Bilevel Optimization by Conditional Bayesian Optimization*

Vedat Dogan[1][0000−0002−3807−5425] and Steven Prestwich[2][0000−0002−6218−9158]

[1] Confirm Centre for Smart Manufacturing, School of Computer Science and Information Technology, University College Cork, Ireland
vedat.dogan@cs.ucc.ie
[2] Insight Centre for Data Analytics, School of Computer Science and Information Technology, University College Cork, Ireland
s.prestwich@cs.ucc.ie

Abstract. Bilevel optimization problems have two decision-makers: a leader and a follower (sometimes more than one of either, or both). The leader must solve a constrained optimization problem in which some decisions are made by the follower. These problems are much harder to solve than those with a single decision-maker, and efficient optimal algorithms are known only for special cases. A recent heuristic approach is to treat the leader as an expensive black-box function, to be estimated by Bayesian optimization. We propose a novel approach called ConBaBo to solve bilevel problems, using a new conditional Bayesian optimization algorithm to condition previous decisions in the bilevel decision-making process. This allows it to extract knowledge from earlier decisions by both the leader and follower. We present empirical results showing that this enhances search performance and that ConBaBo outperforms some top-performing algorithms in the literature on two commonly used benchmark datasets.

Keywords: Bilevel Optimization · Conditional Bayesian Optimization · Stackelberg Games · Gaussian Process

1 Introduction

Many real-world optimization and decision-making processes are hierarchical: decisions taken by one decision-maker must consider the reaction of another decision-maker with their own objective and constraints. In this work, we consider non-cooperative games called Stackelberg Games [37], which are sequential non-zero-sum games with two players. The first player is called the *leader* and the second player is called the *follower*. We shall represent the leader decision by x_u and the follower response by x_l^*. A decision pair x_u, x_l^* represents a choice by the leader and an optimal feasible solution of the follower. The structure of Stackelberg games is asymmetric: the leader has perfect knowledge about the follower's

* This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 16/RC/3918.

objective and constraints, while the follower must first observe the leader's decisions before making its own optimal decisions. Any feasible bilevel solution should contain an optimal solution for the follower problem. The mathematical modelling of these games leads to nested problems called *bilevel optimization problems*. In these problems, the lower-level (follower) optimization problem is a constraint of the upper-level (leader) problem. Because the lower-level solution must be optimal given the upper-level decisions, these optimization problems are hard to solve. Bilevel optimization is known to be strongly NP-hard and it is known that even evaluating a solution for optimality is NP-hard [39]. There are several approaches proposed in the literature for solving bilevel problems. Most focus on special cases, for example, a large set of exact methods was introduced to solve small linear bilevel optimization problems. Another approach is to replace the lower-level problem with its Karush-Kuhn-Tucker conditions, reducing the bilevel problem to a single-level optimization problem [4]. A popular approach is to use nested evolutionary search to explore both parts of the problem [32], but this is computationally expensive and does not scale well.

Bayesian optimization [12] is another possible approach. It is designed to solve (single-level) optimization problems in which the calculation of the objective is very expensive so the number of such calculations should be minimized. This is done via an acquisition function which is learned as optimization proceeds. This function approximates the upper-level objective with increasing accuracy and is also used to select each upper-level decision. The solution of a lower-level problem could be viewed as an expensive objective evaluation, making Bayesian optimization an interesting approach to bilevel optimization.

Several kinds of applications can be modelled as bilevel optimization problems. In the toll setting problem [9] the authority acts as a leader and the network users act as followers, and the authority aims to optimize the tolls for a network of roads. The authority's toll price decision can be improved by taking into account the network users' previous acts. In environmental economics [35], an authority might want to tax an organization or individual that is polluting the environment. The authority acts as the leader and the polluting entity acts as the follower. If the authority uses the knowledge of previous acts of the polluting entity, then the authority can better regulate its tax policy.

In this work we propose an improved Bayesian approach to bilevel optimization, using the Gaussian process surrogate model with *conditional setting*. In [25] the Conditional Bayesian Optimization (ConBO) algorithm improved the knowledge gradient (KG) acquisition function by using a conditional setting. During the bilevel optimization process, it is important to know how the follower reacts to the leader's decisions, and we improved the algorithm by conditioning the follower's decisions. If we approach the leader problem as a black box and use previous leader decisions with the follower's best responses and leader fitness, conditioning makes us more likely to choose the next leader decision wisely, thus reaching optimality more quickly. Bayesian optimization with the Gaussian process embeds an acquisition function for determining the most promising areas to explore during the optimization process. The benefit of the acquisition function

is to reduce the number of function evaluations. The results show that conditioning speeds up the search for the optimal region during bilevel optimization.

The rest of the paper is organized as follows. Section 2 surveys related work on solution methods for bilevel optimization problems. The preliminaries for the proposed ConBaBo algorithm are discussed in Section 3 The algorithm is explained in Section 4. In Section 5, we present the experimental details and results and compare them with the state-of-the-art. Finally, Section 6 is devoted to conclusions and future directions of research.

2 Background

A bilevel optimization problem is a sequential nested optimization process in which each level is controlled by a different decision-maker. In the context of Game Theory, bilevel problems are Stackelberg games [37]. They were introduced by J. Bracked and J. McGill, and a defence application was published by the same authors in the following year [7]. Bilevel problems were modelled as mathematical programs at this time.

A considerable number of exact approaches have been applied to bilevel problems. Karush-Kuhn-Tucker conditions [4] can be used to reformulate a bilevel problem to a single-level problem. Penalty functions compute the stationary points and local optima. Vertex enumeration has been used with a version of a Simplex method [6]. Gradient information for the follower problem can be extracted for use by the leader objective function [28]. In terms of integer and mixed integer bilevel problems, branch-and-bound [5] and parametric programming approaches have been applied to solve bilevel problems [19]. Because of the inefficiency of exact methods on complex bilevel problems, meta-heuristics have been considered solvers.

Several kinds of meta-heuristics have been applied to bilevel problems in the literature. Four existing categories have been published in [38]: the nested sequential approach [18], the single-level transformation approach, the multi-objective approach [27], and the co-evolutionary approach [21]. An algorithm based on a human evolutionary model for non-linear bilevel problems [23], and the Bilevel Evolutionary Algorithm based on Quadratic approximations (BLEAQ), have been proposed by [33]. This is another work that attempts to reduce the number of follower optimizations. The algorithm approximates the inducible region of the upper-level problem through the feasible region of the bilevel problem. In [26] they consider a single optimization problem at both levels. They proposed the Sequential Averaging Method (SAM) algorithm. In different recent works [29] they used a truncated back-propagation approach for approximating the (stochastic) gradient of the upper-level problem. Basically, they use a dynamical system to model an optimization algorithm that solves the lower-level problem and replaces the lower-level optimal solution. Another work [13] developed a two-timescale stochastic approximation algorithm (TTSA) for solving a bilevel problem, assuming the follower problem is unconstrained and strongly convex, and the leader is a smooth objective function.

Bayesian Optimization for Bilevel Problems (BOBP) [16] uses Bayesian optimization to approximate the inducible region. Multi-objective acquisition approach is presented in [11]. In this work, bilevel programming problems have been solved by using priors in Gaussian processes to approximate the upper-level objective functions. Gaussian processes in Bayesian optimization make it possible to choose the next point wisely and use the previous iterations as knowledge to guide the optimization process. We use a conditional Bayesian optimization algorithm for bilevel optimization. We show empirically that conditioning can dramatically reduce the number of function evaluations for the upper-level problem. Many practical problems can be modelled and solved as Stackelberg games. In the field of economics [35] these include principal agency problems and policy decisions. Hierarchical decision-making in management [3] and engineering and optimal structure design [17] are other practical applications. Network design and toll setting problems are the most popular applications in the field of transportation [24]. Finding optimal chemical equilibria, planning the pre-positioning of defensive missile interceptors to counter an attacking threat, and interdicting nuclear weapons are further applications [8].

3 Preliminaries

The description of the algorithm will be divided into three parts. Firstly, we explain bilevel programming problems and structure. Secondly, we discuss conditional Bayesian optimization and Gaussian process settings. Thirdly, we explain the proposed ConBaBo algorithm.

Bilevel Optimization Problems. For the upper-level objective function $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and lower-level objective function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, bilevel optimization problem can be defined as;

$$\begin{aligned} \underset{x_u, x_l}{\text{Minimize}} \;\; & F(x_u, x_l) \\ \text{s.t.} \;\; & x_l \in \underset{x_l}{\operatorname{argmin}} \{ f(x_u, x_l) : g_j(x_u, x_l) \leq 0, \; j = 1, 2, \ldots, J \} \\ & G_k(x_u, x_l) \leq 0, \; k = 1, 2, \ldots, K \end{aligned} \tag{1}$$

where $x_u \in \mathcal{X}_U, x_l \in \mathcal{X}_L$ are vector-valued upper-level and lower-level decision variables in decision spaces. G_k and g_j represent the constraints of the bilevel problem. Because the lower-level decision maker depends on the upper-level variables, for every decision x_u there is a follower-optimal decision x_l^* [34]. In bilevel optimization, a decision point $x^* = (x_u, x_l^*)$ is feasible for the upper-level only if it satisfies all the upper-level constraints and the vector x^* is an optimal solution to the lower-level problem with the upper-level decision as a parameter.

Bayesian Optimization and Gaussian Process. Bayesian optimization [15] is a method used to optimize black-box functions that are expensive to evaluate. BO uses a probabilistic surrogate model, commonly Gaussian Process [36], to obtain a posterior distribution $\mathbb{P}(\mathbf{f}|\mathcal{D})$ over the objective function $\mathbf{f}$ given the

observed data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ where n represents the number of observed data. The surrogate model is assisted by an acquisition function to choose the next candidate or set of candidates $X = \{x_j\}_{j=1}^{q}$ where q is the batch size of promising points found during optimization. The objective function is expensive but the surrogate-based acquisition function is not, so it can be optimized much more quickly than the true function to yield X. Let us assume that we have a set of collection points $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$ and objective function values of these points $\{f(x_1), \ldots, f(x_n)\}$. After we observe n points, the mean vector is obtained by evaluating a mean function μ_0 at each decision point x and the covariance matrix by evaluating a covariance function or kernel Σ_0 at each pair of point x_i and x_j. The resulting prior distribution on $\{f(x_1), \ldots, f(x_n)\}$ is,

$$f(x_{1:n}) \sim \mathcal{N}(\mu_0(x_{1:n}), \Sigma_0(x_{1:n}, x_{1:n})) \tag{2}$$

where $x_{1:n}$ indicates the sequence $x_1, \ldots, x_n$, $f(x_{1:n}) = \{f(x_1), \ldots, f(x_n)\}$, $\mu_0(x_{1:n}) = \{\mu_0(x_1), \ldots, \mu_0(x_n)\}$ and $\Sigma_0(x_{1:n}, x_{1:n}) = \{\Sigma_0(x_1, x_1), \ldots, \Sigma_0(x_1, x_k)$ $; \ldots; \Sigma_0(x_n, x_1), \ldots, \Sigma_0(x_n, x_n)\}$. Let us suppose we wish to find a value of $f(X)$ at some new point X. Then we can compute the conditional distribution of $f(X)$ given the observations

$$f(X)|f(x_{1:n}) \sim \mathcal{N}(\mu_n(X), \sigma_n^2(X)) \tag{3}$$

$$\mu_n(X) = \Sigma_0(X, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}(f(x_{1:n}) - \mu_0(x_{1:n})) + \mu_0(X) \tag{4}$$

$$\sigma_n^2(X) = \Sigma_0(X, X) - \Sigma_0(X, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}\Sigma_0(x_{1:n}, X) \tag{5}$$

where $\mu_n(X)$ is the posterior mean and can be viewed as the prediction of the function value. $\sigma_n^2(X)$ is the posterior variance, and a measure of uncertainty of the prediction. The conditional distribution is called the posterior probability distribution in Bayesian statistics. It is very important during the Bayesian optimization and Gaussian process to carefully choose the next point to evaluate. During the likelihood optimization, acquisition functions are used to guide the search to a promising next point. Several acquisition functions have been published in the literature and nice categorization can be found in [30].

4 Method

Bilevel problems have two levels of optimization task, such that the lower-level problem is a constraint of the upper-level problem. In general bilevel problems, the follower depends on the leader's decisions x_u. The leader has no control over the follower's decision x_l. As declared in [34], for every leader's decision, there is an optimal follower decision, such that those members are considered feasible and also satisfy the upper-level constraints. Because the follower problem is a parametric optimization problem that depends on the leader decision x_u, it is very time-consuming to adopt a nested strategy approach that sequentially solves both levels. In the continuous domain, the computational cost is very high. During the optimization process, it is important to choose wisely the next leader decision x_u, to speed up the search. So the question is: *How to choose the next*

leader decision x_u^*? Treating the leader problem as a black box, and fitting the previous leader's decisions and best follower reactions to the Gaussian process model makes it possible to estimate the next point to evaluate wisely.

Recently an algorithm has been proposed based on Bayesian optimization (BOBP) [16], using differential evolution to optimize the acquisition function. It has been shown to perform better than an evolutionary algorithm based on quadratic approximations (BLEAQ) [32]. In this work, we further improve the algorithm by using the previous iterations, including followers as priors and updating the acquisition function with conditioning on the follower's decisions to improve performance. Moreover, we investigated the contribution of follower decisions for optimizing the overall bilevel problem. The follower can observe the leader's decisions, but if we treat leader fitness as a black box then has been proposed then we also have available the best follower reactions to the leader's decisions during the optimization process. Using this data and extracting knowledge from previous iterations of the optimization process is the main idea of ConBaBo. The other question is: *Can the leader learn from the follower's decisions and use this knowledge for its own decision-making process?* Because of the structure of bilevel problems, the leader has no idea about the follower's reaction at the beginning of the optimization process. But in the following iterations, it can observe the optimum reaction of the follower without knowing the follower's objective.

Problem Statement. Following the idea above, let us assume that we have an expensive black-box function that takes leader decisions in leader decision space $x_u \in \mathcal{X}_u$ and the optimal follower's response $x_l^* \in \mathcal{X}_l$ as input. The upper-level function returns a scalar fitness score, $F(x_u, x_l^*) : \mathcal{X}_u \times \mathcal{X}_l \to \mathbb{R}$. Given a budget of N, the leader makes a decision and the follower responds to the leader accordingly. The leader can observe this information during the optimization process, learn how the follower reacts to the leader's decisions in every iteration, and choose the next leader's decision to optimize the fitness score.

Algorithm Description First, we discuss fitting the decision data to the Gaussian process model, then the motivation behind conditioning. After observing n data points, that is leader decisions, follower responses, and leader's fitness score, $\{(x_{u_i}, x_{l_i}^*, y_i)\}_{i=1}^n$ where $y_i = F(x_{u_i}, x_{l_i}^*)$ upper-level objective, we fit the Gaussian process from $\mathcal{X}_u \times \mathcal{X}_l$ to $y \in \mathbb{R}$. After we have the vector-valued dataset $\mathbf{x}_{1:n} = \{(x_{u_1}, x_{l_1}^*), ..., (x_{u_n}, x_{l_n}^*)\}$ and $\mathbf{y}_{1:n} = \{y_1, ..., y_n\}$, then we construct the Gaussian process by a prior mean μ_0 and prior covariance function Σ_0. Let us suppose to find a value of $F(\mathbf{x})$ at some vector-valued decision $\mathbf{x}$. So, the conditional distribution of $F(\mathbf{x})$ for given observations is,

$$F(\mathbf{x})|F(\mathbf{x}_{1:n}) \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x})) \tag{6}$$

$$\mu_n(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x}_{1:n})\Sigma(\mathbf{x}, \mathbf{x})^{-1}(\mathbf{y}_{1:n} - \mu_0(\mathbf{x}_{1:n})) + \mu_0(\mathbf{x}) \tag{7}$$

$$\sigma_n^2(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x}) - \Sigma_0(\mathbf{x}, \mathbf{x}_{1:n})\Sigma_0(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})^{-1}\Sigma_0(\mathbf{x}_{1:n}, \mathbf{x}) \tag{8}$$

Algorithm 1 ConBaBo: Conditional Bayesian Optimization for Bilevel Optimization

Input: Random starting upper-level decisions with size of n
$\mathbf{x}_{u_{1:n}} : \{x_{u_1}, \ldots, x_{u_n}\}$
1: Find the best *lower-level* responses
$\mathbf{x}^*_{l_{1:n}} : \{x^*_{l_1}, \ldots, x^*_{l_n}\}$ regarding $\mathbf{x}_{u_{1:n}}$ with *SLSQP* algorithm [20]
2: Evaluate *upper-level* fitness values : $\mathbf{y}_{1:n} = \{F(x_{u_1}, x^*_{l_1}), \ldots, F(x_{u_n}, x^*_{l_n})\}$
3: Initialize Gaussian process ($\mathcal{GP}$) model and fit the observed data :
$\mathcal{GP}(\mathbf{x}_{u_{1:n}}, \mathbf{x}^*_{l_{1:n}}, \mathbf{y}_{1:n})$
4: Update the $\mathcal{GP}$ model
5: **for** i = 0 : *Budget* **do**
6: Evaluate the *conditional LCB acquisition function* value by using the Equation 9
7: Suggest the next candidate $x_{u_{n+i+1}}$ by optimizing *acquisition function*
8: Evaluate the best *lower-level* response ($x^*_{l_{n+i+1}}$) regarding $x_{u_{n+i+1}}$ using *SLSQP*
9: Calculate the *upper-level* fitness:
$F(x_{u_{n+i+1}}, x^*_{l_{n+i+1}})$
10: Update the $\mathcal{GP}$ model with new observations
11: **end for**
12: **return** the best decision values (x_u, x^*_l) with *upper-level* fitness value $F(x_u, x^*_l)$

Note that there is an interaction between decision-makers in bilevel problems because of their hierarchical structure. Thus, it is important not to violate the upper-level constraints during lower-level optimization. In this paper, we follow the bilevel definition in [34] to avoid violating the upper-level constraints. As declared in the paper, the optimal lower-level decision satisfies the upper-level constraints. Every optimal follower's decision is a reaction to the leader's decision, which makes possible the use of conditional settings.

After fitting the data to the model, we choose the next leader decision $x_{u_{n+1}}$. Then we find the optimal reaction $x^*_{l_{n+1}}$ and the fitness score of the leader function $y_{n+1} = F(x_{u_{n+1}}, x^*_{l_{n+1}})$. We update the Gaussian process model with new data. Then we update the predicted optimal decisions for every iteration. After updating the model with the new follower optimal point at each iteration, we calculate the *conditional acquisition values*. For each follower optimal point x^*_l there is a global optimization problem over $x_u \in \mathcal{X}_u$. The lower confidence bound (LCB) acquisition function by [10] is used to choose the next point. We choose the exploration weight $w = 2$ for the LCB acquisition function, as it is the default setting. Acquisition function by conditioning on the follower's decision is defined as follows:

$$\alpha_{LCB}(x_u|x^*_l) = \hat{\mu}(x_u|x^*_l) - w\hat{\sigma}(x_u|x^*_l) \tag{9}$$

where w is a parameter that balances exploration and exploitation during optimization. $\hat{\mu}$ represents the conditional mean and $\hat{\sigma}$ represents the standard deviation. The conditional mean is calculated as follows:

$$\hat{\mu}(x_u|x^*_l) = \mu(x_u) + \Sigma(x_u, x^*_l)\Sigma(x^*_l, x^*_l)^{-1}(x^*_l - \mu(x^*_l)) \tag{10}$$

where $\mu(\cdot)$ is mean function and $\Sigma(\cdot)$ is covariance function. In Equation 10 we can see that correlation between the conditioned data set and the leader decision data set affects the mean function which represents the prediction. This change in posterior distribution is vital for the prediction of the Gaussian process.

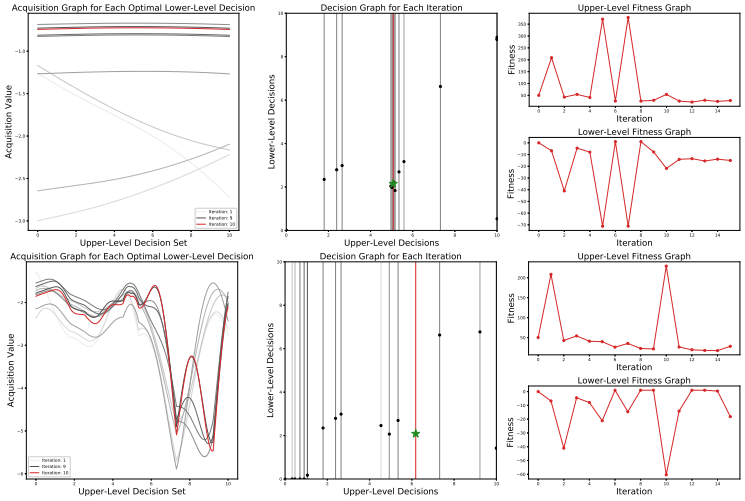

Fig. 1. Three graphs at the top show the results for the ConBaBo algorithm, and the three graphs at the bottom show the results without conditioning the lower-level decisions. Left: Acquisition function graph with and without conditioning on lower-level decisions $LCB_{x_u|x_l=x_l^*}(x_{u_{n+1}})$ during upper-level optimization. The line's colour changes from light grey to dark grey in each iteration, and the red line is the shape of the acquisition function at the end. Middle: Decision space for upper-level decisions and best lower-level reaction at each iteration. Vertical lines show the upper-level decisions and black dots show the decision pair (x_u, x_l^*). The red line shows the last decision pair after 10 iterations with 5 training points. Right: Fitness graphs show ConBaBo converges even after 10 iterations while not conditioning the lower-level decisions still searching for the optimal bilevel solution.

In this way, we used conditional setting on bilevel problems by improving the optimization with the follower's decisions.

We can see the observation for the first 10 iterations for the acquisition graph and how conditioning affects the optimization process in Figure 1. In the figure, there are three different graphs. The left graph shows the acquisition function shape over 10 iterations. The light grey colour represents the previous acquisition graph and the red one represents the 10^{th} iteration. During the Gaussian process, the acquisition function is the key point to select the next iteration and it is being done by optimizing the acquisition function. This is the reason for the importance of acquisition function during optimization. As we can observe from the 1, the conditional acquisition function shape (top-left in Figure 1) is becoming convex even after 8 iterations while the standard acquisition function shape (bottom-left in Figure 1) is maintaining non-convexity. Having convexity in acquisition optimization is vital in Bayesian optimization for improving the optimization performance in terms of computational cost. It shows the importance of gaining the lower-level optimal decision information for the upper-level decision makers. The graphs in the middle show the upper- and lower-level decision space. The

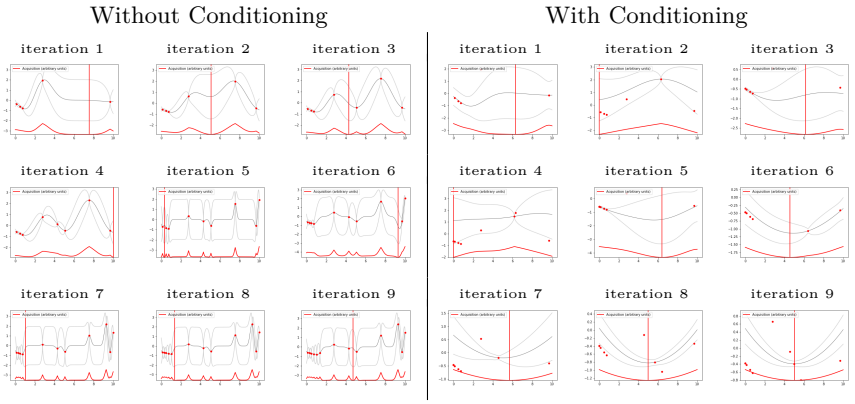

Fig. 2. First 9 iterations of the acquisition function graph with and without conditioning the follower's decision.

vertical lines show the upper-level decisions and the stars show the lower-level responses to the upper-level decision-makers. We shared these graphs because of observing the behaviour of searching during the optimization process. We can see clearly that conditioning the lower-level decisions affects the choice of decisions at both levels and trying to converge at global optima even in 8 iterations while the standard acquisition function tries to minimize the uncertainty at different points in the search space. Practical problems in the bilevel structure are extremely hard and need time and consume computational power in most cases. So, we believe the improvement of ConBaBo algorithm and the technique proposed provides a good technique for applying it to several practical problems.

We shared more acquisition graphs in a detailed way in Figure 2. It shows the acquisition graph for each iteration for 9 iteration. As we can see from Figure 2, the standard acquisition function behaves more explorative and tries different points at each step while trying to obtain a global optimal decision. As we can see from the graphs, the conditional acquisition function reached the near global optima even in 6 iterations. This fact makes us save lots of function evaluation during the bilevel optimization process. The acquisition function shape with conditioning becomes convex around global optima over the iterations quicker compared with the non-conditional setting. The ConBaBo algorithm details are shown in Algorithm 1.

5 Numerical Experiments and Results

We evaluate ConBaBo on two test benchmarks including linear, non-linear, constrained, and unconstrained minimization test problems containing multiple dimensional decision variables.

Test Problems. The first benchmark is standard test problems, called TP problems, and contains 10 bilevel problems. Each problem has different interaction

between upper-level and lower-level decision-makers. The benchmark includes 1 three-dimensional, 6 four-dimensional, 1 five-dimensional, and 2 ten-dimensional bilevel problems. Each problem has different complexity in terms of interaction between decision-makers. The benchmark includes mostly constrained problems and is created for testing the efficiency of proposed methods by the researchers. Mostly the efficiency is determined by function evaluations and obtained fitnesses. More details is can be found in [32]. The second is called SMD problems and it has 6 unconstrained problems. These are unconstrained problems with controllable complexities. Each problem has a different difficulty level in terms of convergence, the complexity of interaction, and lower-level multimodality. They are also scalable in terms of dimensionality. We set the dimensions of all problems as 4 for all experiments. The termination criteria is selected as the same as TP problems. More details about this benchmark can be found in [31].

Parameters. All experiments were conducted on a single core of 1.4 GHz Quad Core i5, 8Gb 2133 Mhz LPDDR3 RAM. The algorithm is executed 31 times for each test function and the results shown are medians. The termination criterion is selected as $\epsilon = 10^{-6}$ representing the difference between the obtained result and the best result of compared algorithms. Bayesian optimization and Gaussian processes are implemented in Python via the GPyOpt library [2] and *optimize_restarts* selected as 10 with the parameters of *verbose=False*, and *exact_feval=True*. The method initialized 5 Sobol points to construct the initial GP model. The LCB acquisition function has been selected for Gaussian Process and the Radial Basis Function (RBF) kernel is used. For optimizing the acquisition function, the L-BFGS method [22] has been used. The exploration-exploitation parameter for LCB, w is set to 2. After we set up the upper-level optimization method by conditional acquisition function, we used Sequential Least Squares Programming (SLSQP) [20] for solving the lower-level problem in Python programming language. For evaluating the ConBaBo performance and making the comparison with the other selected algorithms, we set the termination criteria considering the best-obtained result of the other algorithms. The number of function evaluations is shared considering the Bayesian optimization iteration and compared with the selected algorithms median of 31 runs. We run the algorithm with different random seeds for making the comparison fair.

5.1 Results and the Discussion

We compared the experimental results with some state-of-art algorithms in the literature. To show the effectiveness of ConBaBo we compare it with the results in [32, 16]. Sinha et al. present an improved evolutionary algorithm in [33] based on quadratic approximations of the lower-level problem, called BLEAQ. [16] propose an algorithm called BOBP with a Bayesian approach. Table 1 shows the median fitness values at both upper and lower-level for ConBaBo and other algorithms. We also show the median function evaluations for upper-level and lower-level optimization in Table 2. We did not share the standard deviation over multiple runs because we could not compare it with the other algorithms

as there is no information presented about the confidence interval. The primary purpose of bilevel optimization is to solve the leader problem, so in comparing the algorithms we focus on leader fitness and function evaluations at each level.

Table 1. Median Upper-level and lower-level fitness scores of ConBaBo, BOBP, BLEAQ algorithms for TP1-TP10.

	Median Upper-level (UL) and Lower-level (LL) Fitness Scores					
	ConBaBo		BOBP		BLEAQ	
	UL	LL	UL	LL	UL	LL
TP1	226.0223	98.0075	253.6155	70.3817	224.9989	99.9994
TP2	0.0000	200.0000	0.0007	183.871	2.4352	93.5484
TP3	-18.0155	-1.0155	-18.5579	-0.9493	-18.6787	-1.0156
TP4	-30.9680	3.1472	-27.6225	3.3012	-29.2	3.2
TP5	-4.3105	-1.5547	-3.8516	-2.2314	-3.4861	-2.569
TP6	-1.2223	7.6751	-1.2097	7.6168	-1.2099	7.6173
TP7	-1.9001	1.9001	-1.6747	1.6747	-1.9538	1.9538
TP8	0.0000	200.0000	0.0008	180.6452	1.1463	132.5594
TP9	0.00003	1.0000	0.0012	1.0000	1.2642	1.0000
TP10	0.000285	1.0000	0.0049	1.000	0.0001	1.0000

Table 2. Median function upper-level and lower-level function evaluations for ConBaBo and other known algorithms for TP1-TP10

	Median Upper-level (UL) and Lower-level (LL) Function Evaluations					
	ConBaBo		BOBP		BLEAQ	
	UL	LL	UL	LL	UL	LL
TP1	18.1333	190.6646	211.1333	1,558.8667	588.6129	1543.6129
TP2	17.6772	148.2746	35.2581	383.0645	366.8387	1,396.1935
TP3	10.1836	111.1374	89.6774	1,128.7097	290.6452	973.0000
TP4	14.6129	1,063.901	16.9677	334.6774	560.6452	2,937.3871
TP5	23.5164	222.2308	57.2258	319.7742	403.6452	1,605.9355
TP6	8.3333	323.1333	12.1935	182.3871	555.3226	1,689.5484
TP7	13.7092	274.9229	72.9615	320.2308	494.6129	26,682.4194
TP8	14.6066	114.9942	37.7097	413.7742	372.3226	1,418.1935
TP9	8.9215	1,020.8618	16.6875	396.3125	1,512.5161	141,303.7097
TP10	15.1935	3,689.6452	21.3226	974.0000	1,847.1000	245,157.9000

As can be seen in Table 1, ConBaBo achieved better upper-level results for 7 problems TP2, TP4, TP5, TP6, TP7, TP8, and TP9 in terms of fitness scores obtained. Moreover, the number of function calls by the leader decreases significantly as we encounter near-optimal solutions at both levels as we can see in Table 2. The test benchmark has constraint problems with various dimensional decision variables. The ConBaBo results are promising when we consider the complexity of the problems in the benchmark. We can see that ConBaBo performs well on this standard test set.

Table 3. ConBaBo, CGA-BS [1], BLMA [14], NBLE [14], BIDE [14] and BLEAQ [33] upper-level fitness for SMD1-SMD6.

	Median Upper-level Fitness Scores					
	ConBaBo	CGA-BS	BLMA	NBLE	BIDE	BLEAQ
SMD1	1.8×10^{-6}	0	1.0×10^{-6}	5.03×10^{-6}	3.41×10^{-6}	1.0×10^{-6}
SMD2	9.09×10^{-6}	2.22×10^{-6}	1.0×10^{-6}	3.17×10^{-6}	1.29×10^{-6}	5.44×10^{-6}
SMD3	1.6×10^{-6}	0	1.0×10^{-6}	1.37×10^{-6}	4.1×10^{-6}	7.55×10^{-6}
SMD4	6.2×10^{-6}	3.41×10^{-11}	1.0×10^{-6}	9.29×10^{-6}	2.3×10^{-6}	1.0×10^{-6}
SMD5	1.49×10^{-6}	1.13×10^{-9}	1.0×10^{-6}	1.0×10^{-6}	1.58×10^{-6}	1.0×10^{-6}
SMD6	2.36×10^{-6}	9.34×10^{-11}	1.0×10^{-6}	1.0×10^{-6}	3.47×10^{-6}	1.0×10^{-6}

Table 4. Median function upper-level function evaluations for ConBaBo and other known algorithms for SMD1-SMD6

	Median Upper-level Function Evaluations					
	ConBaBo	CGA-BS	BLMA	NBLE	BIDE	BLEAQ
SMD1	1.1×10^{1}	1.01×10^{4}	1.19×10^{3}	1.52×10^{3}	6.0×10^{3}	1.19×10^{3}
SMD2	1.4×10^{1}	5.0×10^{4}	1.20×10^{3}	1.56×10^{3}	6.0×10^{3}	1.20×10^{3}
SMD3	2.3×10^{1}	1.0×10^{4}	1.29×10^{3}	1.56×10^{3}	6.0×10^{3}	1.29×10^{3}
SMD4	1.01×10^{1}	1.25×10^{5}	1.31×10^{3}	1.53×10^{3}	6.0×10^{3}	1.31×10^{3}
SMD5	2.37×10^{1}	1.0×10^{5}	2.06×10^{3}	3.40×10^{3}	6.0×10^{3}	2.06×10^{3}
SMD6	2.41×10^{1}	1.37×10^{5}	4.08×10^{3}	4.06×10^{3}	6.0×10^{3}	4.08×10^{3}

Table 5. Median function lower-level function evaluations for ConBaBo and other known algorithms for SMD1-SMD6

	Median Lower-level Function Evaluations					
	ConBaBo	CGA-BS	BLMA	NBLE	BIDE	BLEAQ
SMD1	3.8×10^{2}	1.5×10^{4}	2.37×10^{5}	9.520×10^{5}	1.8×10^{7}	2.37×10^{5}
SMD2	3.81×10^{2}	1.5×10^{5}	4.08×10^{5}	9.63×10^{5}	1.8×10^{7}	4.08×10^{5}
SMD3	7.7×10^{2}	2.0×10^{4}	3.02×10^{5}	1.04×10^{6}	1.8×10^{7}	3.02×10^{5}
SMD4	2.58×10^{2}	2.58×10^{5}	3.07×10^{5}	8.33×10^{5}	1.8×10^{7}	3.07×10^{5}
SMD5	9.05×10^{2}	2.58×10^{5}	8.42×10^{5}	2.22×10^{6}	1.8×10^{7}	8.42×10^{5}
SMD6	7.69×10^{2}	3.39×10^{5}	1.98×10^{4}	1.11×10^{5}	1.8×10^{7}	1.98×10^{4}

SMD problems are scalable in terms of the number of decision variables. We compared our results with CGA-BS, BLMA [14], NBLE [14], BIDE [14], and BLEAQ algorithms. We could not compare it with the BOBP algorithm because lack if information in the reference paper. Table 3 shows the upper-level fitness comparison with the other algorithms. We shared the upper-level and lower-level median function evaluations at Table 4 and Table 5 to show the effectiveness of the ConBabo algorithm. Also for SMD problems, we did not share the standard deviation over multiple runs because of the same reason with TP problems. We can observe that from Tables 4 and 5 that the upper- and lower-level function evaluations decrease significantly. These results show that ConBaBo can manage the difficulties resulting from the proposed SMD benchmark by [31]. Moreover, it converges faster to optimal solutions compared to other algorithms in the literature. The problems in the benchmark are scalable and have different dif-

ficulties in terms of the interaction between decision-makers. Considering this, ConBaBo performs well on this unconstrained bilevel benchmark problem.

6 Conclusion

Bilevel optimization problems are a specific kind of problem that feature two decision makers, with a mathematical representation called "Stackelberg Games" in Game Theory. There are two optimization problems in these problems called the "upper-level" and "lower-level". Decisions in each level are affected by those made in the other, so the bilevel optimization scheme is time-consuming in terms of performance. We propose ConBaBo, a conditional Bayesian optimization algorithm for bilevel optimization problems. The conditional Bayesian approach allows us to extract knowledge from previous upper- and lower-level decisions, leading to smarter choices and therefore fewer function evaluations. ConBaBo increases algorithm performance quality and dramatically accelerates the search for an optimal solution. We evaluate our method on two common benchmark sets from the bilevel literature, comparing results with those for top-performing algorithms in the literature. The ConBaBo algorithm can be considered a powerful global technique for solving bilevel problems, which can handle the difficulties of non-linearity and conflict between decision-makers. Moreover, it can deal with constrained and unconstrained problems with multi-dimensional decision variables. In future work, the practical applications of bilevel problems and multi-objective bilevel problem adaptations will be researched.

Acknowledgements This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 16/RC/3918 which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

1. Abo-Elnaga, Y., Nasr, S.: Modified evolutionary algorithm and chaotic search for bilevel programming problems. Symmetry **12** (5 2020). https://doi.org/10.3390/SYM12050767
2. authors, T.G.: GPyOpt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt (2016)
3. Bard, J.F.: Coordination of a multidivisional organization through two levels of management. Omega **11**(5), 457–468 (1983). https://doi.org/10.1016/0305-0483(83)90038-5
4. Bard, J.F., Falk, J.E.: An explicit solution to the multi-level programming problem. Computers Operations Research **9**(1), 77–100 (jan 1982). https://doi.org/10.1016/0305-0548(82)90007-7

5. Bard, J.F., Moore, J.T.: A Branch and Bound Algorithm for the Bilevel Programming Problem. SIAM Journal on Scientific and Statistical Computing **11**(2), 281–292 (mar 1990). https://doi.org/10.1137/0911017
6. Bialas, W., Karwan, M.: On two-level optimization. IEEE Transactions on Automatic Control **27**(1), 211–214 (1982). https://doi.org/10.1109/TAC.1982.1102880
7. Bracken, J., McGill, J.T.: Defense applications of mathematical programs with optimization problems in the constraints. Oper. Res. **22**(5), 10861096 (oct 1974). https://doi.org/10.1287/opre.22.5.1086
8. Brown, G., Carlyle, M., Diehl, D., Kline, J., Wood, R.: A two-sided optimization for theater ballistic missile defense. Operations Research **53**, 745–763 (10 2005). https://doi.org/10.1287/opre.1050.0231
9. Constantin, I., Florian, M.: Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. International Transactions in Operational Research **2**(2), 149–164 (1995). https://doi.org/10.1016/0969-6016(94)00023-M
10. Cox, D.D., John, S.: Sdo: A statistical method for global optimization. In: in Multidisciplinary Design Optimization: State-of-the-Art. pp. 315–329 (1997)
11. Dogan, V., Prestwich, S.: Bayesian optimization withămulti-objective acquisition function forăbilevel problems. In: Longo, L., O'Reilly, R. (eds.) Artificial Intelligence and Cognitive Science. pp. 409–422. Springer Nature Switzerland, Cham (2023)
12. Frazier, P.: A tutorial on bayesian optimization. ArXiv **abs/1807.02811** (2018)
13. Hong, M., Wai, H.T., Wang, Z., Yang, Z.: A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. ArXiv **abs/2007.05170** (2020)
14. Islam, M.M., Singh, H.K., Ray, T., Sinha, A.: An enhanced memetic algorithm for single-objective bilevel optimization problems. Evolutionary Computation **25**, 607–642 (12 2017). https://doi.org/10.1162/EVCO_a_00198
15. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**, 455–492 (12 1998). https://doi.org/10.1023/A:1008306431147
16. Kieffer, E., Danoy, G., Bouvry, P., Nagih, A.: Bayesian optimization approach of general bi-level problems. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 16141621. GECCO '17, Association for Computing Machinery (2017). https://doi.org/10.1145/3067695.3082537
17. Kirjner-Neto, C., Polak, E., Kiureghian, A.D.: An outer approximation approach to reliability-based optimal design of structures. J. Optim. Theory Appl. **98**(1), 116 (jul 1998)
18. Koh, A.: Solving transportation bi-level programs with differential evolution. In: 2007 IEEE Congress on Evolutionary Computation. pp. 2243–2250 (2007). https://doi.org/10.1109/CEC.2007.4424750
19. Koppe, M., Queyranne, M., Ryan, C.T.: Parametric Integer Programming Algorithm for Bilevel Mixed Integer Programs. Journal of Optimization Theory and Applications 2010 146:1 **146**(1), 137–150 (feb 2010). https://doi.org/10.1007/S10957-010-9668-3
20. Kraft, D.: A software package for sequential quadratic programming. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR (1988)
21. Legillon, F., Liefooghe, A., Talbi, E.G.: Cobra: A cooperative coevolutionary algorithm for bi-level optimization. In: 2012 IEEE Congress on Evolutionary Computation. pp. 1–8 (2012). https://doi.org/10.1109/CEC.2012.6256620

22. Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical Programming 1989 45:1 **45**, 503–528 (8 1989). https://doi.org/10.1007/BF01589116
23. Ma, L., Wang, G.: A solving algorithm for nonlinear bilevel programing problems based on human evolutionary model. Algorithms **13**(10) (2020)
24. Migdalas, A.: Bilevel programming in traffic planning: Models, methods and challenge. Journal of Global Optimization **7**, 381–405 (01 1995). https://doi.org/10.1007/BF01099649
25. Pearce, M., Klaise, J., Groves, M.J.: Practical bayesian optimization of objectives with conditioning variables. arXiv: Machine Learning (2020)
26. Sabach, S., Shtern, S.: A first order method for solving convex bi-level optimization problems (2017). https://doi.org/10.48550/ARXIV.1702.03999
27. Sahin, K., Ciric, A.R.: A dual temperature simulated annealing approach for solving bilevel programming problems. Computers Chemical Engineering **23**, 11–25 (1998)
28. Savard, G., Gauvin, J.: The steepest descent direction for the nonlinear bilevel programming problem. Operations Research Letters **15**(5), 265–272 (jun 1994). https://doi.org/10.1016/0167-6377(94)90086-8
29. Shaban, A., Cheng, C.A., Hatch, N., Boots, B.: Truncated back-propagation for bilevel optimization. CoRR **abs/1810.10667** (2018)
30. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., Freitas, N.D.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**, 148–175 (2016). https://doi.org/10.1109/JPROC.2015.2494218
31. Sinha, A., Malo, P., Deb, K.: Unconstrained scalable test problems for single-objective bilevel optimization. In: 2012 IEEE Congress on Evolutionary Computation. pp. 1–8 (2012). https://doi.org/10.1109/CEC.2012.6256557
32. Sinha, A., Malo, P., Deb, K.: Efficient evolutionary algorithm for single-objective bilevel optimization (2013). https://doi.org/10.48550/ARXIV.1303.3901
33. Sinha, A., Malo, P., Deb, K.: An improved bilevel evolutionary algorithm based on quadratic approximations. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 1870–1877 (2014). https://doi.org/10.1109/CEC.2014.6900391
34. Sinha, A., Malo, P., Deb, K.: A review on bilevel optimization: From classical to evolutionary approaches and applications. IEEE Transactions on Evolutionary Computation **22**(2), 276–295 (2018). https://doi.org/10.1109/TEVC.2017.2712906
35. Sinha, A., Malo, P., Frantsev, A., Deb, K.: Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In: 2013 IEEE Congress on Evolutionary Computation. pp. 478–485 (2013). https://doi.org/10.1109/CEC.2013.6557607
36. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W.: Information-theoretic regret bounds for gaussian process optimization in the bandit setting. IEEE Transactions on Information Theory **58**(5), 3250–3265 (may 2012). https://doi.org/10.1109/tit.2011.2182033
37. Stackelberg, H.v.: The theory of the market economy. William Hodge, London (1952)
38. Talbi, E.G.: A taxonomy of metaheuristics for bi-level optimization. Studies in Computational Intelligence **482** (05 2013). https://doi.org/10.1007/978-3-642-37838-6-1
39. Vicente, L., Savard, G., Júdice, J.: Descent approaches for quadratic bilevel programming. J. Optim. Theory Appl. **81**(2), 379399 (may 1994)