of EI treatment on BO itself, then on the BO-IPOPT method. Since BO and BO-IPOPT are based on certain randomness, the numerical experiments are repeated 100 times with outer iterations $K = 300$ to ensure a more thorough evaluation.

The optimization results for the simple test problem using the original BO method are shown in Fig. 1, where the graphics on the left and right compare the averaged minimum objective function value and the averaged CPU time as a function of the number of outer iterations K, respectively. As generally expected, the results show

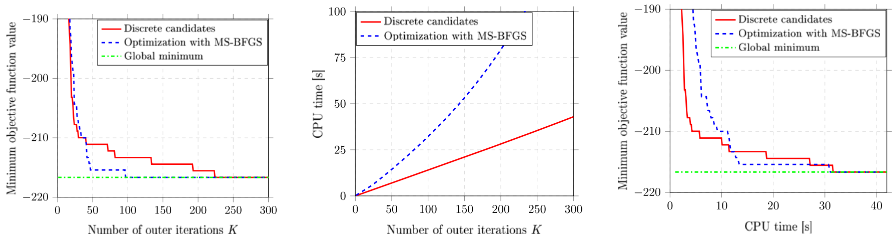

Fig. 2 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials by evaluating the EI over a discrete set of candidates and through a continuous optimization procedure (via MS-BFGS) as a function of the number of outer iterations K. The global minimum $f(\boldsymbol{x}^{\star}) \approx -216.65649$ determined by the BARON solver is also visualized

that evaluating the EI over the entire continuous space can lead to a slightly faster convergence (fewer iterations K required) of BO toward the global minimum than simply evaluating EI over a discrete set of candidates. This fact applies to the optimization with DE, but not for MS-L-BFGS-B, which only indicates a similar convergence to the evaluation of the acquisition function over a discrete set of candidates. In other words, optimizing EI using DE produces more accurate results than using MS-L-BFGS-B, because the local multistart solver is often trapped in local minima and it consequently has a negative influence on BO. Although DE has a higher probability of finding the global optimum of the acquisition function than simply evaluating it over a discrete set of candidates, the method naturally requires drastically higher computational costs because of the many function evaluations. The multistart solver MS-L-BFGS-B is also linked to high computational effort. In addition, we would like to point out that the BO solutions (see all three curves on the left-hand side in Fig. 1) are far from the global minimum and have significantly larger minimum objective function values. This happens because the objective function includes penalty terms (2) that are not all fulfilled in the results shown. This issue (extremely slow convergence rate) can be tackled by using the hybrid BO-IPOPT method, as presented next.

For the BO-IPOPT, we compare the evaluation of EI over a discrete set of candidates with the evaluation of it over the entire continuous space considering only the MS-L-BFGS-B (cf. Fig. 2), because the computational complexity of DE is not practical. On the one hand, Fig. 2 demonstrates that the hybrid use (combination of BO with local solver) enables BO-IPOPT to converge faster to the global minimum (while satisfying the given constraints without changing any hyperparameters) than the original BO, whose results are shown in Fig. 1. On the other hand, the evaluation of EI over the entire continuous space within BO-IPOPT leads to significantly higher computational costs with a simultaneously improved convergence rate (fewer iterations K required) compared to the evaluation of the acquisition function over a discrete set of candidates. For a better comparison, we also visualize the solver performances on the right in Fig. 2, i.e., CPU time versus minimum objective function value, which represents an almost identical efficiency of the two approaches. In this context, however, it should be noted that the CPU time indicates a nearly linear trend with evaluating EI over a discrete set of candidates, while evaluating EI through a continuous optimization procedure causes a superlinear-based increase in CPU time as the number of outer iterations increases. Increasing the problem dimension (and thus also the EI dimension) will have a greater negative effect on the performance in EI evaluation through a continuous optimization procedure than in EI evaluation over a discrete set of candidates. It should also be emphasized that in this work, we consider optimization problems, where the computational cost of optimizing the acquisition function over a continuous space is not negligible compared to evaluating the objective function and constraints.

Overall, the evaluation of EI in BO-IPOPT suggests a better trade-off between convergence rate (toward the global optimum) and computational costs compared to optimizing the acquisition function, especially when considering medium- or large-scale problems. For this reason, we simply evaluate EI for all further experiments in order to select the best candidates at each outer iteration of the hybrid method.

4.2.2 Constraint handling in BO

Another relevant issue within BO-IPOPT that needs to be examined is how the equality and inequality constraints (included in the respective optimization problem) are effectively handled in the BO part. We consider here two frequently used approaches (see Sect. 2.1), firstly by simply augmenting the objective function with the constraints as penalty terms (2), and secondly by the slack AL method (3), in which a sequence of simpler unconstrained subproblems must be solved (see Appendix A). To avoid a significant increase in the CPU time (see Appendix E), we build only one GP model and not independent surrogate models. However, as already mentioned in Sect. 2.3, both constraint handling techniques have a direct influence on the initial GPR (resulting from the hybrid construction), which is built on the first function evaluation of the initialization points, and then indirectly affect the new selected points at each K using the EI. Clearly, both techniques does not affect the computational costs.

In the following, we compare both techniques for BO-IPOPT in terms of accuracy for the simple test problem (9)–(17), the constrained Ackley function (18)–(19) and the DED problem (20)–(23) with 5 units (input data is given in Appendix D). In this context, we emphasize again that the objective function (20) of DED is nonsmooth (not differentiable everywhere), so classical gradient-based NLP solvers such as IPOPT cannot be applied directly. To overcome this difficulty and obtain a suitable smooth optimization problem that can be solved with a gradient-based solver, the cost function must be reformulated. In doing so, we follow the approach in Pan et al. (2018) by applying auxiliary variables in (20) and adding inequality constraints, which are then converted into equality constraints using slack variables. By introducing these additional variables required to achieve a differentiable formulation, the original problem size is enlarged, i.e., the 5-unit DED problem now contains 480 variables.

In contrast, the BO framework in the hybrid approach faces no problem with the nondifferentiable formulation and it does not thus require any reformulation, so the original problem with 120 variables is considered in the BO part. The latter procedure (using two different formulations for BO and the local search in BO-IPOPT) is initially not conclusive from a practical point of view, as the inital starting points for the local solver contain missing information (no slack variables), which is why we address this issue in more detail in Appendix H.

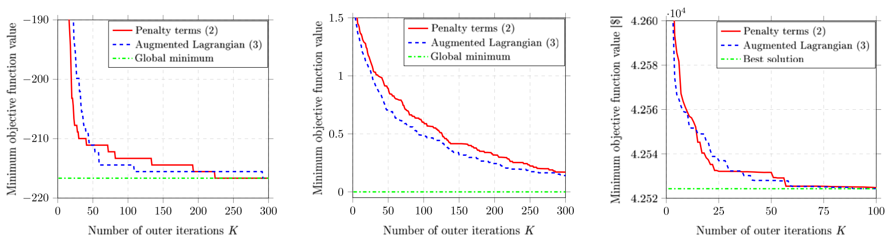

Fig. 3 Optimization results for the simple test problem (9)–(17) (**left**), the constrained Ackley function (18)–(19) (**middle**) and the DED problem with 5 units (20)–(23) (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 trials for the two test cases and 50 trials for the DED problem considering the two constraint handling techniques, i.e., (2) versus (3), as a function of the number of outer iterations K. We display the results for the DED problem only up to $K = 100$ because the best solution is quickly reached on average

We would also like to mention that the global minimum of the 5-unit DED problem can not be computed by a commercial solver like BARON in a reasonable time (number of variables is too large), so we apply MS-IPOPT with 10,000 different starting points to explore the entire parameter space. The best result $f(\boldsymbol{x}^\star) \approx 42{,}524.2785$ is considered as an estimate of the global minimum.

Again, the experiments are repeated 100 times for the two test problems and 50 for the DED problem with outer iterations $K = 300$ to average the stochastic nature of the optimizer, and the same hyperparameters are used as specified in Sect. 4.2.1. The reduced number of trials for the DED is applied due to the larger problem dimension, which leads to a higher computational effort in total.

The optimization results for the three problems are shown in Fig. 3. As displayed in this figure, both techniques show similar convergence rate. However, the application of the AL method (3) has a crucial advantage: the hyperparameters are selected according to predefined rules (see Appendix A), so that this technique can be understood as parameter-free. In contrast, (2) requires user-defined penalty weight parameters, which are generally unknown and problem-dependent. More precisely, setting values is not straightforward because the hyperparameters are vectors that can have different entries depending on the type of constraints and how strongly each constraint is enforced during the optimization process. Although constraint handling mainly affects the inital GPR, choosing good values for the initial surrogate model may improve the optimization process, as the new selected points at each K are influenced as well.

It is important to underline that AL has been combined in such a way with BO and IPOPT that it does not face the limitations regarding high dimensions and large number of constraints mentioned in Sect. 2.1. The GP model in BO-IPOPT is not required to provide a highly detailed approximation of the objective function and constraints across the entire space. Instead, it serves as a global guide, identifying regions that are likely to contain better solutions based on the trade-off between exploration and exploitation. Once a region is selected, IPOPT takes over as a powerful local solver, leveraging gradient information to efficiently navigate the local landscape and optimize the objective function considering the defined constraints. In contrast, Picheny et al. (2016) use multiple surrogate models-typically one for the objective function and separate ones for the constraints. However, this approach does not include a local solver like IPOPT, meaning that the global search must rely solely on the surrogate models to approximate and satisfy constraints, which can lead to inefficiencies, particularly in high-dimensional spaces or when the number of constraints becomes large. Overall, we consider AL as superior approach for dealing with equality and inequality constraints in BO-IPOPT and use this parameter-free technique for all further experiments.

4.2.3 Parameter study and setting

As already mentioned, our proposed BO-IPOPT method possesses several hyperparameters that generally affect the global guidance offered by the BO component and the computational effort of the hybrid method. For this reason, we are interested in finding good values for these parameters to reach a compromise between high convergence speed and low computational costs, providing an effective guidance by the BO and avoiding scalability issues in problems of higher dimensions. Although they naturally strongly depend on the underlying optimization problem, we experimentally study their influence on the hybrid method to determine a reasonable parameter setting and fix them without learning them during the optimization process, as usually done in the traditional BO methodologies.

The hyperparameters investigated within the hybrid method are the following: the length scale l, the regularization term α, the exploration term ξ, the number of initialization points N_0, the number of candidates N_c considered in EI and the number of best candidates N_bc at each outer iteration. It should be clear that the variation of l, α and ξ have no direct influence on the computational effort in an outer iteration in BO-IPOPT (without simultaneously changing N_0, N_c, N_bc), so we simply omit the CPU time comparison for these three parameters.

The parameter study is carried out for the simple test problem, the constrained Ackley function and the 5-units DED problem, while the other medium-scaled problems are excluded due to the high computational effort. Of course, the number of benchmark problems is small to guarantee an optimal parameter selection, nevertheless, it is important to reduce/fix the number of still free design parameters in BO-IPOPT in an appropriate manner. For the experiments, we again use the hyperparameters as in Sect. 4.2.1 and consider this setting as the base case. More precisely, we analyze each parameter individually, while the others are fixed as in the base case. Once again, the experiments are repeated 100 times for the test problems and 50 times for the DED problem to average the method's randomness.

Note that the variation of the involved parameters does not strongly affect the performance of BO-IPOPT in the simple test problem (cf. Figs. 22, 23, 24 and 25 in Appendix G.1) because of the low problem dimension and complexity. Therefore, we focus on the other two optimization problems, where the hyperparameters have an influence on the performance of the optimizer, see Figs. 4, 5, 6, 7, 8 and 9. In high-dimensional problems, the impact of hyperparameter selection becomes more pronounced due to the increased complexity of the search space and the limitations of surrogate models in capturing intricate relationships across many dimensions, making the choice of each parameter very important. Some of these parameters also have

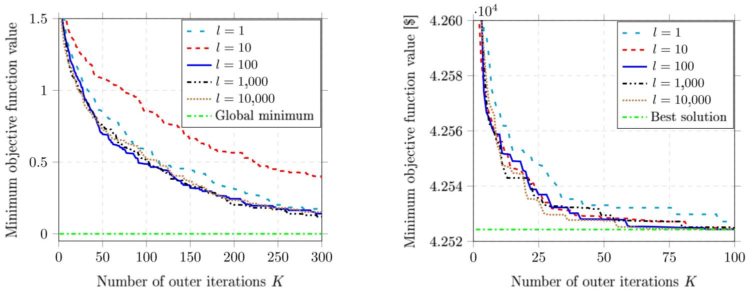

Fig. 4 Optimization results for the constrained Ackley function (18)–(19) (**left**) and the DED problem (20)–(23) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different length scales l as a function of the number of outer iterations K

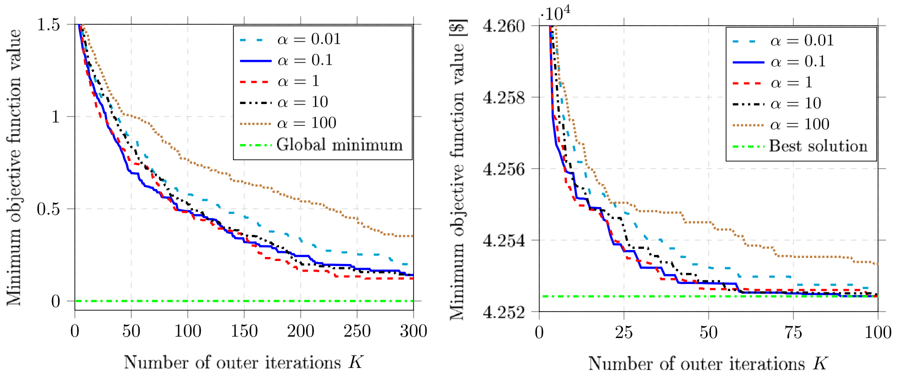

Fig. 5 Optimization results for the constrained Ackley function (18)–(19) (**left**) and the DED problem (20)–(23) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different regularization terms α as a function of the number of outer iterations K

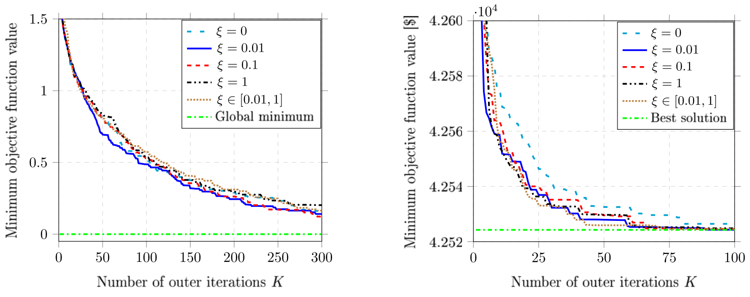

Fig. 6 Optimization results for the constrained Ackley function (18)–(19) (**left**) and the DED problem (20)–(23) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different exploration terms ξ as a function of the number of outer iterations K

a direct impact on the CPU costs of the optimization process, which becomes very substantial as the dimension increases.

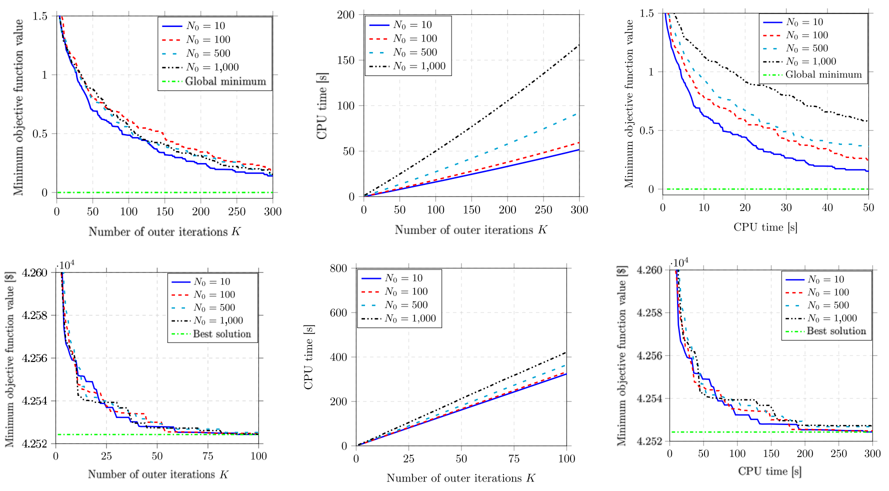

Fig. 7 Optimization results for the the constrained Ackley function (18)–(19) (**top**) and the DED problem (20)–(23) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 and 50 trials for different numbers of initialization points N_0 as a function of the number of outer iterations K

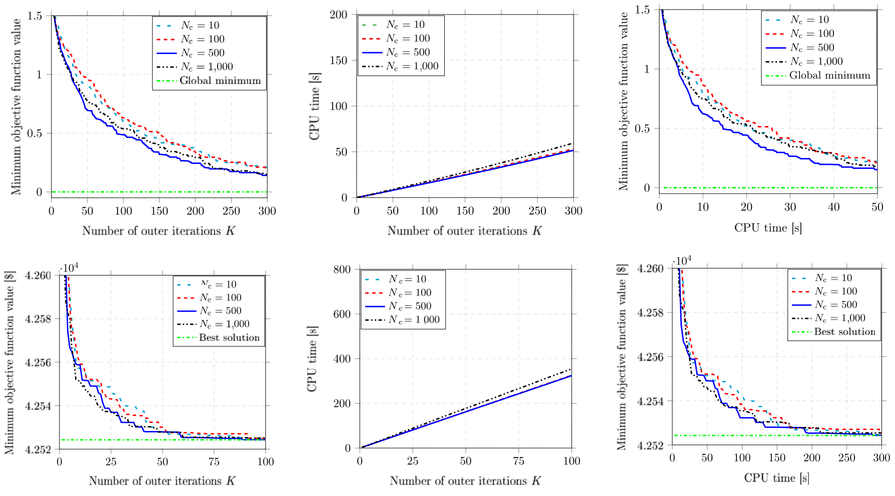

Fig. 8 Optimization results for the the constrained Ackley function (18)–(19) (**top**) and the DED problem (20)–(23) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 and 50 trials for different numbers of candidates N_c as a function of the number of outer iterations K

First, BO-IPOPT is examined with different values for the length scale l, whereby the parameter influence on the accuracy for both problems is visualized in Fig. 4. Small values for l lead the optimizer to a slower convergence (more iterations K required) compared to larger values. In other words, a larger length scale implies smoother variations, while smaller values allow for rapid variations in the function

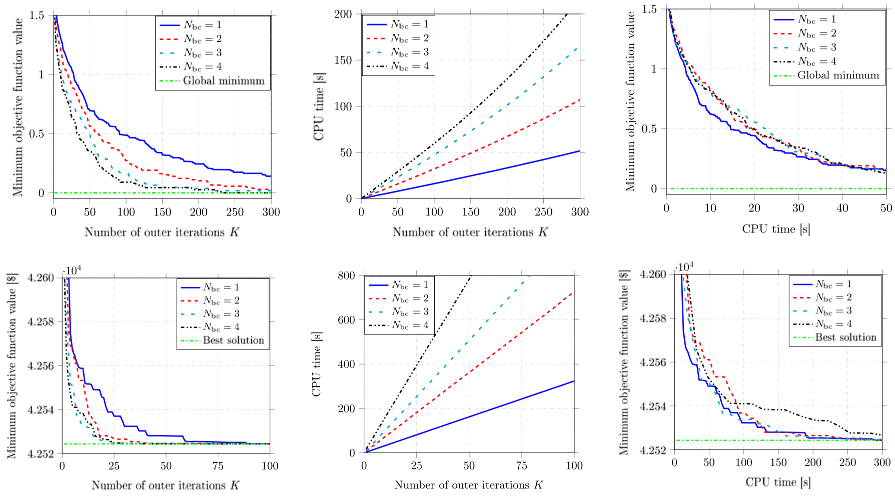

Fig. 9 Optimization results for the constrained Ackley function (18)–(19) (**top**) and the DED problem (20)–(23) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 and 50 trials for different numbers of best candidates N_{bc} as a function of the number of outer iterations K

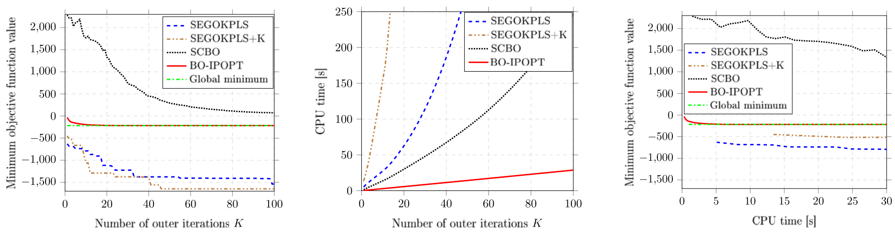

Fig. 10 Optimization results for the simple test problem (9)–(17): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials using SCBO, SEGOKPLS, SEGOKPLS+K and BO-IPOPT as a function of the number of outer iterations K

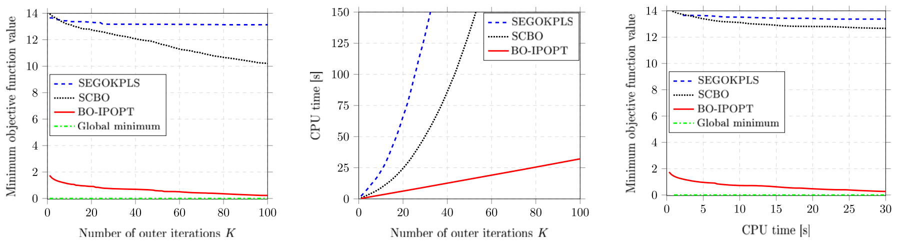

Fig. 11 Optimization results for the constrained Ackley function (18)–(19): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials using SCBO, SEGOKPLS and BO-IPOPT as a function of the number of outer iterations K

being modeled. Consequently, when using small l, the optimizer gets stuck in local minima as it considers small details in its exploration and often converges more slowly.

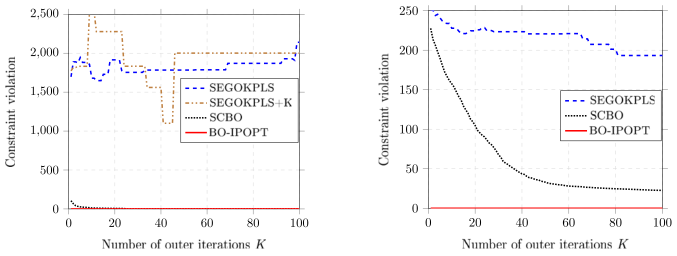

Fig. 12 Optimization results for the simple test problem (9)–(17) (**left**) and the constrained Ackley function (18)–(19) (**right**): comparison of the constraint violation over 100 trials using SCBO, SEGOKPLS, SEGOKPLS+K and BO-IPOPT as a function of the number of outer iterations K

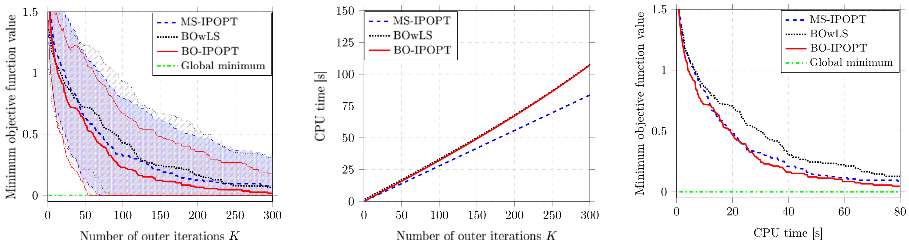

Fig. 13 Optimization results for the constrained Ackley function (18)–(19): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

Based on these results, the choice $l = 100$ offers the best compromise in convergence for BO-IPOPT, since greater values show similar convergence and, moreover, too large values are associated with higher function smoothing, which may have a more negative impact on the optimizer. As a result, the GP model in the BO part explores the entire input space on a global scale, while the IPOPT focuses on the regions of interest provided by the BO part.

The results of BO-IPOPT for various regularization terms α are shown in Fig. 5. As seen for both optimization problems, large values for α underfit the data and oversimplify the GP model, resulting in poor surrogate model performance. Consequently, this leads to a slow convergence rate of the hybrid method. Conversely, if the values are too small, overfitting may occur and yield poor generalization to unseen data in the GP model with slow convergence. For this reason, neither too small nor too large regularization terms are considered sufficient for BO-IPOPT and we suggest using $\alpha = 0.1$.

The third parameter investigated is the exploration term, whose results are shown in Fig. 6. Obviously, the choice of ξ does not have a strong influence on the BO-IPOPT's convergence rate for the problems considered. This is due to the local solver, which weakens the effect of ξ in the BO part without requiring precise values for this parameter. However, no exploration (i.e., $\xi = 0$) leads the optimizer to a slightly slower convergence in the DED problem. Although good performance is achieved with constant values, we propose to use a monotonically decreasing sequence, starting with $\xi = 1$ to a final value $\xi = 0.01$, to encourage exploration in the first few iterations and then focus more on exploitation as the optimization progresses.

Moving on to the remaining parameters N_0, N_c and N_bc, which influence both the accuracy and the computational costs of BO-IPOPT, see Figs. 7, 8 and 9. The solver performances (CPU time vs. minimum objective function value) are also presented for a better comparison.

As seen in Fig. 7 (middle), the number of initialization points N_0 influences the computational effort considerably, especially for very large values $N_0 \geq 500$. In contrast, the corresponding convergence rate on the left in Fig. 7 is not affected significantly. Larger N_0 greatly increase the CPU time of BO-IPOPT because the initial GPR starts with a large number of sampling points, and inverting the correlation matrix at each outer iteration thus becomes computationally intensive as more samples are added. The value $N_0 = 10$ represents a good trade-off (see on the right in Fig. 7) between faster convergence and low computational costs, as it allows the initial GPR to explore the entire space and keeps the optimizer's CPU time low compared to higher values N_0, as more samples are available.

The results of BO-IPOPT with different number of candidates N_c are illustrated in Fig. 8. An increase in N_c is accompanied by an equally increased number of evaluations of the acquisition function at each outer iteration, but this only leads to slightly higher computational costs (see middle figure). At the same time, a larger value for N_c increases the probability of finding a suitable candidate, which can generally improve the surrogate model and, thus, the convergence rate. Nevertheless, too high values (e.g., $N_\mathrm{c} = 1000$) may raise the CPU time without improving the solution accuracy compared to lower values (e.g., $N_\mathrm{c} = 500$). Therefore, setting $N_\mathrm{c} = 500$ in BO-IPOPT is considered a reasonable value (see on the right in Fig. 8) that provides a balance between convergence and CPU time.

The last parameter N_bc studied in BO-IPOPT has the greatest influence on the convergence rate and CPU time of the optimizer as shown in Fig. 9. Each additional best candidate used (at each outer iteration) increases the probability of finding the global minimum faster. Still, it simultaneously leads to a higher computational effort due to the additional IPOPT searches. In general, the solver performance for both problems remains unaffected (see on the right in Fig. 9). Contrary to our expectations, a larger N_bc has principally no clear benefit since the ratio of convergence rate and CPU time is roughly linear.[3] However, we recommend using $N_\mathrm{bc} > 1$ because local searches are well-suited for parallel computing due to their independent nature, so BO-IPOPT generally has a higher probability (with an additional number of best candidates) of finding the global optimum in less CPU time (due to parallel searches) as more candidates are added to the GPR at each outer iteration. The efficient parallelization of BO-IPOPT (also for parts in BO itself) will be the subject of future work to speed up the algorithm (and thus the CPU time required). Overall, we set $N_\mathrm{bc} = 2$ in this work because, on the one hand, the probability of determining the global optimum is

[3] Increasing N_bc at fixed K has a similar performance as vice versa, e.g., for the Ackley problem, $N_\mathrm{bc} = 3$, $K = 100$ and $N_\mathrm{bc} = 1$, $K = 300$ (corresponds to a total number of 300 local searches) achieve almost the same minimum objective function value in almost the same CPU time.

higher compared to $N_{\mathrm{bc}} = 1$ and on the other hand, BO-IPOPT is not parallelized yet so greater values than two lead to extremely high CPU times, especially for larger problems as those presented in Sects. 4.1.3 and 4.1.4.

We recall that the parameter study conducted is obviously based on a small number of optimization problems, leaving a detailed quantitative evaluation for future work.

4.2.4 Evaluation: SCBO, SEGOKPLS, SEGOKPLS+K versus BO-IPOPT

This section compares our fine-tuned hybrid method with the effective BO methods "SCBO", "SEGOKPLS" and "SEGOKPLS+K", developed for higher dimensions and already introduced in Sect. 1, using the simple test problem and the constrained Ackley function. The other benchmark problems are not considered in this comparison since they would lead to remarkably high CPU times.

For the BO-IPOPT, the following parameters are defined: $l = 100$, $\alpha = 0.1$, monotone sequence $\xi \in [0.01, 1]$, $N_0 = 10$, $N_{\mathrm{c}} = 500$, and $N_{\mathrm{bc}} = 2$. The constraints are handled in the BO part of BO-IPOPT using the AL framework (3), and the EI is evaluated over a discrete set of candidates. For a fair comparison between the methods, we consider the same values for N_0, N_{c} and N_{bc} in all the optimizers. SCBO is implemented using BoTorch (Balandat et al. 2020). It considers a Matern kernel and a Thompson sampling extended to constrained optimization for selecting new points for the GPR at each outer iteration K. The KPLS and KPLSK in SEGOKPLS and SEGOKPLS+K are implemented using the Smt library (Saves et al. 2024). The number of principal components is set to 2 for the simple test problem and 9 for the constrained Ackley function. A higher value than 9 is not allowed since we start with 10 initialization points in the surrogate model. Moreover, the number of principal components in the constrained Ackley function is higher than the one in the simple test problem because of the higher dimension of the problem. The acquisition function, consisting of the EI criterion weighted by the probability of satisfying each constraint, is maximized by MS-L-BFGS-B and is restarted 10 times without considering any parallel computing since we do not utilize it in BO-IPOPT either. The remaining hyperparameters for SCBO, SEGOKPLS, and SEGOKPLS+K are set to their default values.

As seen in Figs. 10 and 11, BO-IPOPT achieves better convergence rate compared to the other BO methods with simultaneously lower computational costs, leading to much better performance (lower minimum objective function values at the same CPU time). This becomes even more significant in the constrained Ackley function, where the dimension of the problem increases. It is worth noticing that the SEGOKPLS+K has not been tested in the constrained Ackley function. The reason for that is that a high number of points are required to build the surrogate model after some outer iterations, which is not possible with our setup. To solve this issue, we must drastically increase either N_0 or N_{bc}, but this would force a deviation from the setup of the other optimizers. The higher CPU time of SCBO, SEGOKPLS, and SEGOPLS(+K) is attributed to the use of independent surrogate models for the constraints and the objective function, the learning of the hyperparameters throughout the optimization process, and the solution of an optimization problem over the continuous space for selecting new points for the surrogate model at each outer iteration K. As mentioned, BO-IPOPT builds only

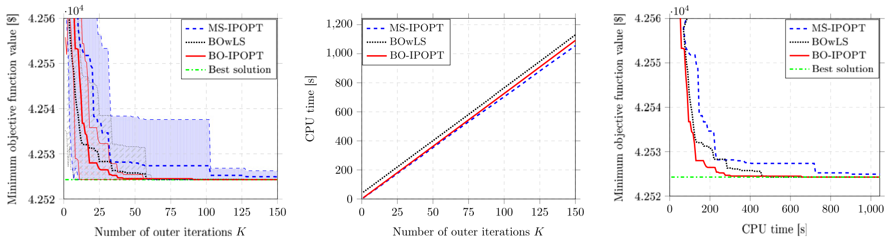

Fig. 14 Optimization results for the DED problem (20)–(23) with 5 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

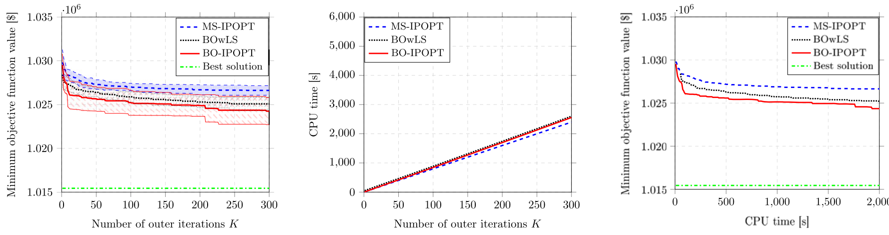

Fig. 15 Optimization results for the DED problem (20)–(23) with 10 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

one surrogate model for the objective function and the constraints and evaluates the EI over a discrete set of candidates, leading to a linear trend in the CPU time. The minimum objective function value over the number of function evaluations is not directly compared between the methods. The BO part of BO-IPOPT follows the same settings as the BO approaches, with two function evaluations at each outer iteration, ensuring a fair basis for comparison in terms of the BO search process. However, the hybrid method has additional IPOPT function evaluations, which are significantly faster that the BO function evaluations (i.e., BO-IPOPT has smaller CPU time than the other BO methods at each K, even though the hybrid method performs more function evaluations at each K because of the local solver). To achieve the total number of function evaluations that BO-IPOPT performs at each iteration, SCBO, SEGOKPLS, and SEGOKPLS+K would require substantially higher computational resources. As a result, comparing total function evaluations directly between BO-IPOPT and other BO methods would be misleading, which is why such a comparison is not considered in this work. It should also be mentioned that only BO-IPOPT fulfills the constraints from the first outer iteration K (cf. Fig. 12). All the other methods face difficulties in

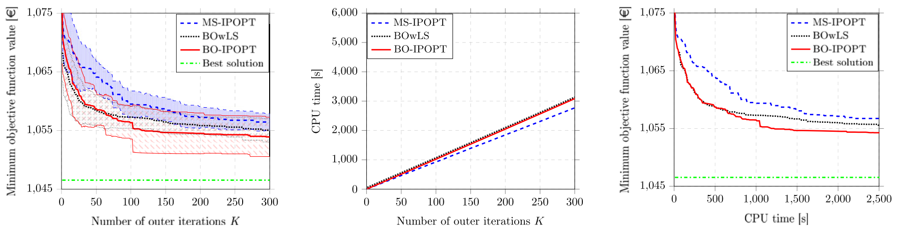

Fig. 16 Optimization results for the RSG problem (24)–(33): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

satisfying them, especially in the constrained Ackley function, where no BO method except for BO-IPOPT fulfills them. In the simple test problem, SCBO fulfills all the constraints at $K = 25$ (cf. Fig. 12). SEGOKPLS and SEGOKPLS+K yield results smaller than the global optimum in Fig. 10 on the left. This occurs because these solutions do not satisfy all constraints, as demonstrated in Fig. 12 on the left. In Fig. 11, SEGOKPLS also fails to fulfill all the constraints, but the minimum objective function value remains higher than the global optimum.

4.2.5 Evaluation: MS-IPOPT, BOwLS versus BO-IPOPT

In this section, we compare our proposed fine-tuned BO-IPOPT method with the competing methods, i.e., the classical random-based MS-IPOPT (cf. Algorithm 4) and BOwLS (cf. Algorithm 5) using the four benchmark problems already mentioned in Sect. 4.1 in terms of accuracy, CPU time, scalability and robustness.

To allow for a fair comparison between the hybrid approaches, we also apply IPOPT as a local solver for BOwLS since the conjugate gradient method from the SciPy package used in Gao et al. (2020) is not designed to handle constrained optimization problems. In the BO part of the hybrid methods, BO-IPOPT uses the AL framework (3) to include the underlying constraints, while BOwLS applies the penalty term formulation (2). In this context, we set $\boldsymbol{\delta} = \mathbf{0}$, $\boldsymbol{\tau} = \mathbf{100}$ and $\boldsymbol{\delta} = \boldsymbol{\tau} = \mathbf{100}$ for the two tests and the real-world problems, respectively. Further details on the explicit use of (2) within BOwLS are given in Appendix C.2. For reasons of consistency, the same parameters are used for both hybrid methods based on the study conducted in Sect. 4.2.3, i.e., $l = 100$, $\alpha = 0.1$, monotone sequence $\xi \in [0.01, 1]$, $N_0 = 10$, $N_\mathrm{c} = 500$, $N_\mathrm{bc} = 2$, and the same acquisition function. MS-IPOPT is also used with $N_\mathrm{bc} = 2$, in which two random points are generated and used as starting points for the local search at each outer iteration.

We would like to point out that the IPOPT solver is used either via Pyomo or Gams, depending on the use case. In other words, IPOPT is used via Pyomo for the two test problems, while Gams is used for the DED and RSG problems. The reason is the following: for the medium-scale optimization problems considered here, IPOPT

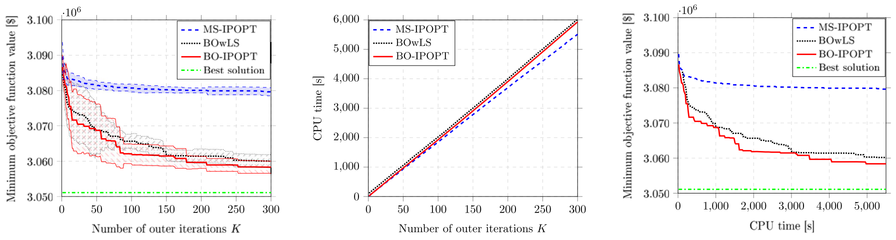

Fig. 17 Optimization results for the DED problem (20)–(23) with 30 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 20 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

in PYOMO sometimes fails to find a feasible point that is acceptable to the filter line search, and it therefore aborts the local search without providing any local optimum. To tackle this issue (multiple restarts of IPOPT in case of failure), we link PYTHON with GAMS via the GAMS API interface for the real-world problems, where IPOPT runs trouble-free.[4] To allow for a proper comparison between the optimization methods in terms of CPU time, each starting point in MS-IPOPT is also first generated in PYTHON and then solved using GAMS.

Before discussing the performance of the methods at hand of the benchmarks, we are providing at this point more information about the setup and the optimal solution of the real-world problems. For both the DED and RSG problem, we consider a typical one-day operation with hourly time step size, where the corresponding input data can be found in Appendix D. The resulting DED optimization problem (20)–(23) for 10 and 30 units contains 240 and 720 decision variables with the best known solutions being 1,015,438.967 \$ and 3,051,105.813 \$, respectively (Santra et al. 2020). As mentioned in Sect. 4.2.2, the cost function of the DED problem (20) must be reformulated so that gradient-based NLP solvers such as IPOPT can be applied. By introducing additional variables, the 5-, 10- and 30-unit DED problem now contain 480, 960 and 2880 variables, respectively, while the BO part in the hybrid approach considers the original problems with 120, 240 and 720 decision variables, respectively, to avoid unreasonable increase in CPU time.

Regarding the RSG system (24)–(33), the underlying 24-h optimal operation problem contains 408 decision variables, which makes it unfeasible for a commercial global optimization solver like BARON. For this reason, the application of MS-IPOPT with 10,000 randomly distributed starting points represents the best solution at 1046.53 € and is considered herein as the reference solution for the best possible minimum.

The optimization results of all three methods for the benchmark problems are shown in Figs. 13, 14, 15, 16 and 17, while the results for the simple test problem can be seen in Fig. 26 in Appendix I. Again, the averaged minimum objective function value,

[4] One reason for this could be that GAMS uses a newer IPOPT version, namely 3.13.3. In contrast, we could only use version 3.12 in PYOMO.

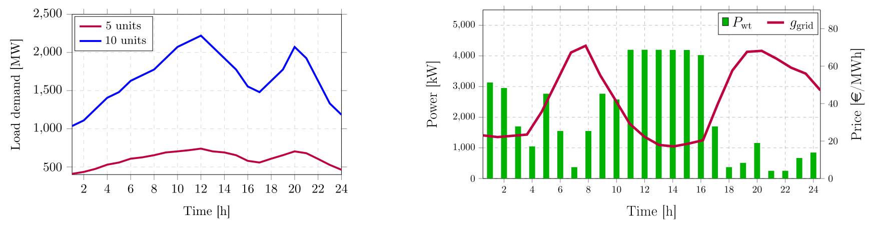

Fig. 18 Visualization of 24 h-scenario input data: load demand for the 5- and 10-unit DED problem (**left**) as well as P_{wt} (WT power output) and g_{grid} (electricity price) for the RSG problem (**right**)

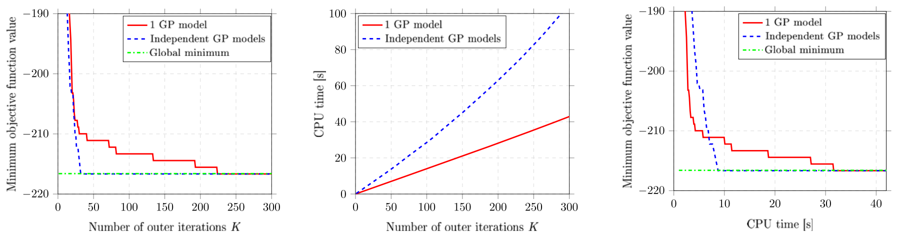

Fig. 19 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for single GP model and independent GP models as a function of the number of outer iterations K

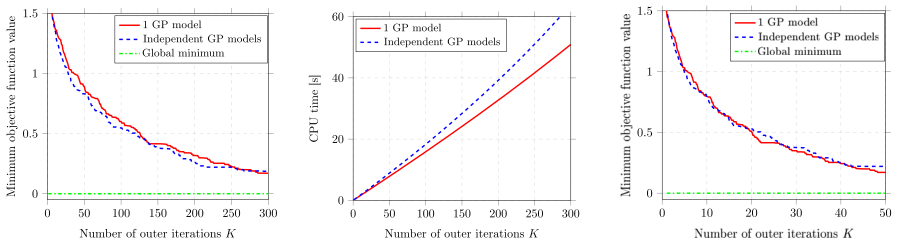

Fig. 20 Optimization results for the constrained Ackley function (18)–(19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for single GP model and independent GP models as a function of the number of outer iterations K

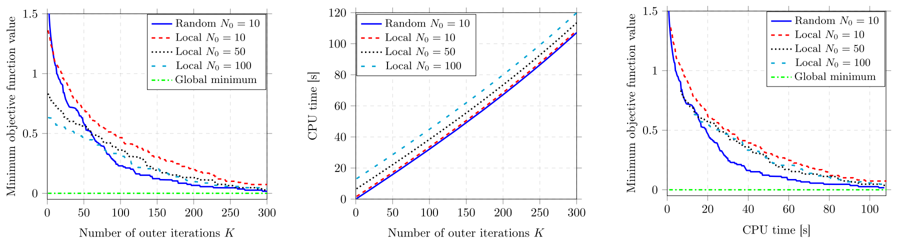

Fig. 21 Optimization results for the constrained Ackley function (18)–(19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for random and local initialization points N_0 as a function of the number of outer iterations K

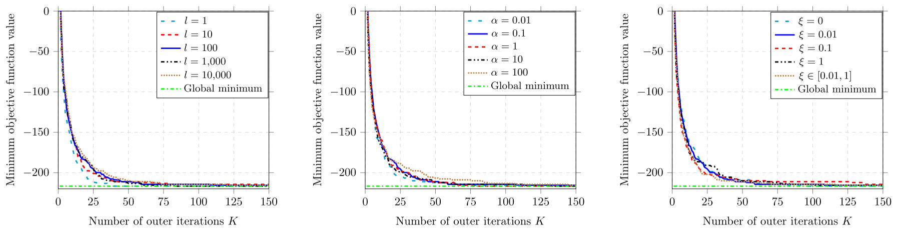

Fig. 22 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 trials for different length scales l (**left**), regularization terms α (**middle**) and exploration terms ξ (**right**) as a function of the number of outer iterations K

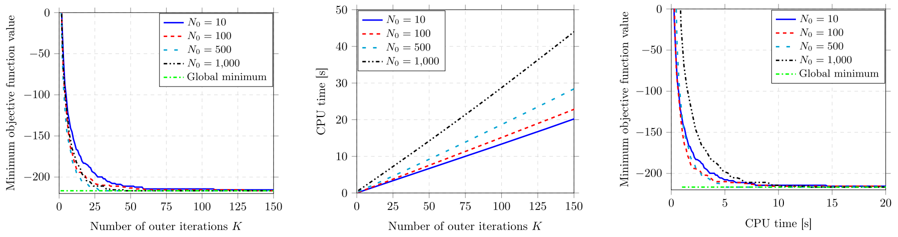

Fig. 23 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for different numbers of initialization points N_0 as a function of the number of outer iterations K

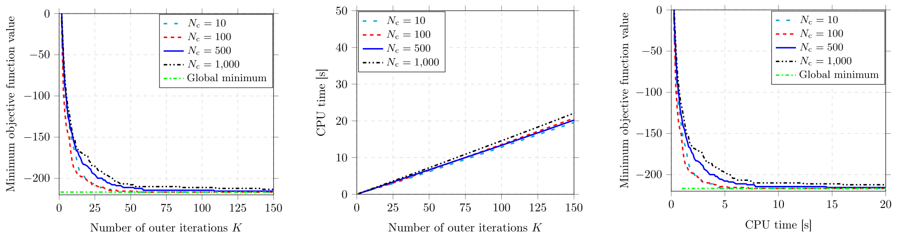

Fig. 24 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for different numbers of candidates N_c as a function of the number of outer iterations K

the averaged CPU time, and the corresponding solver performance are compared, but we also display the respective confidence interval using the standard deviation. In addition to that, the minimum objective function value as a function of the total number of function evaluations is displayed for all the benchmark problems in Figs. 28 and 29 in Appendix I. Furthermore, the experiments are repeated multiple times for averaging the stochastic nature of the optimizers, more precisely, we use 100 trials for the two test problems, 50 trials for the RSG, the 5-unit and 10-unit DED problem, and 20 trials for the 30-unit DED problem. We would like to point out that although the best minima for the DED problems (Santra et al. 2020) are lower than those achieved with the optimizers used here, an explicit comparison would lead to an unfair evaluation, since the hybrid method (combination of two stochastic methods) in the mentioned work[5] is specifically designed to solve the DED problem. In contrast, BO-IPOPT aims to effectively solve a wide range of optimization problems and not only the DED problem.

Our proposed BO-IPOPT method significantly outperforms the other two approaches in terms of convergence and robustness (cf. Figs. 13, 14, 15, 16 and 17). As expected, MS-IPOPT and BO-IPOPT start (at $K = 1$) with a higher minimum objective function value than BOwLS because both methods use a random initialization for the initial GPR, while BOwLS uses already calculated local optima. Nevertheless, BO-IPOPT generally converges on average faster toward the global minimum (fewer iterations K and total number of function evaluations required), which can be explained as follows: on the one hand, MS-IPOPT has a slower convergence as it strongly depends on the randomly chosen starting points. On the other hand, the GPR in BO-IPOPT provides a better surrogate model (and thus accelerates the convergence) because the sampling is more spread than in BOwLS, which narrows the sampling near the initial local optima. The latter issue is discussed in more detail in Appendix F. In terms of robustness, BO-IPOPT shows better worse-case (upper curves) and best-case scenarios (lower curves) compared to the other two optimizers, with the worse-case scenario of BO-IPOPT often being close to the average scenario (mean value) or the best-case scenario of BOwLS and MS-IPOPT, as the dimension of the problem increases. A summary comparing the accuracy and robustness of all three optimizers across the benchmarks is presented in Table 1.

In terms of CPU time, MS-IPOPT outperforms both hybrid methods as no additional computational effort is required to evaluate the EI and train the GPR. These additional computational costs for the hybrid methods increase slightly with increasing K because the inversion of the correlation matrix becomes more time-intensive as more samples are considered in the GPR at each K. Nonetheless, the CPU times show an almost linear trend for both the hybrid methods and MS-IPOPT. Moreover, BOwLS is more computationally expensive than BO-IPOPT due to the fact that the initial GPR is constructed by computing local searches, so the graph for the CPU time of BOwLS is shifted upwards by a constant factor compared to BO-IPOPT. This difference in CPU time between the two hybrid methods becomes even more significant with increasing problem dimension.

The benchmark problems also serve to compare the scalability of the methods. All methods indicate a linear trend in the CPU time with the slope becoming larger as the dimension of the optimization problem increases. To be more precise, we consider the DED problem as an example. As seen in Table 2, the increase of the CPU time of MS-IPOPT, BOwLS and BO-IPOPT shows a linear behavior (with larger slopes for the hybrid methods) as the dimension of the DED problem grows by additional units. The latter fact also demonstrates that the increased computational effort of the GPR (when more samples are available for the training of the surrogate model) weighs much less than the objective function evaluation (local search) itself, which requires significantly more CPU time with increasing problem dimension.

[5] All optimization methods compared in Santra et al. (2020) are specifically designed to solve the DED problem, making a fair comparison (best solution and CPU time required) with BO-IPOPT difficult.

Table 1 Comparison of accuracy and robustness of MS-IPOPT, BOwLS and BO-IPOPT for the constrained Ackley function, RSG, 5-, 10- and 30-unit DED problem at a specific CPU time

Benchmark problem	MS-IPOPT	BOwLS	BO-IPOPT
Ackley function	80 s	80 s	80 s
	0.077±0.254	0.099±0.303	0.046±0.222
DED (5 units)	650 s	650 s	650 s
	42,527.359 $ ±10.231 $	42,524.304 $ ±0.115 $	42,524.279 $ ±0 $
DED (10 units)	2,000 s	2000 s	2000 s
	1,026,649 $ ±576 $	1,025,222 $ ±600 $	1,024,357 $ ±1600 $
RSG	2500 s	2500 s	2500 s
	1057 €±1.623 €	1056 €±1.684 €	1,054 €±3.251 €
DED (30 units)	5500 s	5500 s	5,500 s
	3,079,636 $ ±1136 $	3,060,178 $ ±1825 $	3,058,375 $ ±1721 $

Table 2 CPU time of MS-IPOPT, BOwLS and BO-IPOPT for the 5-, 10- and 30-unit DED problem at $K = 300$

Optimizer	5 Units (s)	10 Units (s)	30 Units (s)
MS-IPOPT	2110	2399	5512
BOwLS	2260	2602	6024
BO-IPOPT	2184	2557	5929

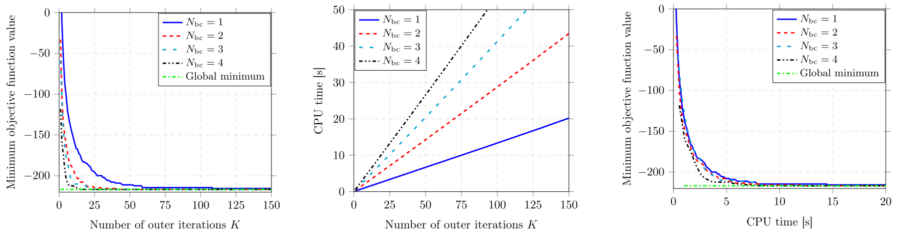

Fig. 25 Optimization results for the simple test problem (9)–(17) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials for different numbers of best candidates N_{bc} as a function of the number of outer iterations K

BO-IPOPT effectively addressed domain-specific challenges such as high-dimensional search spaces, complex multi-modal landscapes, and nonlinear constraints. Key adaptations included: defining the GPR length scale to balance global exploration and model accuracy; implementing a decreasing exploration term to ensure early global exploration and later local refinement; dynamically penalizing constraint violations via the AL framework to transition smoothly toward feasible regions; and starting the GPR correlation matrix with a small initialization set while evaluating the acquisition function on discrete candidates, ensuring linear CPU time growth with iterations.

In this context, the question also arises whether and how the choice of the local solver used in BO-IPOPT can influence the performance of the hybrid method. In addition to IPOPT, we are exemplarily testing two popular commercial solvers, namely XPRESS (Fair Isaac Corporation) and CONOPT4 (Drud 1994), using the 10-unit DED and RSG problem, as illustrated in Figs. 30 and 31 in Appendix J. As might be expected, the selected local solver has a significant influence on the convergence rate and the computational effort of the hybrid method. Although the commercial solvers require less CPU time, they produce inconsistent results regarding the minimum objective function value. Remarkably, BO-XPRESS converges extremely fast to the best known solution of the RSG problem, while its convergence in the DED problem is surprisingly much worse than that of BO-IPOPT. This means in particular that the use of commercial solvers is not always sufficient to achieve the best performance. Analyzing and comparing local solvers is an interesting topic for future work and may yield a further improvement of our proposed hybrid method.

Overall, BO-IPOPT can be considered superior to the other two methods as it combines high accuracy, robustness and computational efficiency for the class of optimization problems considered in this work, especially when the problem's dimension and complexity increase and deterministic global solvers can not thus be applied.

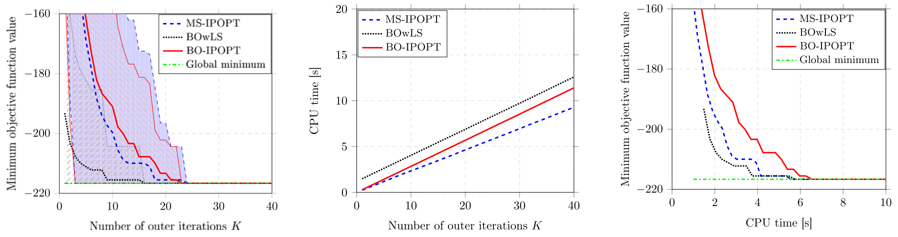

Fig. 26 Optimization results for the simple test problem (9)–(17): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer

Nonetheless, it can be observed that none of the optimizers converge (on average) to the best known solution in the 10- and 30-unit DED and RSG problem within the $K = 300$ outer iterations. This highlights the complexity of the optimization problems at hand, but also motivates us to further improve the BO-IPOPT method in future work.

5 Conclusion

In this work, we have enhanced the recently developed hybrid optimization method, BO-IPOPT, designed to solve nonlinear constrained optimization problems by exploring several aspects of its Bayesian Optimization (BO) component. These enhancements focus on improving the global exploration and computational efficiency of BO, as well as its integration with the local search capabilities of the open-source IPOPT solver.

First, we showed that evaluating the EI acquisition function over a discrete set, rather than through continuous optimization as typically done, is an effective strategy for sample selection in BO-IPOPT. This approach balances convergence speed and computational effort while maintaining BO's global guidance and addressing the scalability challenges often faced in high-dimensional problems. Future work will investigate the integration of novel acquisition functions, including LogEI (Ament et al. 2023) and compositional acquisition functions (Grosnit et al. 2021), into our framework. Correlation between candidates could also be explored to provide more effective batch selection using the q-EI approach (Wang et al. 2020), mitigating the risk of selecting points from the same region, which potentially leads to same local optima. Second, we improved constraint handling in BO by introducing a single surrogate model within the Augmented Lagrangian framework, integrated with IPOPT. Unlike methods that use multiple surrogate models, our approach reduces computational complexity while providing effective global guidance. It smoothly handles equality and inequality constraints without requiring user-defined hyperparameters, offering a robust solution for high-dimensional problems with many constraints. Third, we conducted a comprehensive analysis of BO-IPOPT's hyperparameters to enhance global guidance while minimizing user input. Unlike traditional methods that incur extra computational costs by learning these parameters during optimization, our approach avoids this overhead, maintaining efficiency.

Based on these enhancements, we demonstrated the proposed hybrid method in two test problems (differing in problem size, degree of multimodality, and degree of nonlinearity). We compared the performance of our optimizer with efficient BO approaches such as SCBO, SEGOKPLS, and SEGOKPLS+K. According to the results, BO-IPOPT has a significantly superior performance, i.e., better objective function values at the same CPU time, than state-of-the-art BO methods developed for higher dimensions. Considering the two aforementioned test problems and two real-world problems, we tested the performance and scalability of our algorithm in comparison to the classical random-based MS-IPOPT and the hybrid BOwLS method.

The numerical experiments showed that BO-IPOPT outperforms BOwLS and MS-IPOPT in terms of convergence rate and robustness, except for the simple test problem, where all three methods have similar performance. This highlights the advantage of our method, especially for complex problems of higher dimensions where the other two methods encounter difficulties. In particular, the performance gain over BOwLS is due to the different initialization criteria and the use of appropriate surrogate models that better approximate the original function.

Regarding the computational costs, BO-IPOPT is less computationally intensive than BOwLS because it avoids local searches for the initial GPR, which additionally restricts this surrogate model too close to the initial local optima. The overhead of the hybrid methods in CPU time is mainly related to the construction of the GPR at each outer iteration, but all benchmark problems show a linear trend as the number of outer iterations increases. Nevertheless, it is of great interest to test also other models, e.g., linear models, for the construction of the GPR in future work in order to investigate their impact on the convergence rate and CPU time of our hybrid method. In addition, we experimentally observed that the BO-IPOPT produces a linear increase in CPU time with increasing problem dimension by exemplarily using the DED problem with 5, 10 and 30 units. This showcases the potential to apply our hybrid method also to larger problems than those considered here. Testing BO-IPOPT on large-scale problems ($n > 5{,}000$) will be an important direction for future research, as it would provide valuable insights into the scalability and robustness of the method under more demanding conditions. Additionally, we aim to apply BO-IPOPT to more real-world problems to assess its practical performance and evaluate whether our assumption-that the objective functions considered are expensive enough to make evolutionary algorithms less appealing but have a dimensionality where optimizing the acquisition function is still non-negligible-holds in practical scenarios.

Finally, our hybrid method offers the flexibility to change the local solver, which is advantageous since different types of (including commercial) algorithms perform well on different types of problems. Overall, we successfully combined two optimization methods with complementary properties, resulting in an algorithm that is both easy to implement and effective for solving nonlinear constrained optimization problems.

Although this study focuses solely on continuous variables, this algorithm could be extended to address mixed-integer variables in future work.

Appendix

Supplementary data on this article are given below.

A AL framework with slack variables

AL methods are iterative procedures (Picheny et al. 2016), in which the following subproblem derived from (3) is solved with the current parameter sequence $\big(\boldsymbol{\lambda}_{\boldsymbol{g}}^{j-1}, \boldsymbol{\lambda}_{\boldsymbol{h}}^{j-1}, \rho^{j-1}\big)$:

$$\min_{\boldsymbol{x}\in\Omega} \Big\{ u\big(\boldsymbol{x}, \boldsymbol{\lambda}_{\boldsymbol{g}}^{j-1}, \boldsymbol{\lambda}_{\boldsymbol{h}}^{j-1}, \rho^{j-1}, \boldsymbol{s}(\boldsymbol{x})\big) \Big\} \tag{34}$$

with the slack variable vector (4) dependent on $\boldsymbol{x}$. The Lagrange multiplier vectors $\boldsymbol{\lambda}_{\boldsymbol{g}}$, $\boldsymbol{\lambda}_{\boldsymbol{h}}$ for the inequality and equality constraints, the penalty parameter ρ and the slack variables $\boldsymbol{s}$ are based on Algorithm 3 depending on the solution of (34). The vector $\boldsymbol{\epsilon} \geq \mathbf{0}$ (line 7) serves as a threshold for equality constraints and is set to $\mathbf{10^{-2}}$ as suggested in Picheny et al. (2016). Solving (34) requires an initialization of the parameter sequence; here we follow the settings suggested in Gramacy (2020). In doing so, $\boldsymbol{\lambda}_{\boldsymbol{g}}^{0} = \boldsymbol{\lambda}_{\boldsymbol{h}}^{0} = \mathbf{0}$ and the initial ρ^0 is determined using the rule

$$\rho^0 = \frac{\min\limits_{i=1,\ldots,N_0} \left\{ \sum\limits_{w=1}^{q} \max(0, g_w(\boldsymbol{x}_i))^2 + \sum\limits_{r=1}^{p} h_r^2(\boldsymbol{x}_i) : \exists w, r,\, v_w(\boldsymbol{x}_i) = 0,\, v_r(\boldsymbol{x}_i) = 0 \right\}}{2 \min\limits_{i=1,\ldots,N_0} \{ f(\boldsymbol{x}_i) : \forall w, r,\, v_w(\boldsymbol{x}_i) = 1,\, v_r(\boldsymbol{x}_i) = 1 \}} \tag{35}$$

where $\boldsymbol{v}(\boldsymbol{x}_i)$ is a logical vector, determining the validity of $\boldsymbol{x}$ in a zero-slack setting. If the w^{th} inequality constraint is satisfied, i.e., $g_w(\boldsymbol{x}) \leq 0$, let $v_w(\boldsymbol{x}) = 1$ for $w = 1, \ldots, q$ and if the r^{th} equality constraint is satisfied, i.e., $|h_r(\boldsymbol{x})| \leq \epsilon$, let $v_r(\boldsymbol{x}) = 1$ for $r = 1, \ldots, p$; otherwise let $v_w(\mathbf{x}) = \mathbf{v_r}(\mathbf{x}) = \mathbf{0}$. If the denominator for the initial design is not defined, i.e., the initial design has no valid values, then the median of $f(\boldsymbol{x}_i)$ is used in the denominator instead. Otherwise, if the initial design has no invalid values, i.e., the numerator is not defined, then $\rho^0 = 1$. Finally, to solve (34) approximately (line 3) at each outer iteration K, we terminate the BO solver after one iteration as suggested in Picheny et al. (2016). In other words, at each K, the best $\boldsymbol{x}$ so far is selected from the set of points D, which matches the search of the acquisition function EI performed at each outer iteration as well.

When BO and the AL framework are combined with a local deterministic solver (e.g., IPOPT) as proposed in this work, the following simplifications result in the parameter update process: Let $j = 0$: since $\boldsymbol{\lambda}_{\boldsymbol{g}}^{0} = \boldsymbol{\lambda}_{\boldsymbol{h}}^{0} = \mathbf{0}$ and a local optimum that belongs to D_1 satisfies either $g_w(\boldsymbol{x}_1^{\star}) = 0$ or $g_w(\boldsymbol{x}_1^{\star}) < 0$ (neglecting here rounding

Algorithm 3 AL method with slack variables

```
    Input: Initial Lagrange multiplier vectors $\boldsymbol{\lambda}_{\boldsymbol{g}}^0, \boldsymbol{\lambda}_{\boldsymbol{h}}^0 \geq 0$; initial penalty parameter $\rho^0 > 0$
1:  $j = 0$
2:  while $j < K$ do
3:      Let $\boldsymbol{x}^{j+1} \in D_{j+1}$ approximately solve (34)
4:      Compute $s_w(\boldsymbol{x}^{j+1}) = \max\left\{0, -\lambda_{w,\boldsymbol{g}}^j \rho^j - g_w(\boldsymbol{x}^{\boldsymbol{j+1}})\right\}$ for $w = 1{:}q$
5:      Set $\lambda_{w,\boldsymbol{g}}^{j+1} = \lambda_{w,\boldsymbol{g}}^j + \frac{1}{\rho^j}\left(g_w(\boldsymbol{x}^{j+1}) + s_w(\boldsymbol{x}^{j+1})\right)$ for $w = 1{:}q$
6:      Set $\lambda_{r,\boldsymbol{h}}^{j+1} = \lambda_{r,\boldsymbol{h}}^j + \frac{1}{\rho^j} h_r(\boldsymbol{x}^{j+1})$ for $r = 1{:}p$
7:      if $\boldsymbol{g}(\boldsymbol{x}^{j+1}) \leq \boldsymbol{0}$ and $|\boldsymbol{h}(\boldsymbol{x}^{j+1})| \leq \boldsymbol{\epsilon}$ then
8:          Set $\rho^{j+1} = \rho^j$
9:      else
10:         Set $\rho^{j+1} = \frac{1}{2}\rho^j$
11:     end if
12:     $j := j + 1$
13: end while
```

errors), we have $s_w(\boldsymbol{x}_1^\star) = 0$ or $s_w(\boldsymbol{x}_1^\star) = -g_w(\boldsymbol{x}_1^\star)$ for $w = 1, \dots, q$. This results (line 5) in $\lambda_{w,\boldsymbol{g}}^1 = 0, \forall w$. This also holds for $j > 0$ and therefore the Lagrange multipliers for inequality constraints are constant, i.e., $\lambda_{w,\boldsymbol{g}}^{j+1} = 0$ for all w, j, during the optimization process. The same applies to the Lagrange multipliers for the equality constraints. Since local optima satisfy $h_r(\boldsymbol{x}_1^\star) = 0$ for $r = 1, \dots, p$ (neglecting rounding errors), it holds $\lambda_{r,\boldsymbol{h}}^1 = 0$ and thus $\lambda_{r,\boldsymbol{h}}^{j+1} = 0$, see line 6. Based on this, ρ^{j+1} remains constant at each outer iteration K, since all constraints are fulfilled in local optima. All these simplifications rely on the use of e.g., IPOPT, leading (3) to $u(\boldsymbol{x}_k^\star, \boldsymbol{s}(\boldsymbol{x}_k^\star)) \equiv f(\boldsymbol{x}_k^\star)$ for $k = z + 1, \dots, z + N_{\text{bc}}$ at each K. In simple terms, the objective function value for the AL technique at each local minimum is equal to the local minimum provided by IPOPT (line 9 of Algorithm 2), since $\boldsymbol{\lambda}_{\boldsymbol{g}} = \boldsymbol{\lambda}_{\boldsymbol{h}} = \boldsymbol{0}$ and thus $\boldsymbol{s}(\boldsymbol{x}_k^\star) = -\boldsymbol{g}(\boldsymbol{x}_k^\star)$. As a result, AL has a direct impact only on the initial GPR in BO-IPOPT.

B MS-IPOPT

This section presents the algorithm of the classical random MS-IPOPT (cf. Algorithm 4). As seen in this algorithm, each MS outer iteration consists of two main steps: first the starting point is generated (randomly), then the local search is started from this point. For higher N_{bc} values, the algorithm is simply adapted in line 2, i.e., the corresponding number of random points are generated.

C Hybrid method BOwLS

The competing BOwLS method recently presented in Gao et al. (2020) is similar to our proposed BO-IPOPT. The implementation of BOwLS (cf. Algorithm 5) fundamentally differs from BO-IPOPT (cf. Algorithm 2) from an algorithmic point of view in only

Algorithm 4 MS-IPOPT

```
Input: Number of outer iterations $K$
1: while $j < K$ do
2:     Generate a random point $\boldsymbol{x}_i$ in $\Omega$
3:     Solve $[y_i^\star, \boldsymbol{x}_i^\star] = \text{IPOPT}(\boldsymbol{x}_i)$
4: end while
5: return $y_{\min} = \min\{y_j^\star\}_{j=1}^K$
```

two steps. However, we note that there are some technical differences to our version, which are explained below.

C.1 Original BOwLS

By combining the local search with the BO framework, Gao et al. (2020) also aim to appropriately determine the starting points of the local solver and thus create a suitable MS formulation. The BOwLS method differs algorithmically from our method in two essential points: first, the initial GPR is built from the results of local searches (cf. line 2 of Algorithm 5) rather than random points, as we proposed in our method. This entails the risk that the sampling region of the initial GPR will be too restricted and the convergence of the optimizer will become slower, see discussion in Appendix F. Second, the initial and updated GPR (cf. lines 3 and 10) are based on the objective function values $\boldsymbol{y}^\star$ of the local optima but paired with the initial points before using the local solver, i.e., $\boldsymbol{x}$ instead of $\boldsymbol{x}^\star$. Consequently, at each iteration, the GPR does not approximate the underlying original function, but a different one that has the same minimum values identified so far. This proceeding usually poses the risk that the GPR is based on infeasible solutions, which makes the training of the surrogate model more difficult. For further details, we refer the reader to the original work (Gao et al. 2020).

From a theoretical point of view, the BOwLS method is originally developed to solve unconstrained optimization problems. More precisely, BOwLS uses the conjugate gradient method from the SCIPY package as local search to optimize the objective function f without considering any constraints. However, this severely limits the application of BOwLS because most real-world optimization problems involve constraints, and as mentioned earlier, incorporating constraints into the BO framework is not straightforward theoretically and practically. The latter aspect is delicate and is addressed by our work.

C.2 Constraint handling in BOwLS

Since this work considers constrained optimization problems introduced by (1), the BOwLS method must be adapted to deal with equality and inequality constraints. As substitution for the original local solver (conjugate gradient method), IPOPT is used in this work to ensure a fair comparison. For the BO part in BOwLS, two techniques are available to deal with both equality and inequality constraints (see Sect. 2.1): augmenting the objective function with the constraints as penalty terms (2) or the AL framework (3). However, AL is not suitable/consistent for BOwLS and leads to

Algorithm 5 Adapted Hybrid Method BOwLS

```
Input: Length scale $l$; regularization term $\alpha$; exploration $\xi$; number of initialization points $N_0$; number of candidates $N_{\text{c}}$ considered in EI; number of best candidates $N_{\text{bc}}$; penalty weight vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$; number of outer iterations $K$
1: Generate an initial set of points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_0}\}$ in $\Omega$
2: Solve $[y^\star_{i_0}, \boldsymbol{x}^\star_{i_0}] = \text{LS}(\boldsymbol{x}_{i_0})$ for $i_0 = 1{:}N_0$
3: Let $D_0 = \{(\boldsymbol{x}_{i_0}, y^\star_{i_0})\}_{i_0=1}^{N_0}$
4: Construct a GPR $\hat{u}_0$ from $D_0$
5: $z = N_0,\ j = 0,\ FE := \sum_{i_0=1}^{N_0} FE_{\text{IPOPT}(\boldsymbol{x}_{i_0})}$
6: while $j < K$ do
7:     Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_{N_{\text{c}}}\}$ in $\Omega$
8:     $\{\boldsymbol{x}_{z+1}, \dots, \boldsymbol{x}_{z+N_{\text{bc}}}\} \in \underset{\boldsymbol{x} \in D_j}{\arg\max}\, \text{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_{\text{c}}$
9:     Solve $[y^\star_k, \boldsymbol{x}^\star_k] = \text{LS}(\boldsymbol{x}_k)$ for $k = z + 1{:}z + N_{\text{bc}}$
10:    $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}, y^\star_{z+1}), \dots, (\boldsymbol{x}_{z+N_{\text{bc}}}, y^\star_{z+N_{\text{bc}}})\}$
11:    Update GPR $\hat{u}_{j+1}$ from $D_{j+1}$
12:    $z := z + N_{\text{bc}},\ j := j + 1,\ FE := FE + N_{\text{bc}} + \sum_{k=z+1}^{z+N_{\text{bc}}} FE_{\text{IPOPT}(\boldsymbol{x}_k)}$
13: end while
14: return $y_{\min} = \min\{y^\star_j\}_{j=1}^{KN_{\text{bc}}}$
```

worse optimizer performance compared to (2), which can be explained as follows: the required parameters in the AL approach are updated depending on the best $\boldsymbol{x}$-value in D of the previous outer iteration K (cf. Algorithm 3), which is not a local minimum in BOwLS. Consequently, the constraints are most probably not fulfilled, leading the Lagrange multipliers $\lambda^{j+1}_{w,\boldsymbol{g}}$ and $\lambda^{j+1}_{r,\boldsymbol{h}}$ (for $w = 1, \dots, q$ and $r = 1, \dots, p$) not to vanish during the optimization process. Due to this, we have $u(\boldsymbol{x}_k, \boldsymbol{s}(\boldsymbol{x}_k)) \neq f(\boldsymbol{x}^\star_k)$ for $k = z + 1, \dots, z + N_{\text{bc}}$ (in line 9 of Algorithm 5), since $\boldsymbol{\lambda}_{\boldsymbol{g}}, \boldsymbol{\lambda}_{\boldsymbol{h}} \neq \boldsymbol{0}$ and thus $\boldsymbol{s}(\boldsymbol{x}_k) \neq -\boldsymbol{g}(\boldsymbol{x}_k)$. This leads to large changes in the parameters to be updated along with strong variations in (3), which consequently affects the convergence of the optimizer. In contrast, BO-IPOPT using the AL framework does not exhibit this behavior (as already discussed above in Sect. A), because the parameters are updated based on local solutions. Finally, we recommend the penalty term approach (2) for BOwLS, which we use for the comparison with BO-IPOPT and MS-IPOPT for all experiments in this work.

D Input data for DED and RSG

The optimization problems introduced in Sects. 4.1.3 and 4.1.4 require input data. For both problems, a typical one-day operation with an hourly time step size is considered, i.e., divided into 24 time intervals. For the DED problem (20)–(23), we distinguish between the use of 5, 10 and 30 units. The scenario input data for the 5- and 10-unit system is taken from Santra et al. (2020), where the load demand is visualized in Fig. 18. The data for the 30-unit system are obtained by simply tripling the 10-unit system data. The required data for the WT power production and the grid electricity price for the RSG problem (24)–(33) are shown in Fig. 18.

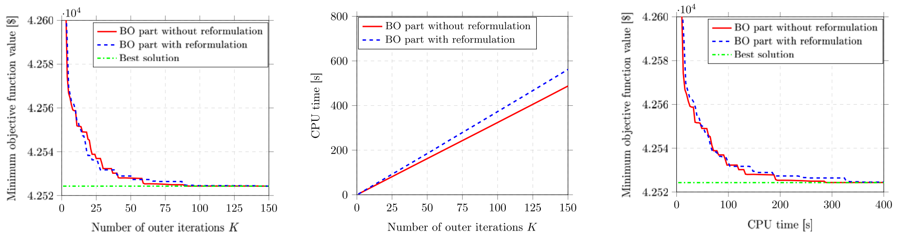

Fig. 27 Optimization results for the DED problem (20)–(23) with 5 units using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 50 trials by reformulating the problem in both parts of BO-IPOPT (i.e., BO and IPOPT) and only in IPOPT as a function of the number of outer iterations K

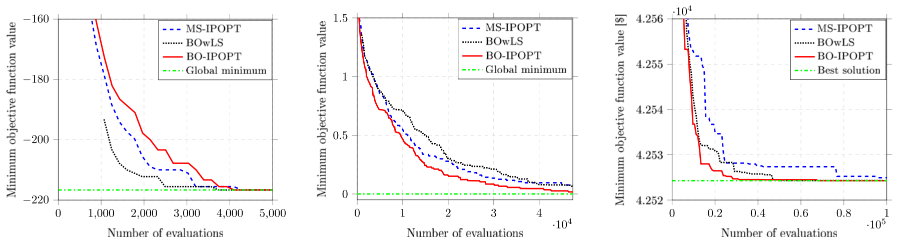

Fig. 28 Optimization results for the simple test problem (9)–(17) (**left**), the constrained Ackley function (18)–(19) (**middle**) and the DED problem (20)–(23) with 5 units (**right**): comparison of the averaged minimum objective function value over 100 and 50 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the total number of function evaluations

E BO-IPOPT: single GP model versus independent GP models for constraint handling

The classical BO builds independent GP models for the objective function and the constraints, as mentioned in Sect. 2.1. In this section, we compare the performance of BO-IPOPT considering one surrogate model for the objective function as well as the constraints and independent GP models on the simple test problem and the Ackley function. In the case of separate GP models, the EI criterion is weighted by the probability of satisfying each constraint (Gelbart et al. 2014; Gardner et al. 2014). Figures 19 and 20 show that the independent GP models for the constraints improve slightly the convergence rate of BO-IPOPT, but this technique increases significantly the computational effort of our hybrid method, especially in the simple model, which consists of more constraints than the Ackley function. Increasing the problem dimension and the number of constraints will have a greater negative impact on the CPU time and we thus build a single GP model for the objective function and all the constraints in BO-IPOPT in this work.

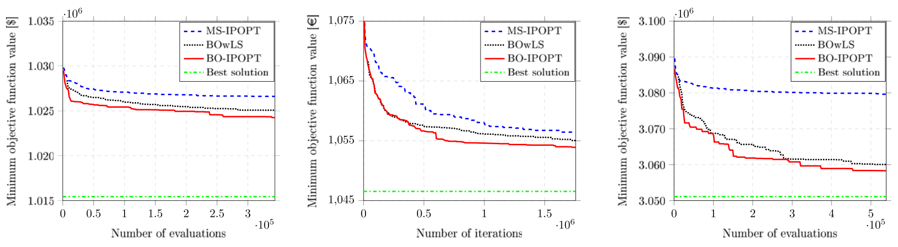

Fig. 29 Optimization results for the DED problem (20)–(23) with 10 units (**left**), for the RSG problem (24)–(33) (**middle**) and the DED problem (20)–(23) with 30 units (**right**): comparison of the averaged minimum objective function value over 50 and 20 trials using MS-IPOPT, BOwLS and BO-IPOPT as a function of the total number of function evaluations

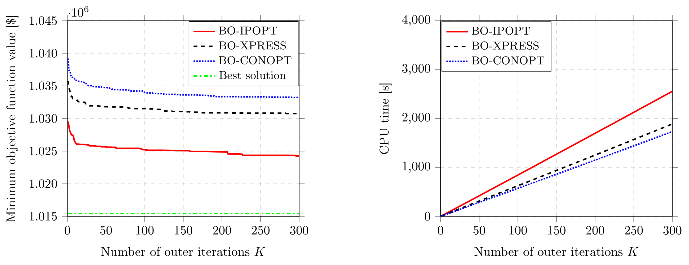

Fig. 30 Optimization results for the DED problem (20)–(23) with 10 units: comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 50 trials using BO-IPOPT, BO-XPRESS and BO-CONOPT as a function of the number of outer iterations K. IPOPT is open-source, while XPRESS and CONOPT are commercial solvers

F BO-IPOPT: random versus local N_0

An important issue that has not been discussed yet is the question of how the initialization points that form the initial GPR are generated. In the construction of our hybrid method BO-IPOPT, we propose the use of random-based initialization points N_0 to allow the GP model with a sampling distribution in the entire space, so that the convergence of the optimizer can be accelerated. The latter strategy is in contrast to the BOwLS method, which uses local searches for the initial GPR. Therefore, we demonstrate the benefit of our approach using the constrained Ackley function. As it can be clearly seen in Fig. 21, BO-IPOPT with random initialization points $N_0 = 10$ shows better performance (figure on the right) compared to using local minima for the construction of the first GPR. As expected, even increasing the number of local minima is not really helpful, since the initial GPR of the hybrid method still remains quite restricted. Consequently, this influences the selection of the next best candidates as well resulting in slower convergence of the optimizer.

G Results of the simple test problem

The results of the simple test problem regarding the influence of the hyperparameters involved in BO-IPOPT on its convergence rate and computational costs as well as the comparison of BO-IPOPT with BOwLS and MS-IPOPT are presented below.

G.1 Parameter study and setting

The variation of the hyperparameters l, α and ξ in BO-IPOPT has only a minor impact on the solver performance due to the low dimension and complexity of the problem, as shown in Fig. 22. Obviously, the influence of the parameters N_0, N_{c} and N_{bc} is a bit more noticeable (cf. Figs. 23, 24 and 25), however, this test problem is not suitable for analyzing the effect of the hyperparameters as the global minimum is obtained very quickly for all parameter settings.

G.2 Evaluation: MS-IPOPT, BOwLS versus BO-IPOPT

As seen in Fig. 26, the hybrid method BOwLS performs slightly better than the other two methods, especially in the first outer iterations K. This is because the initial GPR in BOwLS starts with local minima that are very close to the global minimum due to the low problem dimension and complexity. Nevertheless, all three methods converge to the optimum in almost the same CPU time ($\approx 6\,\mathrm{s}$), since BOwLS requires higher computational costs. If the complexity and dimension of the problem increase (see Sect. 4.2.5), BOwLS converges significantly more slowly toward the optimum.

H Reformulation of the DED problem

As mentioned in Sect. 4.2.2, the nonsmooth DED problem needs to be appropriately reformulated when applying gradient-based solvers such as IPOPT. As a result, the original DED problem (with 5 units) is enlarged by introducing additional variables as suggested in Pan et al. (2018). In contrast, the BO framework does not necessarily need any reformulation and can be directly applied to the original problem. In this case, however, the slack variables in IPOPT must either be randomly initialized or set to zero, as BO does not provide any information about these values. With this in mind, the question arises whether the same problem reformulation in the BO part improves the performance of BO-IPOPT by providing IPOPT better starting values for the slack variables. According to Fig. 27, the reformulation in both parts of BO-IPOPT increases the CPU time of the method without having a significant impact on its convergence, which leads to a worse performance of BO-IPOPT. Increasing the problem dimension considering more units in the DED problem has a greater negative effect on the performance of the BO framework, since a larger matrix must be inverted at each iteration, which drastically increases the computational costs. Therefore, we use in this work the original problem formulation for the BO part of BO-IPOPT.

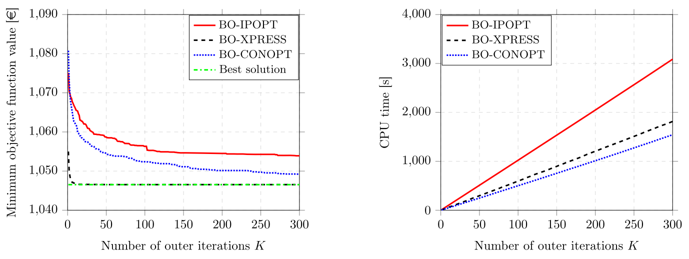

Fig. 31 Optimization results for the RSG problem (24)–(33): comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 50 trials using BO-IPOPT, BO-XPRESS and BO-CONOPT as a function of the number of outer iterations K. IPOPT is open-source, while XPRESS and CONOPT are commercial solvers

I Results over the total number of function evaluations

This section displays the minimum objective function value in relation to the total number of function evaluations for all the benchmark problems, comparing BO-IPOPT with MS-IPOPT and BOwLS. Figures 28 and 29 compare BO-IPOPT with MS-IPOPT and BOwLS based on the total number of function evaluations required to approach the global minimum or the best solution of the benchmark problems considered in this work. As clearly seen in these figures, our method needs less function evaluations than the other two optimizers to converge toward global optimum or best solution.

J Comparison of BO-IPOPT, BO-XPRESS and BO-CONOPT

The advantage of our proposed hybrid method is that it offers flexibility in the choice of the local solver used. In other words, the local solver can easily be substituted by any other local solver. IPOPT is selected in this work because it is a highly efficient and open-source solver. The question arises how competitive IPOPT is compared to commercial solvers. In this context, we exemplarily test two frequently used solvers in our proposed hybrid method, namely XPRESS (Fair Isaac Corporation) and CONOPT4 (Drud 1994). Figures 30 and 31 show the optimization results for DED and RSG, respectively, using the three versions of our hybrid method. Obviously, BO-XPRESS and BO-CONOPT converge faster than BO-IPOPT toward the best solution in the RSG problem, while BO-IPOPT demonstrates a better convergence rate than BO-XPRESS and BO-CONOPT in the DED problem. These contradictory solver performances motivate us to analyze this issue in more detail in future work.

Acknowledgements Not applicable.

Author Contributions L.K.: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing—original draft, Writing—review and editing, Visualization. M.B.: Resources, Writing—original draft, Writing—review and editing, Visualization, Supervision. M.A.M: Writing—review and editing, Supervision. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Data will be made available on request.

Declarations

Ethics approval The manuscript does not report on or involve any animals, humans, human data, human tissue or plants.

Consent for publication Not applicable.

Conflict of interest The authors declare that they have no Conflict of interest as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

Ariafar S, Coll-Font J, Brooks D, Dy J (2019) ADMMBO: Bayesian optimization with unknown constraints using ADMM. J Mach Learn Res 20(123):1–26

Ament S, Daulton S, Eriksson D, Balandat M, Bakshy E (2023) Unexpected improvements to expected improvement for Bayesian optimization. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S (eds) Advances in neural information processing systems, vol 36, pp 20577–20612. https://papers.nips.cc/paper_files/paper/2023/hash/419f72cbd568ad62183f8132a3605a2a-Abstract-Conference.html

Arora JS, Elwakeil OA, Chahande AI, Hsieh CC (1995) Global optimization methods for engineering applications: a review. Struct Optim 9(3):137–159. https://doi.org/10.1007/BF01743964

Attaviriyanupap P, Kita H, Tanaka E, Hasegawa J (2002) A hybrid EP and SQP for dynamic economic dispatch with non-smooth fuel cost function. IEEE Trans Power Syst 17(2):411–416. https://doi.org/10.1109/TPWRS.2002.1007911

Asim M, Mashwani W, Yeniay Ö, Jan MA, Tairan N, Hussian H, Wang G-G (2018) Hybrid genetic algorithms for global optimization problems. Hacettepe J Math Stat 47:539–551

Andrei N (2013) Nonlinear optimization applications using the GAMS technology. In: Springer optimization and its applications, vol 81. https://doi.org/10.1007/978-1-4614-6797-7

Andrei N (2017) Continuous nonlinear optimization for engineering applications in GAMS technology. In: Springer optimization and its applications, vol 121. https://doi.org/10.1007/978-3-319-58356-3

Bouhlel MA, Bartoli N, Regis RG, Otsmane A, Morlier J (2018) Efficient global optimization for high-dimensional constrained problems by using the kriging models combined with the partial least squares method. Eng Optim 50:2038–2053. https://doi.org/10.1080/0305215X.2017.1419344

Brochu E, Cora VM, Freitas N (2010) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599

Binois M, Ginsbourger D, Roustant O (2020) On the choice of the low-dimensional domain for global optimization via random embeddings. J Glob Optim 76:69–90. https://doi.org/10.1007/s10898-019-00839-1

Balandat M, Karrer B, Jiang DR, Daulton S, Letham B, Wilson AG, Bakshy E (2020) BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. In: Advances in neural information processing systems, vol 33. arXiv:1910.06403

Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A (2013) Mixed-integer nonlinear optimization. Acta Numer 22:1–119. https://doi.org/10.1017/S0962492913000032

Bartoli N, Lefebvre T, Dubreuil S, Olivanti R, Priem R, Bons N, Martins JRRA, Morlier J (2019) Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. Aerosp Sci Technol 90:85–102. https://doi.org/10.1016/j.ast.2019.03.041

Berk J, Nguyen V, Gupta S, Rana S, Venkatesh S (2019) Exploration enhanced expected improvement for Bayesian optimization. In: Berlingerio M, Bonchi F, Gärtner T, Hurley N, Ifrim G (eds) Machine learning and knowledge discovery in databases, pp 621–637. https://doi.org/10.1007/978-3-030-10928-8_37

Balamurugan R, Subramanian S (2007) An improved differential evolution based dynamic economic dispatch with non-smooth fuel cost function. J Electric Syst 3:151–161

Bischi A, Taccari L, Martelli E, Amaldi E, Manzolini G, Silva P, Campanari S, Macchi E (2014) A detailed MILP optimization model for combined cooling, heat and power system operation planning. Energy 74:12–26. https://doi.org/10.1016/j.energy.2014.02.042

Cherki I, Chaker A, Djidar Z, Khalfallah N, Benzergua F (2019) A sequential hybridization of genetic algorithm and particle swarm optimization for the optimal reactive power flow. Sustainability 11(14):1–12. https://doi.org/10.3390/su11143862

Cheng R, Jin Y (2015) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45:191–204. https://doi.org/10.1109/TCYB.2014.2322602

Chelouah R, Siarry P (2003) Genetic and Nelder-mead algorithms hybridized for a more accurate global optimization of continuous multi-minima functions. Eur J Oper Res 148:335–348. https://doi.org/10.1016/S0377-2217(02)00401-0. (**Sport and Computers**)

Cao Z, Wang L, Hei X (2018) A global-best guided phase based optimization algorithm for scalable optimization problems and its application. J Comput Sci 25:38–49. https://doi.org/10.1016/j.jocs.2018.02.001

Dowling AW, Kumar R, Zavala VM (2017) A multi-scale optimization framework for electricity market participation. Appl Energy 190:147–164. https://doi.org/10.1016/j.apenergy.2016.12.081

Drud AS (1994) CONOPT: a large-scale GRG code. ORSA J Comput 6:207–216. https://doi.org/10.1287/ijoc.6.2.207

Eriksson D, Poloczek M (2021) Scalable constrained Bayesian optimization. In: Banerjee A, Fukumizu K (eds) Proceedings of the 24th international conference on artificial intelligence and statistics. Proceedings of machine learning research, vol 130, pp 730–738. https://proceedings.mlr.press/v130/eriksson21a.html

Eriksson D, Pearce M, Gardner J, Turner RD, Poloczek M (2019) Scalable global optimization via local Bayesian optimization. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc F, Fox E, Garnett R (eds.) Advances in neural information processing systems, vol 32, pp 1–12. https://proceedings.neurips.cc/paper/2019/hash/6c990b7aca7bc7058f5e98ea909e924b-Abstract.html

Fair Isaac Corporation (2025) FICO Xpress Optimization. https://www.fico.com/en/products/fico-xpress-optimization

Foundation PS (2023) Multiprocessing module. https://docs.python.org/3/library/multiprocessing.html

GAMS Development Corporation (2025) General Algebraic Modeling System (GAMS), 35 Distribution. https://www.gams.com/download/

GAMS Development Corporation (2025) Python API. https://www.gams.com/latest/docs/API_PY_OVERVIEW.html

Grosnit A, Cowen-Rivers AI, Tutunov R, Griffiths R-R, Wang J, Bou-Ammar H (2021) Are we forgetting about compositional optimisers in Bayesian optimisation? J Mach Learn Res 22:1–78. https://doi.org/10.5555/3546258.3546418

Gramacy RB, Gray GA, Le Digabel S, Lee HKH, Ranjan P, Wells G, Wild SM (2016) Modeling an augmented Lagrangian for improved blackbox constrained optimization. Technometrics. https://doi.org/10.1080/00401706.2015.1014065

Gardner JR, Kusner MJ, Zhixiang X, Weinberger KQ, Cunningham JP (2014) Bayesian optimization with inequality constraints. In: Xing EP, Jebara T (eds) Proceedings of the 31st international conference on machine learning, vol 32. PMLR, pp 937–945. https://proceedings.mlr.press/v32/gardner14.html

Gramacy RB (2020) Surrogates: Gaussian process modeling, design and optimization for the applied sciences. Chapman Hall/CRC, Boca Raton. http://bobby.gramacy.com/surrogates/

Greenhill S, Rana S, Gupta S, Vellanki P, Venkatesh S (2020) Bayesian optimization for adaptive experimental design: a review. IEEE Access 8:13937–13948. https://doi.org/10.1109/ACCESS.2020.2966228

Gelbart MA, Snoek J, Adams RP (2014) Bayesian optimization with unknown constraints. In: Conference on uncertainty in artificial intelligence, pp 250–259

Gao Y, Yu T, Li J (2020) Bayesian optimization with local search. In: Nicosia G, Ojha V, La Malfa E, Jansen G, Sciacca V, Pardalos P, Giuffrida G, Umeton R (eds.) Machine learning, optimization, and data science, pp 350–361. https://doi.org/10.1007/978-3-030-64580-9_30

Hebbal A, Balesdent M, Brevault L, Melab N, Talbi E-G (2023) Deep Gaussian process for multi-objective Bayesian optimization. Optim Eng. https://doi.org/10.1007/s11081-022-09753-0

He C, Cheng R, Tian Y, Zhang X, Tan KC, Jin Y (2021) Paired offspring generation for constrained large-scale multi-objective optimization. IEEE Trans Evol Comput 25:448–462. https://doi.org/10.1109/TEVC.2020.3047835

Hernández-Lobato JM, Gelbart MA, Hoffman MW, Adams RP, Ghahramani Z (2015) Predictive entropy search for Bayesian optimization with unknown constraints. In: International conference on machine learning, pp 1699–1707

Hendrix EMT, Tóth BG (2010) Introduction to nonlinear and global optimization. In: Springer optimization and its applications, vol 1. https://doi.org/10.1007/978-0-387-88670-1

Hart WE, Watson J-P, Woodruff DL (2011) Pyomo: modeling and solving mathematical programs in Python. Math Program Comput 3(3):219–260. https://doi.org/10.1007/s12532-011-0026-8

Jeong S, Obayashi S (2005) Efficient global optimization (EGO) for multi-objective problem and data mining. IEEE Congr Evol Comput 3:2138–2145. https://doi.org/10.1109/CEC.2005.1554959

Kelner V, Capitanescu F, Léonard O, Wehenkel L (2008) A hybrid optimization technique coupling an evolutionary and a local search algorithm. J Comput Appl Math 215:448–456. https://doi.org/10.1016/j.cam.2006.03.048

Kyriakidis L, Mendez MA, Bähr M (2024) A hybrid algorithm based on Bayesian optimization and interior point OPTimizer for optimal operation of energy conversion systems. Energy 312:1–10. https://doi.org/10.1016/j.energy.2024.133416

Kumar A, Wu G, Ali MZ, Mallipeddi R, Suganthan PN, Das S (2020) A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm Evol Comput. https://doi.org/10.1016/j.swevo.2020.100693

Lam R, Allaire D, Willcox K (2015) Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources. In: 56th AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference, pp 1–21. https://doi.org/10.2514/6.2015-0143

Luo J, Chen Z, Zheng Y (2022) A gradient-based method assisted by surrogate model for robust optimization of turbomachinery blades. Chin J Aeronaut 35:1–7. https://doi.org/10.1016/j.cja.2021.07.019

Larson J, Menickelly M, Wild SM (2019) Derivative-free optimization methods. Acta Numer 28:287–404. https://doi.org/10.1017/S0962492919000060

Lasdon L, Plummer JC (2008) Multistart algorithms for seeking feasibility. Comput Oper Res 35(5):1379–1393. https://doi.org/10.1016/j.cor.2006.08.008

Lu C, Paulson JA (2022) No-regret Bayesian optimization with unknown equality and inequality constraints using exact penalty functions. IFAC-PapersOnLine 55(7):895–902. https://doi.org/10.1016/j.ifacol.2022.07.558

Lan G, Tomczak JM, Roijers DM, Eiben AE (2022) Time efficiency in optimization with a Bayesian-evolutionary algorithm. Swarm Evol Comput 69:1–14. https://doi.org/10.1016/j.swevo.2021.100970

Long Q, Wu C (2014) A hybrid method combining genetic algorithm and Hooke–Jeeves method for constrained global optimization. J Ind Manage Optim 10(4):1279–1296. https://doi.org/10.3934/jimo.2014.10.1279

Lam R, Willcox K (2017) Lookahead Bayesian optimization with inequality constraints. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems, vol 30, pp 1888–1898. https://proceedings.neurips.cc/paper/2017/hash/83f97f4825290be4cb794ec6a234595f-Abstract.html

Martí R, Aceves R, León MT, Moreno-Vega JM, Duarte A (2019) Intelligent multi-start methods. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, pp 221–243. https://doi.org/10.1007/978-3-319-91086-4_7

Malu M, Dasarathy G, Spanias A (2021) Bayesian optimization in high-dimensional spaces: a brief survey. In: 2021 12th International conference on IISA, pp 1–8. https://doi.org/10.1109/IISA52424.2021.9555522

Ming F, Gong W, Li D, Wang L, Gao L (2023) A competitive and cooperative swarm optimizer for constrained multiobjective optimization problems. IEEE Trans Evol Comput 27:1313–1326. https://doi.org/10.1109/TEVC.2022.3199775

Martí R, Lozano JA, Mendiburu A, Hernando L (2018) Multi-start methods. In: Martí R, Pardalos PM, Resende MGC (eds) Handbook of heuristics, pp 155–175. https://doi.org/10.1007/978-3-319-07124-4_1

Ma T, Wu J, Hao L (2017) Energy flow modeling and optimal operation analysis of the micro energy grid based on energy hub. Energy Convers Manage 133:292–306. https://doi.org/10.1016/j.enconman.2016.12.011

Nayebi A, Munteanu A, Poloczek M (2019) A framework for Bayesian optimization in embedded subspaces. In: Chaudhuri K, Salakhutdinov R (eds.) Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research, vol 97, pp 4752–4761. https://proceedings.mlr.press/v97/nayebi19a.html

Ngo TT, Sadollah A, Kim JH (2016) A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems. J Comput Sci 13:68–82. https://doi.org/10.1016/j.jocs.2016.01.004

Nocedal J, Wright SJ (2006) Numerical optimization. In: Springer series in operations research and financial engineering, vol 2. https://doi.org/10.1007/978-0-387-40065-5

Priem R, Bartoli N, Diouane Y (2019) On the use of upper trust bounds in constrained Bayesian optimization infill criteria. AIAA Aviation 2019 Forum, pp 1–10. https://doi.org/10.2514/6.2019-2986

Priem R, Bartoli N, Diouane Y, Dubreuil S, Saves P (2023) High-dimensional efficient global optimization using both random and supervised embeddings. In: AIAA aviation 2023 forum, pp 1–19. https://doi.org/10.2514/6.2023-4448

Priem R, Bartoli N, Diouane Y, Sgueglia A (2020) Upper trust bound feasibility criterion for mixed constrained Bayesian optimization with application to aircraft design. Aerosp Sci Technol 105:105980. https://doi.org/10.1016/j.ast.2020.105980

Picheny V, Gramacy RB, Wild S, Digabel SL (2016) Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In: Proceedings of the 30th international conference on NIPS, pp 1443–1451. https://papers.nips.cc/paper_files/paper/2016/hash/31839b036f63806cba3f47b93af8ccb5-Abstract.html

Picheny V (2014) A stepwise uncertainty reduction approach to constrained global optimization. In: 17th International conference on artificial intelligence and statistics. Proceedings of the seventeenth international conference on artificial intelligence and statistics, vol 33, pp 787–795. https://proceedings.mlr.press/v33/picheny14.html

Pintér JD (2006) Global optimization: scientific and engineering case studies. In: Nonconvex optimization and its applications, vol 85. https://doi.org/10.1007/0-387-30927-6

Pan S, Jian J, Chen H, Yang L (2020) A full mixed-integer linear programming formulation for economic dispatch with valve-point effects, transmission loss and prohibited operating zones. Electric Power Syst Res. https://doi.org/10.1016/j.epsr.2019.106061

Pan S, Jian J, Yang L (2018) A hybrid MILP and IPM approach for dynamic economic dispatch with valve-point effects. Int J Electric Power Energy Syst 97:290–298. https://doi.org/10.1016/j.ijepes.2017.11.004

Parisio A, Rikos E, Glielmo L (2014) A model predictive control approach to microgrid operation optimization. IEEE Trans Control Syst Technol 22:1813–1827. https://doi.org/10.1109/TCST.2013.2295737

Pelamatti J, Riche RL, Helbert C, Blanchet-Scalliet C (2024) Coupling and selecting constraints in Bayesian optimization under uncertainties. Optim Eng 25:373–412. https://doi.org/10.1007/s11081-023-09807-x

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Duchesnay Perrot M (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12(85):2825–2830

Ravikumar Pandi V, Panigrahi BK (2011) Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm. Expert Syst Appl 38(7):8509–8514. https://doi.org/10.1016/j.eswa.2011.01.050

Roberts AP, Rahat AAM, Jarman DS, Fieldsend JE, Tabor GR (2024) Multi-objective Bayesian shape optimization of an industrial hydrodynamic separator using unsteady Eulerian-Lagrangian simulations. Optim Eng. https://doi.org/10.1007/s11081-024-09907-2

Rasmussen CE, Williams CKI (2005) Gaussian processes for machine learning. https://doi.org/10.7551/mitpress/3206.001.0001

Sankaran JK (1993) A note on resolving infeasibility in linear programs by constraint relaxation. Oper Res Lett 13(1):19–20. https://doi.org/10.1016/0167-6377(93)90079-V

Samanipour F, Jelovica J (2020) Adaptive repair method for constraint handling in multi-objective genetic algorithm based on relationship between constraints and variables. Appl Soft Comput 90:106143. https://doi.org/10.1016/j.asoc.2020.106143

Saves P, Lafage R, Bartoli N, Diouane Y, Bussemaker J, Lefebvre T, Hwang JT, Morlier J, Martins JRRA (2024) SMT 2.0: a surrogate modeling toolbox with a focus on hierarchical and mixed variables gaussian processes. Adv Eng Sofw 188:1–16. https://doi.org/10.1016/j.advengsoft.2023.103571

Santra D, Mukherjee A, Sarker K, Mondal SK (2020) Dynamic economic dispatch using hybrid metaheuristics. J Electric Syst Inf Technol 7:1–30. https://doi.org/10.1186/s43067-020-0011-2

Saves P, Nguyen Van E, Bartoli N, Lefebvre T, David C, Defoort S, Diouane Y, Morlier J (2023) Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design. In: AIAA SCITECH 2022 forum, pp 1–22. https://doi.org/10.2514/6.2022-0082

Sasena MJ, Papalambros PY, Goovaerts P (2001) The use of surrogate modeling algorithms to exploit disparities in function computation time within simulation-based optimization. In: The fourth world congress of structural and mutli-disciplinary optimisation, pp 1–6

Sivasubramani S, Swarup KS (2010) Hybrid SOA-SQP algorithm for dynamic economic dispatch with valve-point effects. Energy 35(12):5031–5036. https://doi.org/10.1016/j.energy.2010.08.018

Sakawa M, Yauchi K (2000) Floating point genetic algorithms for nonconvex nonlinear programming problems: revised GENOCOP III. Electron Commun Jpn Part 3(83):1–9

The Scikit-Optimize Contributors (2020) Scikit-Optimize: Sequential Model-Based Optimization. https://scikit-optimize.github.io/stable/

Tawarmalani M, Sahinidis NV (2002) Convexification and global optimization in continuous and mixed-integer nonlinear programming. Nonconvex Optim Appl. https://doi.org/10.1007/978-1-4757-3532-1

Tian Y, Zheng X, Zhang X, Jin Y (2020) Efficient large-scale multi-objective optimization based on a competitive swarm optimizer. IEEE Trans Cybern 50:3696–3708. https://doi.org/10.1109/TCYB.2019.2906383

Ulder NLJ, Aarts EHL, Bandelt H-J, Laarhoven PJM, Pesch E (1990) Genetic local search algorithms for the travelling salesman problem. In: International conference on parallel problem solving from nature, pp 109–116. https://doi.org/10.1007/BFb0029740

Ugray Z, Lasdon L, Plummer J, Glover F, Kelly J, Martí R (2007) Scatter search and local NLP solvers: a multistart framework for global optimization. INFORMS J Comput 19(3):328–340. https://doi.org/10.2139/ssrn.886559

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P (2020) SciPy 1.0 contributors: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2

Van Rossum G, Drake FL (2009) Python 3 reference manual

Watson AG, Barnes RJ (1995) Infill sampling criteria to locate extremes. Math Geol 27:589–608. https://doi.org/10.1007/BF02093902

Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 106:25–57. https://doi.org/10.1007/s10107-004-0559-y

Walden JVM, Bähr M, Glade A, Gollasch J, Tran AP, Lorenz T (2023) Nonlinear operational optimization of an industrial power-to-heat system with a high temperature heat pump, a thermal energy storage and wind energy. Appl Energy. https://doi.org/10.1016/j.apenergy.2023.121247

Wang H, Chen S, Luo L (2020) A diffusion algorithm based on P systems for continuous global optimization. J Comput Sci. https://doi.org/10.1016/j.jocs.2020.101112

Wang J, Clark SC, Liu E, Frazier PI (2020) Parallel Bayesian global optimization of expensive functions. Oper Res 68:1850–1865. https://doi.org/10.1287/opre.2019.1966

Wang Z, Hutter F, Zoghi M, Matheson D, De Freitas N (2016) Bayesian optimization in a billion dimensions via random embeddings. J Artif Intell Res 55:361–387

Wächter A (2002) An interior point algorithm for large-scale nonlinear optimization with applications in process engineering. PhD thesis, Carnegie Mellon University, Pittsburgh

Xu D, Wu Q, Zhou B, Li C, Bai L, Huang S (2020) Distributed multi-energy operation of coupled electricity, heating, and natural gas networks. IEEE Trans Sustain Energy 11(4):2457–2469. https://doi.org/10.1109/TSTE.2019.2961432

Yadav A, Deep K (2014) An efficient co-swarm particle swarm optimization for non-linear constrained optimization. J Comput Sci 5(2):258–268. https://doi.org/10.1016/j.jocs.2013.05.011

Zapata AAA, Suarez EG, Florez JAV (2019) Application of VRP techniques to the allocation of resources in an electric power distribution system. J Comput Sci 35:102–109. https://doi.org/10.1016/j.jocs.2019.06.010

Zhan Z-H, Shi L, Tan KC, Zhang J (2022) A survey on evolutionary computation for complex continuous optimization. Artif Intell Rev 55(1):59–110. https://doi.org/10.1007/s10462-021-10042-y

Zhang X, Zheng X, Cheng R, Qiu J, Jin Y (2018) A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. Inf Sci 427:63–76. https://doi.org/10.1016/j.ins.2017.10.037

Zhou J, Zhang Y, Suganthan PN (2024) Constrained large-scale multi-objective optimization based on a competitive and cooperative swarm optimizer. Swarm Evol Comput 91:1–12. https://doi.org/10.1016/j.swevo.2024.101735