Multi-Fidelity Optimization Approach under Prior and Posterior Constraints and its Application to Compliance Minimization

Youhei Akimoto[1,2], Naoki Sakamoto[1,2,3], and Makoto Ohtani[4]

[1] University of Tsukuba
[2] RIKEN Center for Advanced Intelligence Project
[3] JSPS Research Fellow (DC1)
[4] Honda R&D
{akimoto@,naoki@bbo.}cs.tsukuba.ac.jp
Makoto_Ohtani@n.t.rd.honda.co.jp

Abstract. In this paper we consider a multi-fidelity optimization under two types of constraints: prior constraints and posterior constraints. The prior constraints are prerequisite to execution of the simulation that computes the objective function value and the posterior constraint violation values, and are evaluated independently from the simulation with significantly lower computational time than the simulation. We have several simulators that approximately simulate the objective and constraint violation values with different trade-offs between accuracy and computational time. We propose an approach to solve the described constrained optimization problem with as least computational time as possible by utilizing multiple simulators. Based upon a probabilistic model based evolutionary approaches, we combines three algorithmic components: prior constraint handling technique, posterior constraint handling technique, and adaptive simulator selection technique for multi-fidelity optimization. We apply the proposed approach to a compliance minimization problem.

Keywords: Prior constraints · Posterior constraints · Multi-fidelity optimization · DD-CMA-ES · Compliance minimization.

1 Introduction

We consider the following simulation based constrained optimization problem

$$\begin{aligned} &\text{minimize: } f(\mathcal{S}(x)) \text{ for } x \in \mathbb{R}^n \\ &\text{subject to: } g_i(x) \leqslant 0 \text{ for } i = 1, \ldots, m_{\text{pri}} \\ &\qquad\qquad h_j(\mathcal{S}(x)) \leqslant 0 \text{ for } j = 1, \ldots, m_{\text{pos}} \ , \end{aligned} \tag{1}$$

where $\mathcal{S}$ is a simulator that takes x as an input and computes the intermediate information, g_i are the prior constraints (also called QUAK constraints in [7]) that the simulator requires x to satisfy to execute the simulation, and f and h_j are the objective and posterior constraint functions (also called QRSK constraints in [7]) that are computed based

on the output of the simulation. Combination of prior constraints and posterior constraints naturally and often appears in engineering optimization problems.

We consider the multi-fidelity situation where we have multiple simulators $\mathcal{S}_k$ ($k = 1, \ldots, m_{\text{sim}}$) whose accuracy and computational time are both higher for a greater k. That is, $\mathcal{S}_1$ is the lowest accuracy simulator with the lowest cost, and $\mathcal{S}_{m_{\text{sim}}}$ is the highest accuracy simulator with the highest cost. We assume that the computational time for g_i are all cheaper than those of $\mathcal{S}_k$, as the prerequisites to the simulator are usually formulated by a simple mathematical expression such as linear functions of x. Our goal is to solve problem (1) with the highest accuracy simulator $\mathcal{S}_{m_{\text{sim}}}$, while we would like to minimize the computational time for the optimization by utilizing $\mathcal{S}_k$ for $k = 1, \ldots, m_{\text{sim}} - 1$ as well.

We consider covariance matrix adaptation evolution strategies [10, 11] as the baseline solver as they are recognized as one of the state-of-the-art search strategies for simulation-based continuous optimization. Since they are mainly developed for unconstrained optimization, constraint handling techniques and strategies for multi-fidelity situation are required. There are existing algorithmic components that address each characteristic of the above mentioned problem. Prior constraints are considered in e.g. [4, 5, 13, 15], where a repair operator is applied either in Lamarckism or Darwinism manner with or without a penalty to the fitness of the solution. Posterior constraints, also called simulation-based constraints, are treated in e.g. [6, 8, 12], where f and h_j values or their rankings are aggregated to virtually transform the problem into an unconstrained one. Efficient use of multiple simulators in evolutionary computation for multi-fidelity optimization has been less addressed in the literature. Adaptive simulator selection mechanisms among more than two possible simulators have been proposed in [2, 3] for unconstrained optimization. However, these algorithmic components are mostly designed and applied to problems having solely one of these problem characteristic. It is not trivial how to combine these existing components to address the above mentioned difficulties simultaneously.

The contributions of this paper are summarized as follows.

1. To treat both prior and posterior constraints, we combine the recent prior constraint handling technique and posterior constraint handling technique. As far as the authors know, there are few researches that addresses these two types of constraints in a systematic way, i.e., problem-independent way. We employ ARCH [13] as a prior constraint handling, and MCR-mod [8] as a posterior constraint handling. Each of them considers solely each type of constraints. We combine these two techniques so that the combination do not disturb the adaptation mechanism in these techniques. The resulting technique is invariant to any strictly increasing transformation of the objective and constraint violation functions: $(f, \{g_i\}_{i=1}^{m_{\text{pri}}}, \{h_j\}_{j=1}^{m_{\text{pos}}}) \mapsto (e^f \circ f, \{e_i^g \circ g_i\}_{i=1}^{m_{\text{pri}}}, \{e_j^h \circ h_j\}_{j=1}^{m_{\text{pos}}})$, where e^f, e_i^g, e_j^h are arbitrary strictly increasing functions satisfying $e_i^g(0) = e_j^h(0) = 0$.
2. To exploit multiple simulators in the above stated multi-fidelity constrained optimization scenario, we generalize the adaptive simulator selection mechanism [2], which is proposed for unconstrained optimization, to constrained optimization. Then we combine the generalized simulator selection mechanism with the proposed constraint handling technique. As far as the authors know, this is the first

Algorithm 1 An Iteration of DD-CMA-ES

1: $\{x_1, \ldots, x_\lambda\} = \text{SAMPLE}_{\text{cma}}(m, \Sigma)$ ▷ generate candidate solutions
2: $\{r_\ell\}_\ell = \text{EVALUTE}(x_1, \ldots, x_\lambda)$ ▷ evaluate candidate solutions
3: $\theta \leftarrow \text{UPDATE}_{\text{cma}}(\theta, \{(x_\ell, r_\ell)\}_\ell)$ ▷ update the parameters including m and Σ

Algorithm 2 $\text{EVALUATE}(x_1, \ldots, x_\lambda)$ for Unconstrained Optimization

1: **for all** $\ell = 1, \ldots, \lambda$ **do** ▷ possibly in parallel
2: execute $s_\ell = \mathcal{S}(x_\ell)$ and evaluate $f_\ell = f(s_\ell)$
3: **end for**
4: $r_\ell = -\frac{1}{2} + \sum_{\ell'=1}^{\lambda} \mathbf{1}\{f_{\ell'} < f_\ell\} + \frac{1}{2}\sum_{\ell'=1}^{\lambda} \mathbf{1}\{f_{\ell'} = f_\ell\}$ for $\ell = 1, \ldots, \lambda$
5: **return** $\{r_\ell\}_\ell$

paper that addresses automatic selection of simulators during search process under constraints. The proposed approach is again invariant to any strictly increasing transformation of the objective and constraint violation functions. We need not care about the balance between the objective value and the constraint violations when defining them. Practically, this is desired especially when there are many constraints, where finding a good balance between f, g_i and h_j is a tedious task and is often impossible.

3. The proposed technique is fully modularized. We can plug the proposed mechanism in place of the ranking mechanism of candidate solutions in ranking-based evolutionary approaches. High modularity is a key to high-affinity of the approach to independently developed state-of-the-art baseline unconstrained optimization problem. To show its high flexibility, we combine the proposed technique to the latest variant of covariance matrix adaptation evolution strategies, namely DD-CMA-ES [1]. We then apply it to a compliance minimization problem [3] as a proof of concept.

2 Existing Algorithmic Components

DD-CMA-ES DD-CMA-ES [1] is the latest variant of covariance matrix adaptation evolution strategies (CMA-ES) that accelerates the covariance matrix adaptation by decomposing the covariance matrix into the variance matrix and the correlation matrix and updating them separately. It is designed for unconstrained optimization

$$\text{minimize: } f(\mathcal{S}(x)) \text{ for } x \in \mathbb{R}^n \quad . \tag{2}$$

A single iteration of DD-CMA-ES is displayed in Algorithm 1 in an abstract manner. DD-CMA-ES generates candidate solutions from the n-variate normal distribution $\mathcal{N}(m, \Sigma)$, where m is the mean vector and Σ is the covariance matrix of the distribution. The covariance matrix Σ is decomposed as $\Sigma = \sigma^2 D \cdot C \cdot D$ and they are updated separately. Generated candidate solutions are evaluated on the objective function (Algorithm 2). Their rankings are computed based on the objective function values. Then, the

Algorithm 3 EVALUATE($x_1, \ldots, x_\lambda$) with ARCH

for all $\ell = 1, \ldots, \lambda$ **do** ▷ possibly in parallel
  $x_\ell^{\text{rep}} = \text{REPAIR}_{\text{pri}}(x_\ell)$ ▷ repaired candidate solution
  $p_\ell = \|x_\ell^{\text{rep}} - x_\ell\|_{\Sigma^{-1}}$ ▷ penalty
  execute $s_\ell = \mathcal{S}(x_\ell^{\text{rep}})$ and evaluate $f_\ell = f(s_\ell)$
end for
$\alpha \leftarrow \text{UPDATE}_{\text{pri}}(\theta)$ ▷ update of the penalty coefficient
$\{r_\ell\}_\ell = \text{RANKING}_{\text{pri}}(\alpha, \{(f_\ell, p_\ell)\}_\ell)$ ▷ eqs. (4) to (6)
return $\{r_\ell\}_\ell$

distribution parameters and other internal parameters, denoted by θ, are update by using the candidate solutions and their rankings. These steps are repeated until a termination condition is satisfied.

In the update step, the recombination weight w_ℓ is assigned to each candidate solution x_ℓ based on its ranking. Let $\bar{w}_1, \ldots, \bar{w}_\lambda$ be the pre-defined weights. If there is no tie in f-values, the ranking values are $0, 1, \ldots, \lambda - 1$. Then, the recombination weight w_ℓ assigned to x_ℓ is simply $\bar{w}_{1+r_\ell}$. If there are ties, a tie candidate solution x_ℓ receives the same recombination weights that are the average of the pre-defined weights $\bar{w}_{1+\sum_{\ell'=1}^{\lambda} \mathbf{1}\{r_{\ell'} < r_\ell\}}, \ldots, \bar{w}_{\sum_{\ell'=1}^{\lambda} \mathbf{1}\{r_{\ell'} \leqslant r_\ell\}}$.

ARCH ARCH [13] is a ranking-based constrained handling technique for optimization under prior constraints

$$\begin{aligned} &\text{minimize: } f(\mathcal{S}(x)) \text{ for } x \in \mathbb{R}^n \\ &\text{subject to: } g_i(x) \leqslant 0 \text{ for } i = 1, \ldots, m_{\text{pri}} \ , \end{aligned} \tag{3}$$

where the input to $\mathcal{S}$ must satisfy the constraints.

Given a candidate solution x, ARCH repairs x to a boundary point of the feasible domain, which is either the nearest feasible solution to x w.r.t. the square Mahalanobis distance $\|x^{\text{rep}} - x\|_{\Sigma^{-1}}^2 = (x^{\text{rep}} - x)^{\mathrm{T}} \Sigma^{-1} (x^{\text{rep}} - x)$ or the nearest feasible solution under the constraints $g_i(x^{\text{rep}}) = 0$ for constraints g_i that are violated by x. Let $x_\ell^{\text{rep}} = \text{REPAIR}_{\text{pri}}(x_\ell)$ be the repaired candidate solution given x_ℓ. Then, the objective function value is computed at x_ℓ^{rep}, denoted by $f_\ell = f(x_\ell^{\text{rep}})$. The penalty for the constraint violation is measured by the Mahalanobis distance between the original and repaired points, $p_\ell = \|x_\ell^{\text{rep}} - x_\ell\|_{\Sigma^{-1}}$. The rankings of the objective value and the penalty value are defined as follows

$$r_\ell^f = -\tfrac{1}{2} + \textstyle\sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{f_{\ell'} < f_\ell\}} + \tfrac{1}{2} \sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{f_{\ell'} = f_\ell\}} \tag{4}$$

$$r_\ell^p = -\tfrac{1}{2} + \textstyle\sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{p_{\ell'} < p_\ell\}} + \tfrac{1}{2} \sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{p_{\ell'} = p_\ell\}} \ . \tag{5}$$

The final ranking of x_ℓ is then computed as

$$r_\ell = r_\ell^f + \alpha \cdot r_\ell^p \ , \tag{6}$$

where α is the penalty coefficient that is adapted based on the penalty of the center m of the search distribution.

Algorithm 4 EVALUATE($x_1, \ldots, x_\lambda$) with MCR-mod

for all $\ell = 1, \ldots, \lambda$ **do** ▷ possibly in parallel
  execute $s_\ell = \mathcal{S}(x_\ell)$ and evaluate $f_\ell = f(s_\ell)$
  $v_\ell^j = \max(h_j(s_\ell), 0)$ for $j = 1, \ldots, m_{\text{pos}}$ ▷ constraint violations
  $N_\ell^v = |\{j : v_\ell^j > 0\}|$ ▷ number of violated constraints
end for
$\{r_\ell\}_\ell = \text{RANKING}_{\text{pos}}(\{(f_\ell, \{v_\ell^j\}_{j=1}^{m_{\text{pos}}}, N_\ell^v)\}_\ell)$ ▷ eqs. (8) and (9)
return $\{r_\ell\}_\ell$

ARCH is implemented as an EVALUATE function in Algorithm 1. We replace Algorithm 2 with Algorithm 3.

MCR-mod MCR-mod [8] is a ranking-based constrained handling technique for optimization under posterior constraints

$$\begin{aligned} &\text{minimize: } f(\mathcal{S}(x)) \text{ for } x \in \mathbb{R}^n \\ &\text{subject to: } h_j(\mathcal{S}(x)) \leqslant 0 \text{ for } j = 1, \ldots, m_{\text{pos}} \ , \end{aligned} \tag{7}$$

where $\mathcal{S}$ accepts any value in $\mathbb{R}^n$ as an input.

For each input x_ℓ, the objective function value $f_\ell = f(\mathcal{S}(x_\ell))$ and the constraint violation $v_\ell^j = \max(h_j(\mathcal{S}(x_\ell)), 0)$ are computed. Then calculate the number N_ℓ^v of the violated constraints $N_\ell^v = |\{j : v_\ell^j > 0\}|$. MCR-mod defines the rankings of the objective values, constraint violations, and the number of violated constraints as follows

$$r_\ell^f = \textstyle\sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{f_{\ell'} < f_\ell\}}, \quad r_\ell^{v_j} = \sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{v_{\ell'}^j < v_\ell^j\}}, \quad r_\ell^{N^v} = \sum_{\ell'=1}^{\lambda} \mathbf{1}_{\{N_{\ell'}^v < N_\ell^v\}} \ . \tag{8}$$

Let $\beta = |\{\ell' : N_{\ell'}^v = 0\}|/\lambda$ be the ratio of the feasible candidate solutions among λ current candidate solutions. Then, the final ranking is computed as follows

$$r_\ell = \beta \cdot r_\ell^f + (1 - \beta) \left(r_\ell^{N^v} + \tfrac{1}{m_{\text{pos}}} \textstyle\sum_{j=1}^{m_{\text{pos}}} r_\ell^{v_j} \right) \ . \tag{9}$$

MCR-mod can be implemented as an EVALUATE function. We replace Algorithm 2 with Algorithm 4.

Adaptive Simulator Selection Adaptive objective selection mechanism [2] selects the simulator $\mathcal{S}_k$ among m_{sim} available ones, $\mathcal{S}_1, \ldots, \mathcal{S}_{m_{\text{sim}}}$, for an unconstrained multi-fidelity optimization problem

$$\text{minimize: } f(\mathcal{S}_{m_{\text{sim}}}(x)) \text{ for } x \in \mathbb{R}^n \ . \tag{10}$$

As discussed in the introduction, we assume that $\mathcal{S}_k$ with a lower fidelity index k is less accurate and computationally cheaper.

The adaptive objective selection mechanism works as a wrapper of EVALUATE in Algorithm 2 with $\mathcal{S}_1, \ldots, \mathcal{S}_{m_{\text{sim}}}$, summarized in Algorithm 5, where EVALUATE_k function has the same functionality as EVALUATE in Algorithm 2 except that $\mathcal{S}$ is replaced

Algorithm 5 EVALUATE$(x_1, \ldots, x_\lambda)$ with Adaptive Simulator Selection

$\{r_\ell\}_\ell = \text{EVALUATE}_k(x_1, \ldots, x_\lambda)$
$T \leftarrow T + t_k$, where t_k is the total time for $\mathcal{S}_k$ call above
if $k < m_{\text{sim}}$ and $T > \gamma \cdot T_+$ **then** ▷ check for fidelity index increment
  $\{r_\ell^+\}_\ell = \text{EVALUATE}_{k+1}(x_1, \ldots, x_\lambda)$
  $T_+ \leftarrow T_+ + t_{k+1}$, where t_{k+1} is the total time for $\mathcal{S}_{k+1}$ call above
  compute Kendall's rank coefficient τ for $\{(r_\ell, r_\ell^+)\}_\ell$
  update $\langle\tau\rangle_+ \leftarrow \langle\tau\rangle_+ + c_\tau^+(\tau - \langle\tau\rangle_+)$
else if $1 < k$ and $T > \gamma \cdot T_-$ **then** ▷ check for fidelity index decrement
  $\{r_\ell^-\}_\ell = \text{EVALUATE}_{k-1}(x_1, \ldots, x_\lambda)$
  $T_- \leftarrow T_- + t_{k-1}$, where t_{k-1} is the total time for $\mathcal{S}_{k-1}$ call above
  compute Kendall's rank coefficient τ for $\{(r_\ell, r_\ell^-)\}_\ell$
  update $\langle\tau\rangle_- \leftarrow \langle\tau\rangle_- + c_\tau^-(\tau - \langle\tau\rangle_-)$
end if
if $\langle\tau\rangle_+ < \tau_{\text{thresh}}$ **then**
  $k \leftarrow k + 1$ and reset $T = T_+ = T_- = 0$ and $\langle\tau\rangle_+ = \langle\tau\rangle_- = 0$
else if $\langle\tau\rangle_- > \tau_{\text{thresh}} + \tau_{\text{margin}}$ **then**
  $k \leftarrow k - 1$ and reset $T = T_+ = T_- = 0$ and $\langle\tau\rangle_+ = \langle\tau\rangle_- = 0$
end if
return $\{r_\ell\}_\ell$

with $\mathcal{S}_k$. Let k be the current fidelity index. The main idea for the adaptation of the fidelity index is to check if the objective function value sequences evaluated with $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ (or $\mathcal{S}_{k-1}$) have a low rank correlation (or a high rank correlation). If there is a high correlation between $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ (or $\mathcal{S}_{k-1}$), there is no need to use the one with a higher fidelity index as they results in similar rankings of candidate solutions and a computationally cheaper simulator is preferred. To avoid spending too much time to check the rank correlation, the execution time for each simulator is recorded and we compute the rank correlation only if the execution time for the current simulator $\mathcal{S}_k$ is γ times more than the execution time for $\mathcal{S}_{k+1}$ or $\mathcal{S}_{k-1}$.

3 Proposed Approach

Combination of ARCH and MCR-mod As a first step, we combine DD-CMA-ES with ARCH and MCR-mod to tackle optimization problems with prior and posterior constraints (1) with a single simulator $\mathcal{S}$. We combine ARCH (Algorithm 3) and MCR-mod (Algorithm 4) in a way described in Algorithm 6. First, it repairs candidate solutions for prior constraints and compute the penalty values. The repaired candidate solutions are given to the simulator and their objective function values and posterior constraint violations are computed. Then, the final rankings are computed.

How to assign the ranking to each point is the only part that requires a careful decision make for the combination of ARCH and MCR-mod to work well. A straightforward way to construct the final ranking may simply add all the rankings that appear in eqs. (6) and (9). However, since the adaptation of α in ARCH is designed to balance

Algorithm 6 EVALUATE$(x_1, \ldots, x_\lambda)$ with ARCH and MCR-mod

1: **for all** $\ell = 1, \ldots, \lambda$ **do** ▷ possibly in parallel
2: repair $x_\ell^{\text{rep}} = \text{REPAIR}_{\text{pri}}(x_\ell)$
3: compute $p_\ell = \|x_\ell^{\text{rep}} - x_\ell\|_{\Sigma^{-1}}$
4: execute $s_\ell = \mathcal{S}(x_\ell^{\text{rep}})$ and evaluate $f_\ell = f(s_\ell)$
5: evaluate $v_\ell^j = \max(h_j(s_\ell), 0)$ for all $j = 1, \ldots, m_{\text{pos}}$
6: calculate $N_\ell^v = |\{j : v_\ell^j > 0\}|$
7: **end for**
8: update $\alpha \leftarrow \text{UPDATE}_{\text{pri}}(\theta)$
9: compute $\{r_\ell^h\}_\ell = \text{RANKING}_{\text{pos}}(\{(f_\ell, v_\ell^1, \ldots, v_\ell^{m_{\text{pos}}}, N_\ell^v)\}_\ell)$
10: compute $\{r_\ell\}_\ell = \text{RANKING}_{\text{pri}}(\alpha, \{(r_\ell^h, p_\ell)\}_\ell)$
11: **return** $\{r_\ell\}_\ell$

r^f and r^p, the additional rankings r^{N^v} and r^{v_j} will disturb the adaptation mechanism. To avoid it, we first compute the rankings by MCR-mod (9), then we pass them to ARCH as if the rankings computed by MCR-mod are the objective function values.

Adaptive Simulator Selection for Constrained Optimization We now combine Algorithm 5 with Algorithm 6 for multi-fidelity constrained optimization. Since Algorithm 5 is based only on the rankings of the candidate solutions, we simply replace EVALUATE_k function in Algorithm 5 with EVALUATE in Algorithm 6 using $\mathcal{S}_k$. Arguably, this is the most natural combination of these components. The adaptive simulator selection tries to increase or decrease the fidelity index if the resulting rankings of the candidate solutions are different for different simulators, which results in affecting the behavior of the baseline algorithm, here DD-CMA-ES. Even if the rankings of the objective or the rankings of the constraint violations are significantly different for different simulators, the behavior of the algorithm may not change drastically if α in eq. (6) is very large. Then, a computationally cheaper simulator is preferred. Our design choice is to reflect this demand.[5]

Discussion on the Choice of Algorithmic Components For posterior constraint handling, there are other existing approaches such as augmented Lagrangian (AL) approaches [6]. Replacing MCR-mod with AL (in an appropriate way) needs a careful decision since AL aggregates f and h_j with adaptive coefficients and its adaptation

[5] We remark that if a simulator $\mathcal{S}_k$ is computationally very cheap to evaluate, the computational time for EVALUATE_k may be dominated by the time for $\text{REPAIR}_{\text{pri}}$, which is computationally demand compared to the other parts as it internally solves an minimization problem. In such a case, it may be more reasonable to record the computational time for EVALUATE_k to t_k, rather than the total time of $\mathcal{S}_k$ in EVALUATE_k. This is also expected to affect the performance of the proposed strategy when it is implemented on a parallel machine, especially when the computational time for Line 2 to 6 in Algorithm 6 differs substantially among $\ell = 1, \ldots, \lambda$. It is highly involved with parallel computation and is out of scope of this paper. Further studies on this line is left for future work.

mechanism can be broken due to the existence of g_i and the repair operator for them. Moreover, since AL is not invariant to strictly increasing transformation of f and h_j, we observe an undesired behavior that the adaptation of AL fails if the ranges of f and h_j values are hugely different. On the other hand, if the original MCR [12] is preferred to MCR-mod, or once their improvements are proposed, one can easily replace them.[6]

For prior constraint handling, ARCH is one of the most generalized approaches that can treat both linear and non-linear constraints. Limiting the focus on box constraint, there are several approaches such as mirroring, death penalty, resampling, and adaptive penalty method [9]. Even for box constraints, ARCH is reported a better performance over other box constraint handling techniques. Moreover, ARCH itself is reported as invariant to strictly increasing transformation of f and g_i, resulting in the invariance of the proposed approach mentioned in the introduction, and invariant to affine transformation of the objective function.

For adaptive simulator selection, one can use an approach proposed in [16]. Though it is proposed for PSO, the idea can be translated to other algorithms such as CMA-ES. One reason of our choice of [2] is that the method in [16] requires one execution of the highest fidelity simulator to check if the fidelity level should be increased or not. It is undesired when the highest accuracy simulator is unnecessarily accurate and computationally expensive, which happens in practice as we often do not know how accurate the simulator should be in advance. Another reason is that it is non-trivial to generalize the fidelity check mechanism to constrained optimization scenario.

4 Compliance Minimization Problem

A compliance minimization problem is an example problem that can be considered as the multi-fidelity optimization of form (1). A compliance minimization is a classical topology optimization. Given a design space, the material distribution over the design space is parameterized by x. The objective is to find x achieving the optimal material distribution in terms of compliance under different constraints. The objective is typically computed through a finite-element method, and its granularity is a user-parameter. A finer granularity leads to a more accurate but computational more expensive simulation. Hence, it forms a multi-fidelity optimization. Here we describe the compliance minimization problem formulated in [3] with Normalized Gaussian network (NGnet) based design representation [14].

[6] Both the original MCR and MCR-mod rank the candidate solutions based only on the constraint violations when the candidates are all infeasible. On one hand it is nice as the search distribution is forced to sample feasible solutions with high probability. On the other hand, we empirically observe that it is trapped by a sub-optimal feasible point since it tends to satisfy the constraints at the beginning of the search without considering the objective function landscape. A possible solution is to *include the objective function as a posterior constraint as* $h(\mathcal{S}(x)) = f(\mathcal{S}(x)) - f_{\mathrm{thre}}$, where f_{thre} is the worst allowable objective value. In engineering optimization, it is often the case that the objective is to find a solution that satisfies all demands. In such a situation, f_{thre} is available.

NGnet based Material Distribution Representation The design space is a two dimensional rectangular area $\mathcal{D} = [0, L_h] \times [0, L_v]$. The design space is discretized by square finite elements with length δ. The (i_h, i_v)-th element takes a value in $\{0, 1\}$, representing that there is a material (if 1) or not (if 0).

NGnet based representation places n Gaussian basis functions on the design space and the value at each element is represented by the sum of the basis function values evaluated at each place. Let $(\mu_h^d, \mu_v^d) \in \mathcal{D}$ and $(\sigma_h^d, \sigma_v^d) \in \mathbb{R}^2_{>0}$ be the center and the coordinate-wise standard deviation of the dth basis function, where the basis functions are pre-located by a user. Let $x = ([x]_1, \ldots, [x]_n) \in [-1, 1]^n$ be the vector consisting of the height parameters of the basis functions, which is the parameter vector to be optimized. The output of the NGnet is then computed as follows

$$\phi(h, v; x) = \frac{\sum_{d=1}^n [x]_d \exp\left(-\frac{(h-\mu_h^d)^2}{2(\sigma_h^d)^2} - \frac{(v-\mu_v^d)^2}{2(\sigma_v^d)^2}\right)}{\sum_{d=1}^n \exp\left(-\frac{(h-\mu_h^d)^2}{2(\sigma_h^d)^2} - \frac{(v-\mu_v^d)^2}{2(\sigma_v^d)^2}\right)} \quad . \tag{11}$$

It takes value in $[-1, 1]$. The value of the (i_h, i_v)-th element is $\mathbf{1}\{\phi(h, v; x) \geqslant 0\}$ evaluated at the center of the element, $h = \delta \cdot (i_h - 0.5)$ and $v = \delta \cdot (i_v - 0.5)$.

In practice the distribution of the bases should reflect some prior knowledge so that one can represent a more complex structure in one place than the others. In this paper, we place them on a grid pattern. Let n_h and $n_v = n/n_h$ be the numbers of the basis functions in horizontal and vertical axes, respectively. Let $\Delta_h = L_h/n_h$ and $\Delta_v = L_v/n_v$. The center μ^d and the standard deviation σ^d of the basis function of index $d = d_h + (d_v - 1) * n_h$ is $\mu_h^d = \Delta_h * (d_h - 0.5)$, $\mu_v^d = \Delta_v * (d_v - 0.5)$, $\sigma_h^d = r \cdot \Delta_h$ and $\sigma_v^d = r \cdot \Delta_v$, where $r > 0$ is the scaling parameter.

Compliance Minimization Let $\Phi_\delta(x)$ be the $L_v/\delta \times L_h/\delta$ dimensional matrix whose (i_h, i_v)-th element is $\mathbf{1}\{\phi(h, v; x) \geqslant 0\}$ evaluated at the center of the element, $h = \delta \cdot (i_h - 0.5)$ and $v = \delta \cdot (i_v - 0.5)$. This corresponds to the material distribution. A finite-element method is performed with $\Phi_\delta(x)$ as an input, and the compliance is calculated based on the simulation results. The compliance of the design $\Phi_\delta(x)$ is the objective function value $f(\mathcal{S}(x))$. The design variable x must live in $[-1, 1]^n$, hence we have $m_{\text{pri}} = 2n$ prior constraints $g_i(x) = x - 1$ (for upper bound) and $g_{n+i}(x) = -x - 1$ (for upper bound) for $i = 1, \ldots, n$. A posterior constraint is the volume fraction constraint: the number of the elements of $\Phi_\delta(x)$ being 1 is no greater than $\text{volfrac}_{\text{max}} \cdot (L_v/\delta) \cdot (L_h/\delta)$. The posterior constraint violation is defined as the volume fraction minus the maximum allowed volume fraction $\text{volfrac}_{\text{max}} \in (0, 1]$.

Both the objective value and the posterior constraint violation depend on δ. A smaller δ (a finer granularity) leads to a better accuracy at the risk of longer computational time. Since δ is a user parameter, we can prepare multiple simulators with different δ, leading to multi-fidelity constrained optimization (1).

5 Experiments

We applied the proposed approach to the above mentioned compliance minimization problem to see how the performance-per-cost is improved by the the proposed multi-

fidelity approach. The baselines are the performance graph of the proposed algorithm without adaptive simulator selection solving the constrained optimization using $\mathcal{S}_k$ (for $k = 1, \ldots, m_{\text{sim}}$).

We set the problem parameters as follows: numbers of basis functions in horizontal axis $n_h = 24$ and vertical axis $n_v = 8$, number of elements in horizontal axis $(L_h/\delta) = 12k$ and vertical axis $(L_v/\delta) = 4k$ for simulator $\mathcal{S}_k$, the volume fraction is $\text{volfrac}_{\text{max}} = 0.4$ and the scaling parameter for NGnet is $r = 1$. The number of design variables is $n = n_h \cdot n_v = 192$. All the hyper-parameters for the proposed algorithm are set by following the original references [1,2,13], except that we set $c_\tau^+ = 0.1, 0.3, 0.5.0.7, 0.9$, instead of $c_\tau^+ = 1$ to make the fidelity adaptation more stable. We set $m_{\text{sim}} = 20$.

The experiments were implemented in PYTHON. For each setting, 10 independent runs with different random number generator seeds were conducted in parallel on Intel(R) Core(TM) i9-7900X CPU (10 cores) with 4 8192MB DDR4 RAMs (2133MT/s), and each run was limited to 24 hours (86400 seconds) on a single thread. The code to reproduce the results of this paper is available at `url-will-be-added-after-acceptance`.

Performance Assessment The performance assessment must be done on $\mathcal{S}_{m_{\text{sim}}}$ as our objective is to locate the optimal solution of (1) with $\mathcal{S}_{m_{\text{sim}}}$. Ideally we can re-evaluate all the candidate solutions on $\mathcal{S}_{m_{\text{sim}}}$ and show the graph of the best-so-far objective value among the feasible solutions. However, it is computationally very expensive, and is not possible in practice. To have a fair comparison, at each iteration t we pick the best ranked repaired candidate solution, $x_{t,\text{rec}}^{\text{rep}}$, and regard it as the recommendation point at iteration t. We re-evaluate the recommendation point on $\mathcal{S}_{m_{\text{sim}}}$, check if the recommendation point is feasible, and record its objective function value $f_{t,\text{rec}} = f(\mathcal{S}_{m_{\text{sim}}}(x_{t,\text{rec}}^{\text{rep}}))$ if it is feasible. Note that g_i are independent but h_j depends on k, hence it is not guaranteed that a feasible solution on $\mathcal{S}_k$ is also feasible on $\mathcal{S}_{m_{\text{sim}}}$. Figure 1 shows percentiles of the best-so-far $f_{t,\text{rec}}$ values, plotted against the elapsed CPU time excluding the time for re-evaluation.

Results We first summarize the results obtained by optimizing the fixed fidelity problems with $k = 4, 7, 11, 14, 17, 20$. A lower fidelity simulator resulted in a quicker drop of the objective value and a higher (worse) final function value as it is trapped at a local optimum defined on $\mathcal{S}_k$, which is not necessarily a local optimal solution on $\mathcal{S}_{m_{\text{sim}}}$. A higher fidelity simulator tended to result in a better final performance while wasting the time at the beginning, where f-values are comparatively high ($\gg 10^4$) and h_j-constraints are often violated. Due to the multimodality, the median and 25%ile performances at the end of the optimization process were not the best for $k = 20$ ($k = 11$ was the best), but the 10%ile performance was monotonic with respect to k (equal for $k \geqslant 14$).

The adaptive fidelity approach decreased the objective function value as fast as the results on low fidelity simulators did and kept improving the objective function value till the end of the experiments, whereas the results on low fidelity simulators stopped improving. The observed trend is similar to those reported in [2], where the constraints are not taken into account. The new insight here is that the adaptive simulator selection mechanism works as well with constraint handling technique proposed in this paper.

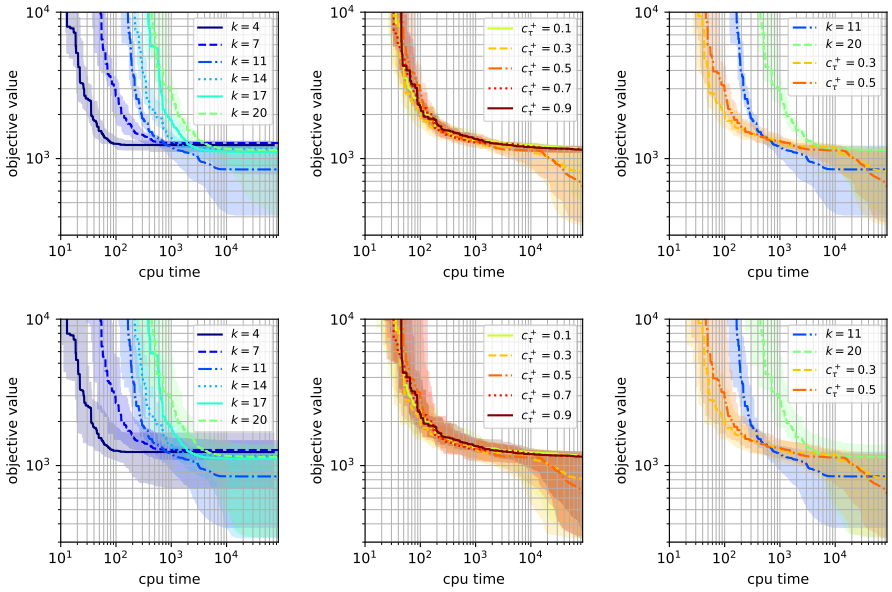
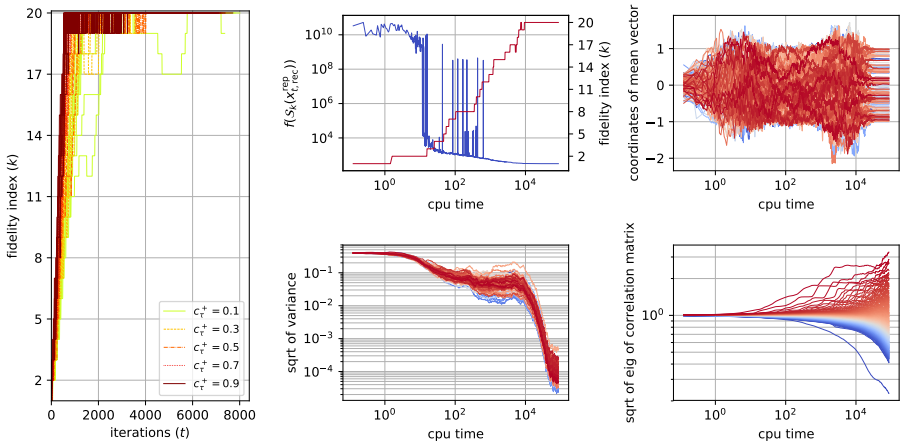


Fig. 1: Best-so-far objective function values of feasible recommendation points evaluated on $\mathcal{S}_{m_{\text{sim}}}$. Median (solid line) and percentile range (transparent area, 25%-75%ile (top) and 10%-90%ile (bottom)) of 10 independent runs are shown for each cpu time.

The sensitivity of the newly-introduced hyper-parameter, c_τ^+, was investigated. The proposed strategy was run with $c_\tau^+ = 0.1, 0.3, 0.5, 0.7, 0.9$. We observed that all the median graphs are inside 10%-90%ile ranges of others. Figure 2a shows that the fidelity index increased later for a smaller c_τ^+ as we expected. In other words, a smaller c_τ^+ results in spending more time on low fidelity simulators. It is advantageous for adaptation of the parameters of the baseline search algorithm (m and $\Sigma = \sigma^2 D \cdot C \cdot D$ for the case of DD-CMA-ES, see Figure 2b) as long as the landscapes of the problems on low fidelity simulators translate to those on high fidelity simulators. However, if the landscape is multimodal and an absorbing local optimum on a low fidelity simulator is not a good local optimum on a high fidelity simulator, spending more time on low fidelity simulators may be more likely to be trapped by a poor local optimum. Such a situation was observed in Figure 1 for $c_\tau^+ = 0.1$.

6 Discussion

Many engineering optimization problems can be framed as multi-fidelity optimization as formulated in this paper. However, few researches have been conducted so far in evolutionary computation communities. Reference [16] proposes benchmark problems for multi-fidelity optimization, but constraints are not taken into account. Constraints appear in most of engineering optimization. Therefore, a benchmark suite with prior and



(a) Fidelity level (b) Typical run of the proposed approach with $c_\tau^+ = 0.3$

Fig. 2: (a) Fidelity index k during the optimization. Results of 10 independent runs are shown for each setting. (b) Typical run result of the proposed approach. Shown are $f(\mathcal{S}_k(x^{\mathrm{rep}}_{t,\mathrm{rec}}))$, k, the square roots of the variances ($\sigma^2 D^2$), the square roots of the eigenvalues of the correlation matrix (C), and each coordinates of the mean vector m.

posterior constraints is highly desired for evaluation and development of multi-fidelity search strategies.

At the same time, the theoretical foundation for the multi-fidelity optimization problem and its approaches should be investigated to guarantee the performance or to reveal the limitation. Some theoretical investigation has been done in [2], revealing when the approach fails to adapt the fidelity level and how to avoid it. In the multi-fidelity optimization, it is assumed that a low fidelity simulator and a high fidelity simulator are somewhat similar, without formally defining the similarity. The assumption on the similarity between simulators can be different between approaches, but they should be reasonable in practice. The development of the benchmark suite should be done in parallel with the theoretical foundation to have a maximal coverage of the difficulties of multi-fidelity optimization with a minimal set of test problems.

This paper proposes an approach to deal with multi-fidelity optimization with prior and posterior constraints. As a proof of concept, the proposed approach has been applied to a multi-fidelity compliance minimization problem, showing its efficacy over the fixed fidelity formulation. Since the compliance minimization shares characteristics with other topology optimization problems, we expect that the proposed approach is applied to other topology optimization problems as well. However, as a generic framework for multi-fidelity constrained optimization, the evaluation of the proposed approach is definitely missing. The above mentioned benchmark suite and theoretical foundation are desired for this purpose.

Acknowledgement. This work is partially supported by JSPS KAKENHI Grant Number 19H04179 and 19J21892.

References

1. Akimoto, Y., Hansen, N.: Diagonal acceleration for covariance matrix adaptation evolution strategies. Evolutionary Computation (2019). https://doi.org/10.1162/evco_a_00260, to appear
2. Akimoto, Y., Shimizu, T., Yamaguchi, T.: Adaptive objective selection for multi-fidelity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 880–888. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321709
3. Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B.S., Sigmund, O.: Efficient topology optimization in matlab using 88 lines of code. Structural and Multidisciplinary Optimization **43**(1), 1–16 (2011). https://doi.org/10.1007/s00158-010-0594-7
4. Arnold, D.V.: An active-set evolution strategy for optimization with known constraints. In: Parallel Problem Solving from Nature – PPSN XIV. pp. 192–202. Springer International Publishing, Cham (2016). https://doi.org/doi.org/10.1007/978-3-319-45823-6_18
5. Arnold, D.V.: Reconsidering constraint release for active-set evolution strategies. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 665–672. GECCO '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071294
6. Atamna, A., Auger, A., Hansen, N.: Augmented lagrangian constraint handling for cma-es — case of a single linear constraint. In: Parallel Problem Solving from Nature – PPSN XIV. pp. 181–191. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_17
7. Digabel, S.L., Wild, S.M.: A taxonomy of constraints in simulation-based optimization. arXiv:1505.07881 (2015), https://arxiv.org/abs/1505.07881
8. Dwianto, Y.B., Fukumoto, H., Oyama, A.: On improving the constraint-handling performance with modified multiple constraint ranking (mcr-mod) for engineering design optimization problems solved by evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 762–770. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321808
9. Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation **13**(1), 180–197 (2009). https://doi.org/10.1109/TEVC.2008.924423
10. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evol. Comput. **11**(1), 1–18 (2003). https://doi.org/10.1162/106365603321828970
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
12. de Paula Garcia, R., de Lima, B.S.L.P., de Castro Lemonge, A.C., Jacob, B.P.: A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. Computers & Structures **187**, 77 – 87 (2017). https://doi.org/https://doi.org/10.1016/j.compstruc.2017.03.023
13. Sakamoto, N., Akimoto, Y.: Adaptive ranking based constraint handling for explicitly constrained black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 700–708. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321717
14. Sato, T., Watanabe, K., Igarashi, H.: Multimaterial topology optimization of electric machines based on normalized gaussian network. IEEE Transactions on Magnetics **51**(3), 1–4 (2015). https://doi.org/10.1109/TMAG.2014.2359972

15. Spettel, P., Beyer, H., Hellwig, M.: A covariance matrix self-adaptation evolution strategy for optimization under linear constraints. IEEE Transactions on Evolutionary Computation **23**(3), 514–524 (2019). https://doi.org/10.1109/TEVC.2018.2871944
16. Wang, H., Jin, Y., Doherty, J.: A generic test suite for evolutionary multifidelity optimization. IEEE Transactions on Evolutionary Computation **22**(6), 836–850 (2018). https://doi.org/10.1109/TEVC.2017.2758360