Bayesian Optimization Approach of General Bi-level Problems

Emmanuel Kieffer,
Grégoire Danoy,
Pascal Bouvry
University of Luxembourg
firstname.name@uni.lu

Anass Nagih
University of Lorraine
firstname.name@univ-lorraine.fr

ABSTRACT

Real-life problems including transportation, planning and management often involve several decision makers whose actions depend on the interaction between each other. When involving two decision makers, such problems are classified as bi-level optimization problems. In terms of mathematical programming, a bi-level program can be described as two nested problems where the second decision problem is part of the first problem's constraints. Bi-level problems are NP-hard even if the two levels are linear. Since each solution implies the resolution of the second level to optimality, efficient algorithms at the first level are mandatory. In this work we propose BOBP, a Bayesian Optimization algorithm to solve Bi-level Problems, in order to limit the number of evaluations at the first level by extracting knowledge from the solutions which have been solved at the second level. Bayesian optimization for hyper parameter tuning has been intensively used in supervised learning (e.g., neural networks). Indeed, hyper parameter tuning problems can be considered as bi-level optimization problems where two levels of optimization are involved as well. The advantage of the bayesian approach to tackle multi-level problems over the BLEAQ algorithm, which is a reference in evolutionary bi-level optimization, is empirically demonstrated on a set of bi-level benchmarks.

CCS CONCEPTS

•Mathematics of computing → Bayesian nonparametric models; Bayesian computation; Evolutionary algorithms; •Theory of computation → Gaussian processes; Bayesian analysis;

KEYWORDS

Bi-level programming, Bayesian optimization

ACM Reference format:
Emmanuel Kieffer,
Grégoire Danoy,
Pascal Bouvry and Anass Nagih. 2017. Bayesian Optimization Approach of General Bi-level Problems. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082537

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '17 Companion, Berlin, Germany
© 2017 ACM. 978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3082537

1 INTRODUCTION

According to Roger B. Myerson, Game Theory can be described as "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers". The outcome of a competitive game is generally a equilibrium (e.g., Cournot, Nash, Stackelberg) where none of the decision makers would benefit by modifying their decision.

In this work, we focus our attention to some particular non-cooperative games called Stackelberg Games [32] which are sequential non-zero sum games involving 2 players. The first player called "leader" is aware of the problem of the second player called "follower". Taking the point of view of the leader, its problem depends on the reaction of the follower. In order to take the right decision, it has to compute the optimal strategy of the follower called " the follower rational decision".

These games can be mathematically modeled as two nested optimization problems, also referred to as bi-level problems, where the inner level is part of the outer problem constraints. The outer problem is generally called the "upper level" (UL) while the inner problem is called the "lower-level" (LL). The order between levels is crucial which means that swapping the two levels leads to different optimal solutions if they exist. This nested structure implies that a feasible solution at the UL should be optimal for the LL problem. This is the reason why bi-level optimization problems are very difficult to solve. Even two nested linear continuous levels lead to a NP-hard single-level problem.

Different approaches have been proposed in the literature to tackle bi-level problems. However, most of them have been designed to tackle only specific versions of bi-level problems. For instance, a large set of exact methods was introduced to solve small linear bi-level optimization problems. The last decade has seen a renewed interest for bi-level optimization, especially in the field of evolutionary computing. This is probably due to the new optimization needs to solve problems with multiple decision makers.

A very interesting example of bi-level problem is the toll setting problem introduced by Brotcorne in [11]. An authority operates a network of roads which may have tolls. This authority would like to maximize its revenues by finding the optimal toll prices. However, tolls are paid by network users who may decide to take secondary roads in order to minimize their cost. The authority has to take into account the network user problem which consists in minimizing his travel costs. Indeed, the authority only controls a part of the decision vector, i.e., the prices, while the network users control the path they take in the network. Generally to solve such nested optimization problems, the LL problem is replaced by its Karuhn-Kuhn-Tucker conditions to obtain a single-level problem. This kind of transformation cannot be employed in all cases. An

alternative would be to sequentially solve both levels but this is not suitable for large instances. In [31], Sinha *et al.* have approximated the LL optimal solutions using the UL decision variables to reduce the number of LL optimizations. Although this approach is very innovative, a genetic algorithm is employed at the UL which necessitates a lot of fitness evaluations. In addition, the authors are required to perform a multi-output approximation. To tackle this issue, they consider m independent single-output regressions by building as many approximate functions as LL variables. In this work, we propose BOBP to tackle these issues by considering Bayesian optimization. Instead of generating several approximate functions for each LL variable, we make use of Gaussian processes not to approximate LL optimal solutions but to estimate directly their UL fitness value. Bayesian optimization also embeds an acquisition function which permits to determine the most promising search areas during optimization. Therefore, this approach attempts to minimize the number of fitness evaluations. Generally in the literature, acquisition functions are optimized using local search approaches, i.e. L-BFGS-B [26] which tend to be trapped in local optimum. This is the reason why we propose to use an evolutionary algorithm and more precisely Differential evolution" to detect better acquisition points and thus better search areas. It is well-suited for hyper-parameter optimization which requires to discover a set of optimal parameters for a complex or black-box model.
The remainder of this article is organized as follows. Section 2 introduces the existing works on bi-level optimization as well as Bayesian optimization. Section 3 formalizes the problem and then states explicitly how Bayesian optimization can be adapted to solve bi-level problems. Section 4 focuses on the experimentation protocol and provides comparison results with a state-of-the-art algorithm, namely the Bi-level Evolutionary Algorithm based on Quadratic approximations (BLEAQ). Finally, sections 5 concludes this work and propose future investigations.

2 RELATED WORK

Multi-level and bi-level optimization stems from the need to tackle sequential optimization problems in which each level is controlled by a different decision maker. Connected to Game Theory, bi-level problems can be seen as the mathematical programming counterpart of Stackelberg games[32] introduced by H. von Stackelberg in 1952. The foundations and formulation of bi-level optimization problems have been first proposed by J. Bracken and J. McGill in[9]. They also provide some defense application examples in [10]. At this time, the definition "bi-level" was still not in use and the common designation of bi-level problems was "mathematical programs with optimization problems in the constraints".

Many practical problems involving a bi-level structure have been studied in the literature. Transportation [23], planning [35] and management [2] are the main topics modeled as bi-level problems in practice. Indeed, they are naturally involving several decision makers. Bi-level problems are intrinsically hard even for convex levels. The simplest bi-level linear programs have been proven to be strongly NP-hard [3]. Exact approaches are thus not efficient but have been mainly investigated. The first one is based on a reformulation of the bi-level model into a single-level model by replacing the follower problem by its Karush Kuhn Tucker conditions [4]. The second uses vertex enumeration with the aid of a modified version of the well-know simplex method [7]. The third one extracts gradient information of the LL problem and generates directional derivatives using by the UL objective function[30]. The last category introduces penalty functions which compute stationary points and local optimum [6].

Concerning integer and mixed integer bi-level problems, three main categories can be identified in the literature. In the first case, reformulation methods have been employed like decomposition methods [14]. The second category is based on Branch and Bound [5] while the last one includes parametric programming approaches [24].

Most of the studies published so far focus on special cases. Concerning multi-objective exact approaches where each decision maker wants to optimize a set of objectives, the LL problem is often replaced in order not to be treated as a bi-level problem [8]. The complexity induced by multiple levels and/or multiple objectives makes exact approaches non-efficient even though very important results have been discovered. For that reason, researchers turned towards metaheuristics.

Metaheuristics have been successfully and widely used in single-level optimization cases to tackle NP-hard problems. Indeed for those problems and under the assumption that $P \neq NP$ [15], it does not exist algorithms with polynomial complexity solving each instance to optimality. The optimality condition is relaxed to keep fast resolution procedures. Since convex bi-level problems are NP-hard contrary to their single-level convex problems, the scope of metaheuristics has been extended. According to the taxonomy provided in [33], four existing categories have been considered in the literature:

(1) Nested sequential (NSQ)
    - Repairing approach (REP)
    - Constructive approach (CST)
(2) Single-level transformation (STA)
(3) Co-evolutionary (COE)
(4) Multi-objective (MOA)

Two kinds of NSQ categories have been reported. The first one (REP) considers the LL problem as a constraint and applies a repairing procedure to be sure that it is satisfied [17]. The second one is more trivial and consists in solving sequentially the two levels [20]. The STA category aims at transforming a bi-level problem into a general single-level problem [29] while the COE category focuses on decentralized algorithms trying to decouple both levels [19]. The MOA category relies on the transformation of bi-level problems into equivalent multi-objective problems. These transformations are not trivial since a Pareto optimal solution is not necessarily bi-level optimal. In [13], the authors introduced a new methodology ensuring the equivalence between the original problem and its multi-objective version. Recently, Deb and Sinha proposed the Bi-level Evolutionary Algorithm based on Quadratic approximations (BLEAQ), which could be part of a new fifth class. Indeed, this algorithm attempts to approximate the inducible region which is basically the resulting feasible region of a bi-level problem. This strategy reduces the number of LL optimizations. Nevertheless, it has some drawbacks, like the use of a population-based metaheuristic at the UL which tends to increase the number of evaluations.

Another weakness is the fact that BLEAQ has to build as many approximate functions as LL decision variables. In this paper, we present an alternative approach which does not try to approximate the LL decision variables but directly the objective value of solutions located in the inducible region.

Figure 1 summarizes the existing categories [33] and the new one, i.e., APP, which attempts to approximate either solutions (IRS) or the objective values (IRF) in the inducible region using surrogate models. IRF stands for approximation of Inducible Region Fitness and IRS for

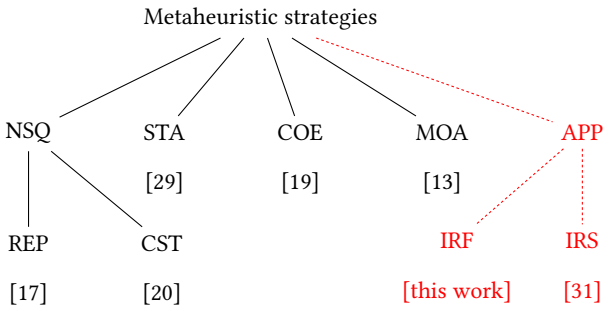

Figure 1: Extension of the Bi-level metaheuristics taxonomy

approximation of Inducible Region Solutions. In this work, bi-level optimization problems are solved with a Bayesian Optimization algorithm which uses Gaussian processes to approximate complex objective functions. Since discovering IR could be a hard task, it is therefore complex to evaluate the UL objective function on it. Bayesian optimization has been mostly employed in machine learning [12, 27] as an alternative to the Grid Search and Random Search algorithms. They are well-suited for multi-level optimization problems as they take into account the evaluation cost. The next section will introduce Bayesian optimization as well as the way to adapt it for general bi-level optimization.

3 DESCRIPTION OF THE BOBP APPROACH

This paper is a two-fold contribution. It first explains how Bayesian optimization can be applied on bi-level optimization problems. Secondly, we propose to employ global optimization algorithms to optimize acquisition functions generally solved with gradient-based algorithms. Differential evolution has been selected to perform this task. Hereafter we introduce formally bi-level problems and provide some definitions. Then, we describe the general mechanism behind Bayesian optimization. Finally, we show how to adapt classical Bayesian optimization to bi-level problems.

3.1 Introduction to bi-level problems

As described in section 1, bi-level problems have a nested structure. The LL problem is part of UL constraints.

$$\begin{aligned} \min \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & \min \quad f(x,y) \\ & \text{s.t.} \quad g(x,y) \leq 0 \\ & x,y \geq 0 \end{aligned}$$

where F, f : $\mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}$, $G : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^p$ and $\mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^q$.

Let us introduce some definitions and properties:

- The constraint region of the general bi-level problem: $S = \{(x,y) : x \in X, y \in Y, G(x,y) \leq 0, g(x,y) \leq 0\}$
- The feasible set for the LL problem parametrized by $x \in X$: $S_L(x) = \{y \in Y : g(x,y) \leq 0\}$.
- The projection of S onto the UL decision space: $Proj_{S(X)} = \{x \in X : \exists y \in Y, G(x,y) \leq 0, g(x,y) \leq 0\}$
- The LL rational decision set for $x \in S(X)$: $P(x) = \{\hat{y} \in Y : \hat{y} \in \arg\min[f(x,y) : y \in S_L(x)]\}$
- The Inducible Region: $IR = \{(x,\hat{y}) \in S, \hat{y} \in P(x)\}$

The LL problem is a parametric optimization problem which depends on the UL decision variable x. With no control on y, the UL decision maker has to determine $P(x)$ in order to compute its objective value. For every x, an optimal solution $\hat{y} \in P(x)$ for the LL problem has to be computed in order to observe the rational reaction. As aforementioned, the UL feasibility is determined by LL optimality. This is the reason why IR is the resulting feasible region of the general bi-level problem. Figure 3 is an example provided by Mersha and Dempe in [22] which depicts a very interesting example of discontinuous IR. Although this example is a linear bi-level problem, it reflects the inherent difficulties to cope with multiple levels.

First let us notice that a LL solution y may not belong to S. Indeed the LL problem is indifferent to UL constraints which explains why the order between the two decision makers is so important. The feasible set for the LL problem may be outside the UL feasible region defined by $G(x,y) \leq 0$. This situation can be observed in Figure 3. The optimal reaction $P(x = 5) = \{\hat{y} \in Y : \hat{y} \in \arg\min[f(5,y) : y \in S_L(5)]\} = \{12\}$ is clearly non-feasible for the UL problem. This could only be determined after LL optimization. Hence, there is a risk for the UL problem to be non-feasible if the LL decision maker provides a rational reaction outside UL feasible area. In the context of a Chess game, we could say that the LL player checkmate the UL player.

It is very time consuming to adopt a nested strategy approach which sequentially solves both levels. In [31], the authors focused on localization of IR. Under the assumption that $P(x) = \{\hat{y}\}$ is a singleton for each x, they tried to establish a relation between x and $\hat{y}$ to avoid repetitive LL optimizations. They assume that under Lipschian continuity 2 close solutions x and x' will lead to close rational solutions $\hat{y}$ and $\hat{y'}$. It means that approximate functions could be considered to estimate $\hat{y}$ from x. Nevertheless, to generate these approximate functions, they have to deal with multi-output regression to determine m functions such that the approximation error is minimal. In addition, these approximate functions are restricted to be quadratic functions. In this work, we do not attempt to approximate $\hat{y}$ but $F(x,\hat{y})$ directly. Under the

same assumption that $P(x)$ is a singleton, the optimal solution $\hat{y}$ is unique and we can conclude that it may exist a surrogate function $F' : \mathbf{R}^n \mapsto \mathbf{R}$ with $F'(x) = F(x, \hat{y})$. Notice that if $P(x)$ is not a singleton, two alternatives exist:

- The optimist case: we choose $\hat{y} = \arg\min\{F(x, y) : y \in P(x)\}$.
- The pessimist case: we choose $\hat{y} = \arg\max\{F(x, y) : y \in P(x)\}$.

According to this new representation, we only focus on x. By doing so, we obtain some kind of black-box function and we do not need to deal explicitly with the LL decision variables y at the UL. Therefore, it allows us to decouple x and y. In this case, a very promising approach for black-box optimization is Bayesian optimization. A complex model is replaced by a function modelling its effects. Based on Gaussian processes, no assumption are needed to approximate the function contrary to the work proposed in [31] which only uses quadratic approximations. The next section introduces formally the notion of Bayesian optimization.

3.2 Bayesian optimization

Bayesian optimization is a model-based approach which aims at solving very costly problems. It can be assimilated as a black-box optimization algorithm where the formal expression of the objective function may be unknown or very difficult to obtain. To overcome this issue and reduce computation cost, Bayesian optimization generates a surrogate model of the unknown function using Gaussian processes[25]. It samples promising zones in the feasible region by computing a distribution of the objective function. This distribution give us a prior knowledge on location of the optimal solution. Bayesian optimization is thus characterized by two important mechanisms:

- A probability measure on F describing our prior beliefs on F ;
- The acquisition function which allows to gain information on the location of the minimum value of the objective function.

Considering a cost function $F(x)$, Gaussian processes determine the probability distribution of the function $F(x)$ at each x. These distribution are Gaussian and thus characterized by a mean value $\mu(x)$ and a variance $\sigma^2(x)$. Hence a probability distribution over functions can be defined as follows:

$$P(F(\mathbf{x})|\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})) \tag{1}$$

Obviously, the parameters $\mu(x)$ and $\sigma^2(x)$ have to be estimated. This is done by fitting the Gaussian processes to the data. Using several observations, we obtain a sample of a multivariate Gaussian distribution [28], determined by a mean vector and a covariance matrix. In fact, Gaussian processes generalize the notion of multivariate Gaussian distribution. For complex non-linear functions, the covariance matrix can be defined using a kernel function $k(x, x')$. This covariance matrix defines the correlation between data. Two distant data x and x' should not influence each other while two close data are strongly correlated.

$$F(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \tag{2}$$

where $\mathcal{GP}$ stands for Gaussian process. The squared exponential kernel is often used and defined as follows:

$$k(x, x') = l \cdot \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \text{with parameters } l \text{ and } \sigma^2 \tag{3}$$

To fit the Gaussian process to the data, the likelihood is optimized from the evaluations of each observations. Each time a new point is added to the model, a re-optimization is performed to maximize the likelihood. The question is now :'How should we determine a new point ?'. This is achieved by optimizing an acquisition function which statistically models our confidence to find the location of the optimal value. Several acquisition functions exist such as the Maximum Probability of Improvement (MPI), the Expected Improvement (EI), or the Lower-Confidence Bounds (LCB) are computed as follows:

- $\text{acq}_{MPI}(x) = \Phi(\gamma(x))$.
- $\text{acq}_{EI}(x) = \sigma x(\gamma(x)\,\Phi(\gamma(x)) + \phi(\gamma(x)))$.
- $\text{acq}_{LCB}(x) = \mu(x) - k\sigma(x)$.

where $\gamma(x) = \frac{F(x_{best} - \mu(x))}{\sigma(x)}$, ϕ is the standard cumulative distribution function, ϕ the standard normal probability density function and k is a parameter allowing to balance exploration-exploitation.

Finally, Algorithm 1 depicts the different steps of the standard bayesian optimization algorithm.

Algorithm 1 Bayesian Optimization

```
1:  function Solve(problem,n,k)
2:      X =initRandom(n);
3:      Y =problem.evaluate(X)
4:      model=GP(X, Y)
5:      model.update()
6:      while not has_converged() do
7:          acq = getAcquisition(k);
8:          x_new =acq.optimize();
9:          y_new =problem.evaluate(x_new);
10:         model.update(x_new,y_new);
11:     end while
12:     return model.best;
13: end function
```

In the case of bi-level optimization, the UL fitness is complex to determine since it depends on the rational reaction of the LL decision maker. Therefore, even if $F(x, y)$ is a convex function, we can only consider all $F(x, \hat{y})$ with $\hat{y} \in P(x)$. In the case of two linear levels, IR is piecewise-linear and potentially discontinuous (see Figure 3). In this work, the LCB acquisition function will be preferred which allows to detect the most promising solutions in terms of mean and deviation. Large k values put the emphasis on exploration while small k values focus on the best expected performance.

3.3 Bayesian Optimization for Bi-level Problems (BOBP)

The contribution of this work is two-fold. First, we adapt the Bayesian algorithm (see algorithm 1 to bi-level problems). And finally, we improve the optimization of the acquisition function by considering a differential evolution algorithm.

3.3.1 Adaptation to Bi-level problems. Bi-level problems are evaluated in two-steps in order to compute UL function $F(x, y)$. Indeed, the UL decision maker has no control on y. He can only observe the LL rational decision $\hat{y} \in P(x)$ which strongly depends on x. In some ways, $F(x, y)$ can be rewritten by $F(x, \hat{y})$. The UL fitness function F is thus a function which only dependents on x. So we will now consider $F'(x) = F(x, \hat{y} = \arg\min\{F(x, y) : y \in P(x)\}$ in the singleton or the optimistic case while the pessimistic case would be $F'(x) = F(x, \hat{y} = \arg\max\{F(x, y) : y \in P(x)\}$. Figure 2 depicts a surrogate function obtained from the example described in Figure 3. The updated GP model allows us to plot at each UL decision x, the mean fitness value and a confidence interval around the mean value . We can distinguishes two types of UL solutions x:

- The first one represents the UL solutions which have been evaluated to compute $F'(x)$ according to the LL problem. This is the reason why the variance at these points is null. They have already been selected by the acquisition solver during the optimization.
- The second one represents the UL solutions which have not been explicitly evaluated. This is the reason why we only have a confidence interval around $F'(x)$. They are potential points to be selected by the acquisition solver.

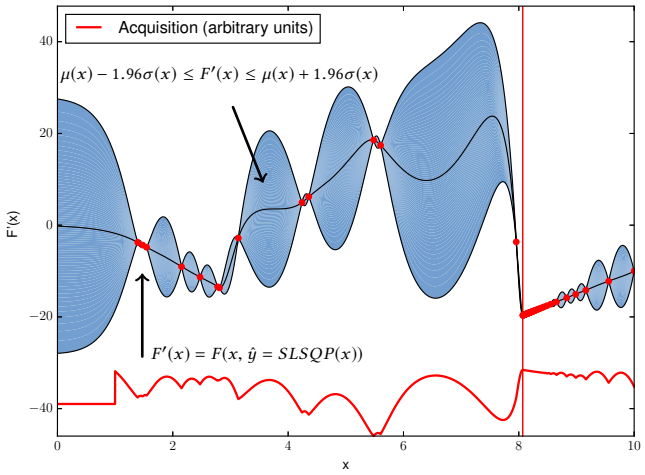

Figure 2: Surrogate function

In order to evaluate UL solution x and obtain $F'(x)$, we have to perform a LL optimization to determine $\hat{y}$. This optimization is realized using the Sequential Least Squares Programming (SLSQP) algorithm developed by Kraft [18]. SLSQP is a gradient-based procedure for non-linear optimization problems supporting inequality and equality constraints. This algorithm requires a initial guess to start the optimization. Instead of using a random initial guess, we select as starting point the LL optimal solution obtained by the closest UL solution x'. Figure 3 depicts this situation where we wish to optimize the LL problem according to the UL decision $x = 3$. The UL decision $x' = 2$ is the closest UL decision variables which has been explicitly evaluated so far. The rational decision set according to x' is $P(2) = \{3\}$. This strategy is based on the idea that two close UL decision variables should have closed optimal LL solutions. In this case, the number of evaluations required by the SLSQP algorithm is dramatically reduced.

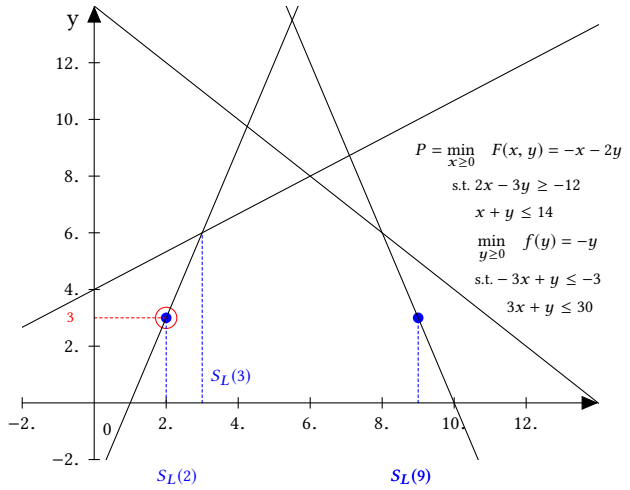

Figure 3: Select initial guess to solve $P(3) = \{\hat{y} \in Y : \hat{y} \in \arg\min[f(3, y) : y \in S_L(3)]\}$

3.3.2 Improving the acquisition function using differential evolution. As described in the previous section, Bayesian optimization is well-adapted to problem with time-consuming function evaluations. For instance, it has been used with success to optimize machine learning algorithms[16]. The main advantage is a clever search in order to limit the number of evaluations contrary to some other metaheuristics. As its name suggests it, the acquisition function allows to find the next point to evaluate according to some information gain obtained from the previous iterations. This is clearly a very efficient way of learning promising areas of the objective function. The acquisition function should be globally optimized since it is generally a function with possible several local maxima as depicted in Figure 4. Therefore and contrary to most of the Bayesian optimization implementation we have seen so far, we considered a differential evolutionary algorithm to search the global optimal solution becoming the next acquisition point. Such global algorithms have been successfully applied inside Bayesian optimization algorithms in [21] . Since we need to spare a maximum number of function evaluations, it is really important to find the most promising areas. Different strategies can be considered:

- Select the best acquisition point
- Select a set of promising acquisition point

In the second case, the number of function evaluations will increase but we are more likely to escape local optimal solutions. In this work, we only test the first case where only a single acquisition point is selected. Future investigations will focus on different strategies of optimizing the acquisition function.

4 NUMERICAL EXPERIMENTS

4.1 Comparison with BLEAQ on Bi-level Benchmarks

In this work, the 10 bi-levels problems proposed in [31] have been selected to evaluate the potential of Bayesian optimization to solve bi-level problems. Table 1 describes the best known fitnesses for

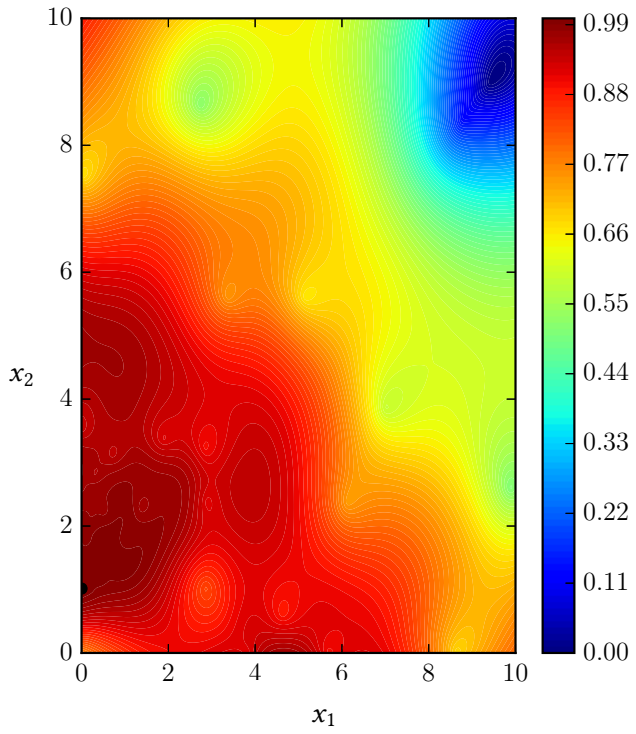

Figure 4: Acquisition landscape of two UL decision variables

these benchmarks at both levels. This set of benchmarks including linear and non-linear levels will afford us to compare the Bayesian approach with the BLEAQ algorithm. The latter is an evolutionary algorithm in which the LL decision variable are approximated by quadratic functions. The authors applied this strategy to minimize the number of LL optimization. To the best of our knowledge, the LL problem is optimized only when the approximation error exceeds a certain threshold.

Best known fitnesses	UL fitnesses	LL fitnesses
TP1	225.0	100.0
TP2	0.0	100.0
TP3	-18.6787	-1.0156
TP4	-29.2	3.2
TP5	-3.6	-2.0
TP6	-1.2091	7.6145
TP7	-1.96	1.96
TP8	0.0	100.0
TP9	0.0	1.0
TP10	0.0	1.0

Table 1: Benchmarks

4.2 Experimental protocol and parameters

For both approaches, experiments have been conducted on the High Performance Computing (HPC) platform of the university of Luxembourg[34]. Each run was completed on a single core of an Intel Xeon E3-1284L v3 @ 1,8 GHz, 32Gb of RAM server, which was dedicated to this task. The GPyOpt python library [1] has been selected to apply Bayesian optimization. The LCB acquisition function has been considered to determine the next promising point with parameter $k = 2$. The differential algorithm implemented to optimize the acquisition function is the one proposed by the python scipy library with default parameters: strategy='best1bin', maxiter=1000, popsize=15,mutation=(0.5, 1), recombination=0.7. The scipy library implements SLSQP as well. Concerning the BLEAQ algorithm, the authors provide a MATLAB code located at http://www.bilevel.org in the resources section. Unfortunately, the core of this MATLAB code has been obfuscated and we do not know what kind of local search algorithm is used in addition with their genetic algorithm. We kept the same parameters described in [31]. In addition, the stopping criteria for both algorithms is the convergence. The Bayesian algorithm proposed in this work uses the same criteria as BLEAQ except that the criteria is not applied on a population but on the last acquired points.

4.3 Experimental results

Numerical results have been summarized in Tables 2,4,3 and 5. The results for the BLEAQ algorithm for the 10 instances are the one reported in [31]. Tables 2 and 4 represent respectively the average and best fitnesses obtained over all runs. A Wilcoxon rank-sum test [36] has been realized to determine if the fitness differences are statistically significant for both levels. P-values have been provided in Table 3. In addition, Table 5 illustrates the average number of evaluations for both levels over all runs. The "ULcalls" column represents the average number of objective function calls at Upper-Level while the "LLcalls" column indicates the average aggregated number of objective function calls at Lower-Level. We can easily observe that the number of evaluations obtained with the Bayesian algorithm is much lower than for BLEAQ. This is the reason why no statistical tests have been computed in this case. For some benchmarks, BLEAQ achieves better accuracy. Indeed, BLEAQ makes use of an evolutionary algorithm to perform LL optimization while the Bayesian algorithm implemented to tackle these bi-level benchmarks only invokes a local search, i.e SLSQP. Nevertheless on TP9 benchmark, the best fitness at UL is better than the best known solution. For TP9 and TP10, the average fitnesses are better for the Bayesian algorithm despite the fact that these both benchmarks have been generated from complex single-level benchmarks with multiple local optimal solutions (see http://www.bilevel.org). We also noticed the ability of the Bayesian algorithm to face multi-modal problems. Indeed, the Bayesian algorithm provides two solutions with fitnesses $(F, f) = (0, 100)$ and $(F, f) = (0, 200)$ which both are UL optimal solutions. The same observation has been made for the TP8 benchmark. On the contrary, BLEAQ never detects the solution providing the fitness values $(0, 200)$. In a competitive and real-life application, such solution could be preferred by the UL decision maker since the fitness associated to the LL problem is definitely higher while still optimal for the UL decision maker. These

promising results show the advantage of such surrogate optimization algorithms over traditional evolutionary computing techniques when function evaluations are complex and time-consuming.

	Bayesian		BLEAQ	
Average	ULf itness	LLf itness	ULf itness	LLf itness
TP1	253.6155	70.3817	224.9989	99.9994
TP2	0.0007	183.871	2.4352	93.5484
TP3	−18.5579	−0.9493	−18.6787	−1.0156
TP4	−27.6225	3.3012	−29.2	3.2
TP5	−3.8516	−2.2314	−3.4861	−2.569
TP6	−1.2097	7.6168	−1.2099	7.6173
TP7	−1.6747	1.6747	−1.9538	1.9538
TP8	0.0008	180.6452	1.1463	132.5594
TP9	0.0012	1.0	1.2642	1.0
TP10	0.0049	1.0	0.0001	1.0

Table 2: Average fitnesses for both levels

p-values	UL fitnesses	LL fitnesses
TP1	5.09354939843e-08	6.62154466203e-08
TP2	1.99180208303e-05	0.000455639542651
TP3	1.33535344389e-11	1.33535344389e-11
TP4	1.33535344389e-11	0.827259346563
TP5	4.88497305946e-08	9.352489315e-05
TP6	1.33535344389e-11	3.21862967172e-09
TP7	1.06973503522e-10	1.06973503522e-10
TP8	1.91908665211e-09	0.00204782236
TP9	2.58028430416e-08	7.11788655392e-06
TP10	1.97034447118e-11	1.97034447118e-11

Table 3: Wilcoxon Rank-Sum Test for both levels

	Bayesian		BLEAQ	
Best values	ULf itness	LLf itness	ULf itness	LLf itness
TP1	225.0011	99.9984	225.0	100.0
TP2	0.0	200.0	5.4204	0.0
TP3	−18.6786	−1.0156	−18.6787	−1.0156
TP4	−29.1991	3.2001	−29.2	3.2
TP5	−3.8998	−2.0039	−2.4828	−7.705
TP6	−1.2099	7.6173	−1.2099	7.6173
TP7	−1.6833	1.6833	−1.8913	1.8913
TP8	0.0	200.0	12.2529	0.0007
TP9	0.0007	1.0	3.5373	1.0
TP10	0.0011	1.0	0.001	1.0

Table 4: Best obtained fitnesses for both levels

Figure 5 depicts the posterior mean, the standard deviation and the current acquisition function when TP5 optimization ended. The posterior mean shows that a large number of points concentrate around the best solution. Furthermore, the standard deviation is also very low in this area indicating a strong confidence on the location of the optimal solution. Finally, the landscape of the acquisition function suggests that the next acquired point would be probably in this area confirming the convergence.

	Bayesian		BLEAQ	
Average	ULcalls	LLcalls	ULcalls	LLcalls
TP1	211.1333	1558.8667	588.6129	1543.6129
TP2	35.2581	383.0645	366.8387	1396.1935
TP3	89.6774	1128.7097	290.6452	973.0
TP4	16.9677	334.6774	560.6452	2937.3871
TP5	57.2258	319.7742	403.6452	1605.9355
TP6	12.1935	182.3871	555.3226	1689.5484
TP7	72.9615	320.2308	494.6129	26 682.4194
TP8	37.7097	413.7742	372.3226	1418.1935
TP9	16.6875	396.3125	1512.5161	141 303.7097
TP10	21.3226	974.0	1847.1	245 157.9

Table 5: Average number of function evaluations for both levels

5 CONCLUSION AND FUTURE WORKS

Bi-level optimization problems are special kind of optimization problems involving two decisions makers. Closely related to Game theory, they are the mathematical representation of so called "Stackelberg games". These hierarchical problems require to optimize iteratively two problems called the Upper-Level and the Lower-Level because the decision variables are partially controlled by each of them. This nested optimization scheme is time-consuming. Different approaches have been proposed in the literature and are more and less efficient. Generally, they replace the LL problem with its KKT conditions to obtain a single-level problems which can be solved with traditional techniques. In this work, a surrogate-based algorithm, i.e. BOBP has been proposed to tackle this complex problems. Based on Gaussian processes, Bayesian optimization is a Black-box algorithm which has been intensively employed to optimize machine learning model (e.g. neural network, support vector machine) by creating a surrogate model which is fine-tune until the convergence occurs. Bayesian optimization makes no assumption on the characteristic of the optimized function to approximate it. It also embeds a mechanism to detect new promising points after refinement of the model at each iteration. These properties make Bayesian optimization a very good candidate algorithm to tackle hierarchical problems such as bi-level problems. Numerical experiments on 10 bi-level benchmarks confirmed that Bayesian optimization can dramatically reduce the number of evaluations and thus the number of LL optimizations while guaranteeing very good and accurate results. Future works will consider different algorithm to perform LL optimizations other than SLSQP. In fact, the combination of multiple an different LL optimization solver could be the best approach to do such optimizations. In addition, new optimization techniques could be proposed to solve the acquisition function. For example,multi-objective optimization could use to consider several acquisition functions instead of optimizing only one at time.

REFERENCES

[1] The GPyOpt authors. 2016. GPyOpt: A Bayesian Optimization framework in Python. http://github.com/SheffieldML/GPyOpt. (2016).
[2] J.F. Bard. 1983. Coordination of a multidivisional organization through two levels of management. 11 (1983), 457–468.

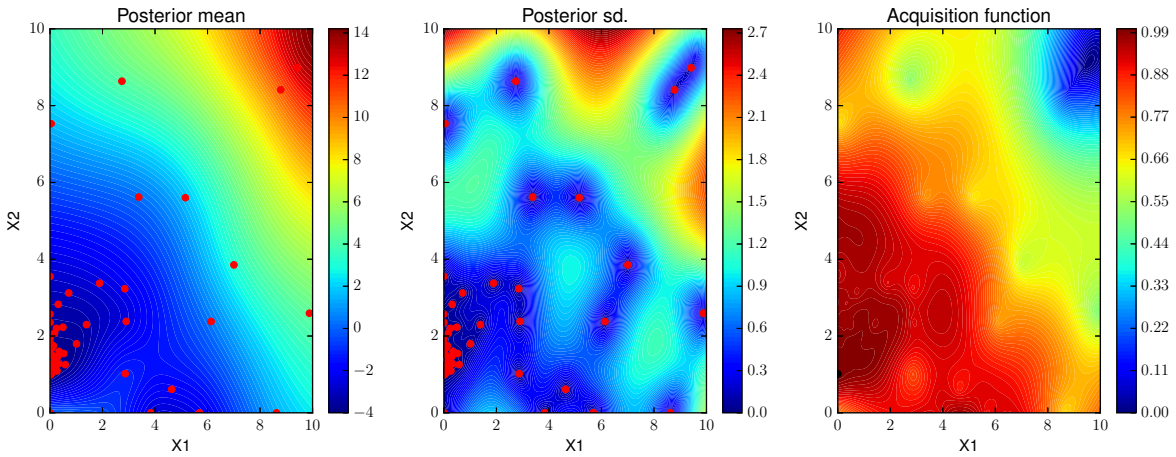

Figure 5: Gaussian process state after TP5 optimization

[3] J.F. Bard. 1991. Some properties of the bilevel programming problem. 68 (1991), 371–378. Technical Note.

[4] J.F. Bard and J. Falk. 1982. An explicit solution to the multi-level programming problem. 9 (1982), 77–100.

[5] J.F. Bard and J. Moore. 1990. A branch and bound algorithm for the bilevel programming problem. 11 (1990), 281–292.

[6] Z. Bi, P. Calamai, and A. Conn. 1989. *An exact penalty function approach for the linear bilevel programming problem.* Technical Report #167-O-310789. ftp://dial.uwaterloo.ca/pub/tech_reports

[7] W. Bialas and M. Karwan. 1982. On two-level optimization. 27 (1982), 211–214.

[8] L. Bianco, M. Caramia, and S. Giordani. 2009. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging Technologies* 17, 2 (2009), 175 – 196. https://doi.org/10.1016/j.trc.2008.10.001 Selected papers from the Sixth Triennial Symposium on Transportation Analysis (TRISTAN VI).

[9] J. Bracken and J. McGill. 1973. Mathematical programs with optimization problems in the constraints. 21 (1973), 37–44.

[10] J. Bracken and J. McGill. 1974. Defense applications of mathematical programs with optimization problems in the constraints. 22 (1974), 1086–1096.

[11] L. Brotcorne, M. Labbé, Patrice Marcotte, and G. Savard. 2001. A Bilevel Model for Toll Optimization on a Multicommodity Transportation Network. *Transportation Science* 35, 4 (2001), 345–358. https://doi.org/10.1287/trsc.35.4.345.10433 arXiv:http://dx.doi.org/10.1287/trsc.35.4.345.10433

[12] A. Chernyshev. 2016. *Bayesian Optimization of Spiking Neural Network Parameters to Solving the Time Series Classification Task.* Springer International Publishing, Cham, 39–45. https://doi.org/10.1007/978-3-319-32554-5_6

[13] J. Fliege and L.N. Vicente. 2006. Multicriteria Approach to Bilevel Optimization. *Journal of Optimization Theory and Applications* 131, 2 (2006), 209–225. https://doi.org/10.1007/s10957-006-9136-2

[14] P. Fontaine and S. Minner. 2014. Benders Decomposition for Discretefi?!-Continuous Linear Bilevel Problems with application to traffic network design. *Transportation Research Part B: Methodological* 70, 0 (2014), 163 – 172. https://doi.org/10.1016/j.trb.2014.09.007

[15] M.R. Garey and D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)* (first edition ed.). W. H. Freeman. http://www.amazon.com/Computers-Intractability-NP-Completeness-Mathematical-Sciences/dp/0716710455

[16] J.Snoek, H. Larochelle, and P.R. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.

[17] A. Koh. 2007. Solving transportation bi-level programs with Differential Evolution. In *2007 IEEE Congress on Evolutionary Computation*. 2243–2250. https://doi.org/10.1109/CEC.2007.4424750

[18] D. Kraft. 1988. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[19] F. Legillon, A. Liefooghe, and E. Talbi. 2012. CoBRA: A cooperative coevolutionary algorithm for bi-level optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 1–8. https://doi.org/10.1109/CEC.2012.6256620

[20] X Li, P. Tian, and X. Min. .

[21] Bo Liu, Qingfu Zhang, and Georges G. E. Gielen. 2014. A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization Problems. *Trans. Evol. Comp* 18, 2 (April 2014), 180–192. https://doi.org/10.1109/TEVC.2013.2248012

[22] A.G. Mersha and S. Dempe. 2006. Linear bilevel programming with upper level constraints depending on the lower level solution. *Appl. Math. Comput.* 180, 1 (2006), 247 – 254. https://doi.org/10.1016/j.amc.2005.11.134

[23] A. Migdalas. 1995. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization* 7, 4 (1995), 381–405. https://doi.org/10.1007/BF01099649

[24] M.Köppe, M. Queyranne, and C. T. Ryan. 2009. A parametric integer programming algorithm for bilevel mixed integer programs. *ArXiv e-prints* (July 2009). arXiv:math.OC/0907.1298

[25] J. Mockus. 2012. *Bayesian approach to global optimization: theory and applications.* Vol. 37. Springer Science & Business Media.

[26] José Luis Morales and Jorge Nocedal. 2011. Remark on &Ldquo;Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound Constrained Optimization&Rdquo;. *ACM Trans. Math. Softw.* 38, 1, Article 7 (Dec. 2011), 4 pages. https://doi.org/10.1145/2049662.2049669

[27] C.E. Rasmussen and C.K.I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press.

[28] C.E. Rasmussen and C. K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press.

[29] K.H. Sahin and A.R. Ciric. 1998. A dual temperature simulated annealing approach for solving bilevel programming problems. *Computers and Chemical Engineering* 23, 1 (1998), 11 – 25. https://doi.org/10.1016/S0098-1354(98)00267-1

[30] G. Savard and J. Gauvin. 1994. The steepest descent direction for the nonlinear bilevel programming problem. 15 (1994), 275–282.

[31] A. Sinha, P. Malo, and K. Deb. 2014. An improved bilevel evolutionary algorithm based on Quadratic Approximations. In *2014 IEEE Congress on Evolutionary Computation (CEC)*. 1870–1877. https://doi.org/10.1109/CEC.2014.6900391

[32] H. Stackelberg. 1952. *The theory of the market economy.* 328 pages.

[33] E.G. Talbi. 2013. *A Taxonomy of Metaheuristics for Bi-level Optimization.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–39. https://doi.org/10.1007/978-3-642-37838-6_1

[34] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. 2014. Management of an Academic HPC Cluster: The UL Experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*. IEEE, Bologna, Italy, 959–967.

[35] W.Candler, J.Fortuny-Amat, and B.McCarl. 1981. The Potential Role of Multilevel Programming in Agricultural Economics. *American Journal of Agricultural Economics* 63, 3 (1981), 521–531. https://doi.org/10.2307/1240543 arXiv:http://ajae.oxfordjournals.org/content/63/3/521.full.pdf+html

[36] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* (1945), 80–83.