Parallel Surrogate-assisted Optimization: Batched Bayesian Neural Network-assisted GA versus q-EGO

Guillaume Briffoteaux, Maxime Gobert, Romain Ragonnet, Jan Gmys, Mohand Mezmaz, Nouredine Melab, Daniel Tuyttens

Parallel Surrogate-assisted Optimization: Batched Bayesian Neural Network-assisted GA versus q-EGO

Guillaume Briffoteaux[a,c], Maxime Gobert[a,c], Romain Ragonnet[b], Jan Gmys[a,c], Mohand Mezmaz[a], Nouredine Melab[c], Daniel Tuyttens[a]

[a]*Mathematics and Operational Research Department (MARO), University of Mons, Belgium*
[b]*School of Public Health and Preventive Medicine, Monash University, Australia*
[c]*Inria Lille - Nord Europe, CNRS/CRIStAL, University of Lille, France*

Abstract

Surrogate-based optimization is widely used to deal with long-running black-box simulation-based objective functions. Actually, the use of a surrogate model such as Kriging or Artificial Neural Network allows to reduce the number of calls to the CPU time-intensive simulator. Bayesian optimization uses the ability of surrogates to provide useful information to help guiding effectively the optimization process. In this paper, the Efficient Global Optimization (EGO) reference framework is challenged by a Bayesian Neural Network-assisted Genetic Algorithm, namely BNN-GA. The Bayesian Neural Network (BNN) surrogate is chosen for its ability to provide an uncertainty measure of the prediction that allows to compute the Expected Improvement of a candidate solution in order to improve the exploration of the objective space. BNN is also more reliable than Kriging models for high-dimensional problems and faster to set up thanks to its incremental training. In addition, we propose a batch-based approach for the parallelization of BNN-GA that is challenged by a parallel version of EGO, called q-EGO. Parallel computing is a highly important complementary way (to surrogates) to deal with the computational burden of simulation-based optimization. The comparison of the two parallel approaches is experimentally performed through several benchmark functions and two real-world problems within the scope of Tuberculosis Transmission Control (TBTC). The study presented in this paper proves that parallel batched BNN-GA is a viable alternative to q-EGO approaches being more suitable for high-dimensional problems, parallelization impact, bigger data-bases and moderate search budgets. Moreover, a significant improvement of the solutions is obtained for the two TBTC problems tackled.

Keywords: Surrogate-assisted Optimization, Bayesian Optimization, Efficient Global Optimization, Simulation, Massively Parallel Computing, Evolutionary Algorithm.

Email addresses: guillaume.briffoteaux@umons.ac.be (Guillaume Briffoteaux), maxime.gobert@umons.ac.be (Maxime Gobert), romain.ragonnet@monash.edu (Romain Ragonnet), jan.gmys@umons.ac.be (Jan Gmys), mohand.mezmaz@umons.ac.be (Mohand Mezmaz), nouredine.melab@univ-lille.fr (Nouredine Melab), daniel.tuyttens@umons.ac.be (Daniel Tuyttens)

1. Introduction

Finding the optimum point from a landscape generated by a CPU time-intensive black-box simulator is not an easy task. The characteristics of the fitness space are unknown and difficult to gauge. Moreover, when the simulation involves the resolution of differential equations, as it is the case in studies focused on dynamical processes [1], it is reasonable to assume the relation between input-output pairs to be non-linear and non-convex.

To deal with these complicated optimization problems, parallel optimization methods have been proposed [2]. Indeed, the development of heterogeneous supercomputers [3] now allows to run several complex models in parallel. These iterative approaches are composed of the three main steps presented in Figure 1: the initial Design of Experiment (DoE), the Acquisition Process and the parallel simulations. The DoE method initiates the search by sampling the search space. The initial candidate solutions are then simulated in parallel. Based on the database of known solutions, the Acquisition Process proposes a batch of new candidates. The long duration involved to run a simulation directly suggests to evaluate the candidate solutions in parallel.

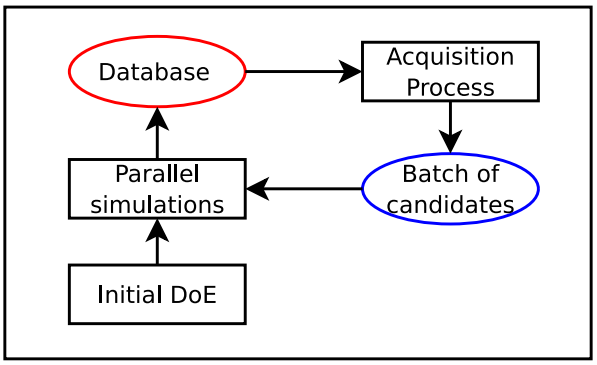

Figure 1: Structure of parallel methods to optimize long-running black-box simulator. The red ellipse represents a set of simulated solutions. The blue ellipse represents a set of solutions to simulate.

Along with parallel computations, surrogate models are introduced as a complementary way to endure the high simulation duration [4] [5] [6]. Surrogate models, also called approximated models or meta-models, aim at imitating the behavior of the simulator in timely fashion but provide a lower accuracy. The surrogate is generally a machine learning algorithm trained from the database of already known solutions. Its predictions may help exclusively the Acquisition Process to identify promising candidate solutions as depicted in Figure 2. This kind of integration is called Indirect Fitness Replacement, according to the classification proposed in [7]. Its predictions may also help at the evaluation step, depicted in Figure 3, where the Acquisition Process produces a batch of predicted solutions. This kind of integration is qualified as Direct Fitness Replacement [7].

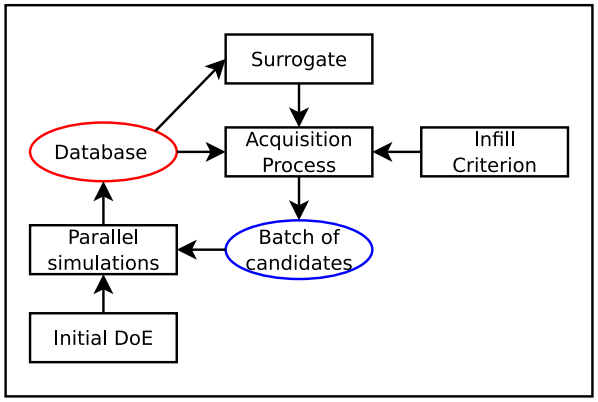

Figure 2: Structure of Indirect Fitness Replacement parallel surrogate-assisted optimization methods. The red ellipse represents a set of simulated solutions. The blue ellipse represents a set of solutions to simulate.

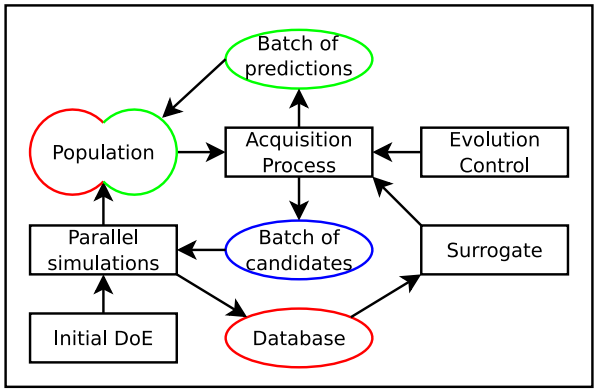

Figure 3: Structure of Direct Fitness Replacement parallel surrogate-assisted optimization methods. The colorful shapes represent sets of solutions. Red shapes are made of simulated solutions, green shapes are made of predicted solutions and the blue ellipse is made of solutions to simulate.

The Infill Criterion or Evolution Control describes the qualities that a solution should present in order to be considered as a candidate solution to be simulated. In other terms, it reflects the desirability of a candidate solution. Several Infill Criteria and Evolution Controls that focus either on exploration, exploitation or on a trade-off between both have been proposed [8] [9]. Infill Criteria, introduced by Kushner in 1964 [10] and popularized by Jones *et al.* in 1998 [11], rely on surrogates that provide uncertainty information around their predictions. Training such surrogates can be computationally expensive since it consists in determining the probability distribution that produces the surrogate parameters and not the parameter values directly [12]. In so-called Bayesian Optimization (BO) methods [13] the Acquisition Process consists in optimizing the Infill Criterion to select candidate solutions. Evolution Controls, introduced by Jin in 2002 [14], are tied to evolutionary algorithms. Using Direct Fitness Replacement, the Evolution Control decides which are the candidates to be simulated and which are the ones to be predicted by the surrogate. Such methods are classified as Surrogate-Assisted Evolutionary Algorithms (SAEA) [4]. In this paper, we focus on the well-known Expected Improvement (EI) Infill Criterion [11] often used in Bayesian Optimization (e.g. EGO), and propose it as an Evolution Control in a parallel SAEA.

In this context several challenges arise. EI is the core component of the Efficient Global Optimization (EGO) method [11], a BO method that relies on the Kriging meta-model. Kriging is known for the uncertainty information it provides and its complicated training when the number of input variables and training samples increase [15]. In Surrogate-Assisted Genetic Algorithms (SAGA) [16], the surrogate used is often an Artificial Neural Network (ANN) [17] [18] [19]. Contrary to Kriging, ANNs demonstrate an easier training but do not provide uncertainty information around their predictions [20]. Since the computation of EI requires the uncertainty information, the integration of EI as Evolution Control in an ANN-assisted Genetic Algorithm (GA) is challenging.

Computational power of super-computers may be favorable to BO and SAEA approaches when tackling optimization problems involving CPU time-intensive black-box simulators. The scalability of parallel variants of EGO and SAGA [6] has to be studied to highlight their respective strengths and limitations. Moreover, in a context defined by a given computational budget, a limited simulation computational burden and a given landscape to optimize, it is not clear which optimization method to choose. A computationally intensive comparison between parallel EGO and SAGA should be led contemplating diverse contexts.

The studies presented in this paper take up the challenges detailed previously. The main contributions are the following:

- A new parallel approach integrating the Expected Improvement as Evolution Control in a Genetic Algorithm is proposed. Development of Bayesian Neural Networks based on Monte Carlo Dropout (MCDropout-based BNNs) proved to retain the best of both Kriging and ANNs meta-models [21] [22]. A MCDropout-based BNN is then adopted as surrogate to realize the integration of EI. The new parallel method is referred to as Parallel Batched BNN-assisted GA in the rest of the paper.

- The new method is compared to a parallel variant of Kriging-based EGO, named Parallel Kriging-based q-EGO in the remainder of this paper. The comparison is realized considering benchmark functions under different search budgets and two real-world applications.

- The recent emergence of mathematical models describing the transmission of infectious diseases open new horizons to control epidemics. The computationally-intensive simulator proposed in [1] [23] allows to evaluate the effectiveness of an intervention plan to decrease the tuberculosis prevalence in a given country [24] [25]. The calibration of this tuberculosis transmission dynamic model is realized with success in this study thanks to Parallel Kriging-based q-EGO and Parallel Batched BNN-assisted GA.

- Finally, we successfully apply the considered surrogate-assisted methods on the search for the optimal distribution of a limited number of preventive treatments across several categories of a population. The objective lies in decreasing the tuberculosis prevalence in the Philippines. While the high efficiency of preventive treatment in such high incidence settings was previously demonstrated in [26], the optimal approach to its implementation remains unclear.

The remainder of this article is organized as follows. The next section details the Parallel q-EGO algorithm based on the Expected Improvement and the Kriging surrogate model. In Section 3, we present the Parallel Batched BNN-assisted GA based on Expected Improvement. The two algorithms considered are compared on several benchmark problems in Section 4 and in problems related to Tuberculosis Transmission Control (TBTC) in Section 5. Finally, conclusions are drawn in Section 7 and directions for future works are suggested.

2. Parallel Kriging-based q-EGO

2.1. Framework

Efficient Global Optimization is a framework developed by Jones *et al.* [11]. It is a BO technique that uses the Expected Improvement metric as Infill Criterion and where the surrogate operates indirectly within the optimization process as depicted in Figure 2. The classical EGO algorithm is sequential and does not allow parallel simulation of candidate solutions.

In order to solve this obstacle, Ginsbourger *et al.* proposed in [27] a multi-points version of EGO, presented in Algorithm 1 and called q-EGO. This alternative allows parallel simulation of a batch of q candidates. q-EGO runs several cycles that are repeated until the search budget runs out. A cycle, presented by lines 5 to 11 in Algorithm 1, is composed of three steps. The first step is the built-up of the Kriging model thanks to the database of already simulated solutions. Immediately afterwards, the Acquisition Process, based on the surrogate and the best cost found so far, proposes a new batch of candidate points. Finally, the candidates are evaluated and the database is enriched.

Algorithm 1 Framework of the q-EGO algorithm

Input
simulator: real fitness function
n: initial sampling size
$\mathcal{D}$: solution space
budget: budget for the search
surrogate: Kriging meta-model
q: number of solutions per batch

1: $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}] \leftarrow$ initial_sampling(n, $\mathcal{D}$)
2: $\boldsymbol{y} \leftarrow$ parallel_simulation(*simulator*, $\boldsymbol{X}$)
3: $y_{min} \leftarrow$ get_best_cost($\boldsymbol{y}$)
4: **while** *budget* $\neq 0$ **do**
5: $\quad \boldsymbol{X}_c \leftarrow \varnothing$ ▷ batch of candidate solutions
6: $\quad$ *surrogate* $\leftarrow$ fit_Kriging_model ($\boldsymbol{X}, \boldsymbol{y}$)
7: $\quad \boldsymbol{X}_c \leftarrow$ acquisition_process(q, $\mathcal{D}$, *surrogate*, $\boldsymbol{X}$, $\boldsymbol{y}$, y_{min})
8: $\quad \boldsymbol{y}_c \leftarrow$ parallel_simulation(*simulator*, $\boldsymbol{X}_c$)
9: $\quad \boldsymbol{X} \leftarrow \boldsymbol{X} \cup \boldsymbol{X}_c$
10: $\quad \boldsymbol{y} \leftarrow \boldsymbol{y} \cup \boldsymbol{y}_c$
11: $\quad (\boldsymbol{x}_{min}, y_{min}) \leftarrow$ get_best($\boldsymbol{X}$, $\boldsymbol{y}$)
12: **end while**
13: **return** $\boldsymbol{x}_{min}$, y_{min}

Bayesian Optimization relies on efficient surrogate models that give information on the variance around their prediction. In the next subsection, we present the Kriging model used as surrogate in this study.

2.2. Surrogate Building using Kriging models

In order to determine which surrogate model is better fitted within an EI-based EGO approach, a comparative study is led in [28]. Support Vector Regression, Radial Basis Function, Kriging, linear Shepard and an ensemble surrogates made of these latter are integrated into EGO and compared on 2-D and 6-D test functions and on a 4-D engineering problem. The outcomes of the experiments demonstrate the robustness of the Kriging-EI coupling over the landscapes to optimize.

Kriging has been first developed in the field of geostatistics by D. G. Krige [29] and formalized by G. Matheron in [30]. The main idea of the method is to use the spatial correlation between data to build the Best Linear Unbiased Predictor. Intuitively, spatial correlation can be represented as a way to measure how sensible is the observation according to the location variation.

One observation y at location $\boldsymbol{x} \in \mathcal{D}$, writes as the sum of a trend function $\mu(\boldsymbol{x})$ and a centered Gaussian Process $z(\boldsymbol{x})$:

$$y(\boldsymbol{x}) = \mu(\boldsymbol{x}) + z(\boldsymbol{x}) \tag{1}$$

Let us denote a set of observations as $\boldsymbol{y}$= $(y^{(1)}, \ldots, y^{(n)})$ at locations $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)})$. In other words, $y(\boldsymbol{x}^{(i)}) = y^{(i)} \in \mathbb{R}$ is the output of a simulation for the point $\boldsymbol{x}^{(i)} \in \mathcal{D}$.

For this study, we used Kriging with trend, also named Universal Kriging (UK). The trend $\mu(\boldsymbol{x})$ is assumed to be the sum of a linear combination of basis functions, so that Equation 1 becomes:

$$y(\boldsymbol{x}) = \boldsymbol{\beta}.\boldsymbol{\phi}(\boldsymbol{x}) + z(\boldsymbol{x}) \tag{2}$$

The $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_L)$ vector contains the coefficients of the combination of the L trend functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_L)$. The basis functions are arbitrarily chosen by the user while the coefficients are determined optimally during the Kriging model fitting. In order to express the spatial correlation between observations $\boldsymbol{u}$ and $\boldsymbol{v}$, a covariance function $C(\boldsymbol{u}, \boldsymbol{v})$ is defined. The covariance function used for this article is built upon the following expression from Roustant *et al.* [31]:

$$c(\boldsymbol{h}) = C(\boldsymbol{u}, \boldsymbol{v}) = \sigma^2 \prod_{j=1}^{d} \gamma(h_j, \theta_j) \tag{3}$$

where γ is the kernel function, d is the dimension, $\boldsymbol{h} = \boldsymbol{u} - \boldsymbol{v}$ and $\sigma = C_{i,i} = C\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}\right) = \mathbb{V}ar\left(\boldsymbol{x}^{(i)}\right)$ (assumed constant over the search space). $\boldsymbol{\theta}$ is a vector of hyper-parameters that must be fitted. This is often done by numerical Maximum Likelihood Estimation. The kernel function is chosen beforehand among well known forms. Indeed, to get the resulting covariance matrix positive definite the kernel has to respect some properties. One can find those conditions in N. Cressie's book [32], and examples of kernel functions in [31]. This allows to compute the covariance matrix $\boldsymbol{C} = \left(C(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})\right)_{1 \leq i,j \leq n}$.

Let us assume we want to predict the output at location $\boldsymbol{x}^*$, the Kriging prediction writes $\hat{y}(\boldsymbol{x}^*) = \hat{y}^* = \sum_{i=1}^{n} \omega_i y^{(i)}$, which re-writes according to Equation 2 as

$$\hat{y}^* = \sum_{i=1}^{n} \omega_i \sum_{l=1}^{L} \beta_l \phi_l(\boldsymbol{x}^{(i)}) + z^{(i)} \tag{4}$$

The Kriging prediction is the Best Linear Unbiased Predictor, hence it satisfies:

$$\hat{y}^* = argmin(\mathbb{V}ar[\hat{y}^* - y^*]) \text{ such that } \mathbb{E}[\hat{y}^* - y^*] = 0 \tag{5}$$

Solving this optimization problem corresponds to the fitting of the Kriging model. Let us call $\hat{s}(\boldsymbol{x}^*)$ the variance around $\hat{y}^*$. The solution of the problem gives the Universal Kriging predictor [31] which writes

$$\hat{y}^* = \boldsymbol{\phi}(\boldsymbol{x}^*)\bar{\boldsymbol{\omega}}^T + \boldsymbol{c}(\boldsymbol{x}^*)^T \boldsymbol{C}^{-1}(\boldsymbol{y} - \Phi\bar{\boldsymbol{\omega}}) \tag{6}$$

$$\begin{aligned} \hat{s}(\boldsymbol{x}^*) = {} & C(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{c}(\boldsymbol{x}^*)^T \boldsymbol{C}^{-1} \boldsymbol{c}(\boldsymbol{x}^*) \\ & + (\boldsymbol{\phi}(\boldsymbol{x}^*)^T - \boldsymbol{c}(\boldsymbol{x}^*)^T \boldsymbol{C}^{-1} \boldsymbol{\Phi})^T (\boldsymbol{\Phi}^T \boldsymbol{C}^{-1} \boldsymbol{\Phi})^{-1} \\ & (\boldsymbol{\phi}(\boldsymbol{x}^*)^T - \boldsymbol{c}(\boldsymbol{x}^*)^T \boldsymbol{C}^{-1} \boldsymbol{\Phi}) \end{aligned} \tag{7}$$

with $\bar{\boldsymbol{\omega}} = (\boldsymbol{\Phi}^T \boldsymbol{C}^{-1} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{C}^{-1} \boldsymbol{y}$, and $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}^{(1)}), \ldots, \boldsymbol{\phi}(\boldsymbol{x}^{(n)}))$, $\boldsymbol{\phi}(\boldsymbol{x})$ being the vector of trend functions values at location $\boldsymbol{x}$, $\boldsymbol{C}$ the covariance matrix and $\boldsymbol{c}(\boldsymbol{x}^*) = C(\boldsymbol{x}^*, \boldsymbol{x}^{(i)})$, $i = 1, \ldots, n$ the correlation vector.

The Kriging model has been widely used in Bayesian Optimization thanks to its ability to provide a variance for the prediction of the model. This variance helps guiding the optimization process over the search space.

2.3. Multi-point Acquisition Process within q-EGO

The Acquisition Process proposed in q-EGO by Ginsbourger *et al.* [27] is presented in Algorithm 2. The process operates iteratively to propose q new candidate solutions as shown by lines 3 to 9 of Algorithm 2. At each iteration, the EI Infill Criterion is maximized thanks to a GA coupled with the Kriging meta-model to produce a new candidate that is then predicted. To provide q different points, the EI function needs to be updated with a new Kriging model at each iteration. The Kriging model is then updated considering the prediction of the candidate point revealed by the EI. It allows lowering the EI values in the area of already selected points, so that the next candidate maximizing EI will be different than the previous one. This heuristic is called Kriging Believer and allows to select q new candidates without any simulation.

However, the Kriging Believer has an extra cost: q model updates and q EI maximizations to select a batch of q points, using the heuristic is essential to avoid the optimization of the analytical multi-points Expected Improvement. Indeed, even if the maximization of the multi-points criteria is possible and enables the determination of q points simultaneously, the process becomes prohibitively expensive when q increases. It is explained in [33] that the optimization of a problem of dimension $d * q$ must be carried out. Furthermore, even though fitting the Kriging model q times per cycle is also time consuming, due to matrix algebra operations executed on big matrices [20], in our study the number of decision variables and the total number of simulations are limited. Fitting the Kriging model is thus negligible compared to a simulation.

Algorithm 2 Acquisition Process in q-EGO

Input
q : number of candidate solutions per batch
$\mathcal{D}$: solution space
surrogate: Kriging model
$\boldsymbol{X}$, $\boldsymbol{y}$: already simulated solutions and their respective cost
y_{min}: best simulated cost found so far
GA: genetic algorithm

1: $\boldsymbol{X}_c \leftarrow \varnothing$ ▷ batch of candidate solutions
2: $\hat{\boldsymbol{y}} \leftarrow \varnothing$
3: **for** $i = 1 : q$ **do**
4: $\quad \boldsymbol{x}_{new} \leftarrow$ EI_maximization(*surrogate*, y_{min}, $\mathcal{D}$, *GA*)
5: $\quad \hat{y}_{new} \leftarrow$ prediction(*surrogate*, $\boldsymbol{x}_{new}$)
6: $\quad \boldsymbol{X}_c \leftarrow \boldsymbol{X}_c \cup \boldsymbol{x}_{new}$
7: $\quad \hat{\boldsymbol{y}} \leftarrow \hat{\boldsymbol{y}} \cup \hat{y}_{new}$
8: $\quad$ *surrogate* $\leftarrow$ fit_Kriging_model $(\boldsymbol{X} \cup \boldsymbol{X}_c, \boldsymbol{y} \cup \hat{\boldsymbol{y}})$ ▷ Kriging Believer
9: **end for**
10: **return** $\boldsymbol{X}_c$

The Expected Improvement used in the Acquisition Process of q-EGO is computed thanks to the variance that is provided by the Kriging model, at Equation 7. Considering a single-objective cost function that needs to be minimized, let $\boldsymbol{x}$ be a solution, y_{min} the best cost found so far, $\hat{y}(\boldsymbol{x})$ the predicted cost provided by the surrogate and $\hat{s}(\boldsymbol{x})$ the variance around the surrogate prediction. The Expected Improvement for the candidate solution $\boldsymbol{x}$ is given by

$$EI(\boldsymbol{x}) = (y_{min} - \hat{y}(\boldsymbol{x}))\Phi\left(\frac{y_{min} - \hat{y}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) + \hat{s}(\boldsymbol{x})\phi\left(\frac{y_{min} - \hat{y}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) \tag{8}$$

where Φ is the cumulative distribution function for the normal law and ϕ is its probability density function. Over the search space, the EI value is high on regions of improvement, enhancing exploitation, and high on regions of high uncertainty, enhancing exploration.

Expected Improvement is a widespread Infill Criterion in the literature. In [34], the authors consider six Infill Criteria within the Kriging-based EGO framework. Maximization of EI is compared to

minimization of the predicted cost, minimization of the Lower Confidence Bound, maximization of the Probability of Improvement and maximization of the Mean Squared Error. The optimization problems tackled consist in minimizing the drag of an airfoil represented by 8 to 20 decision variables with evaluation budgets ranging from 80 to 200. EI shows to perform consistently well over the instances treated. Noe and Husmeier propose a new Infill Criterion in [35]. Arguing that EI does not take into account the uncertainty in the improvement value, they defined a new acquisition function that outperforms EI on 1-D and 2-D benchmark functions. In [36], it is proposed to select candidate solutions according to the information gain they should provide. A novel algorithm based on the entropy metric, which value indicates the quantity of information carried by a solution, is conceived. The approach reveals slight improvement over EI on 2-D test functions only and a computational extra charge in comparison with EGO.

The inner part of the Acquisition Process considered in this paper for q-EGO is described in Algorithm 3. It is decided to resort to a GA as it allows to take into account linear constraints by designing its reproduction operators. Such constraints appear in one of the TBTC problems tackled in Section 5. The procedure begins at line 1 in Algorithm 3 by the creation of the population. During a generation, the reproduction loop presented by lines 5 to 12 in Algorithm 3 consists in generating new candidates called children. A random permutation is performed to create couples of parents. This ensures that all parents have the possibility to pass down their genes since we are looking for keeping a maximum of gene diversity. From a couple of parent solutions, two children are generated with the crossover operator, and go through mutation operator that occurs at a certain rate. Finally, children are evaluated and a selection is carried out between the two parents and two children to replace the parents in the population. This operation is repeated for each couple of parents in the population in order to form a new population of the same size so that we can go back to parent matching at line 6 and pursue until the maximum number of generations is reached.

Algorithm 3 Maximization of Expected Improvement by GA

Input
- n_{pop}: population size
- n_{gen}: number of generations
- η: mutation rate
- $\mathcal{D}$: solution space
- $(\hat{y}(), \hat{s}())$: surrogate prediction and variance functions
- y_{min}: best simulated cost found so far

1: $\mathcal{P} \leftarrow$ create_population(n_{pop}, $\mathcal{D}$)
2: $\boldsymbol{x}_{max} \leftarrow \text{argmax}_{\mathcal{P}}(\text{EI}(\boldsymbol{x}, y_{min}, \hat{y}(), \hat{s}()))$
3: **for** $i = 1 : n_{gen}$ **do**
4: $\quad\mathcal{P}_{new} \leftarrow \varnothing$
5: $\quad$**for** $j = 1 : n_{pop}/2$ **do**
6: $\quad\quad(\boldsymbol{p}_1^j, \boldsymbol{p}_2^j) \leftarrow$ select_parents($\mathcal{P}$) ▷ draw without replacement
7: $\quad\quad(\boldsymbol{c}_1, \boldsymbol{c}_2) \leftarrow$ crossover($\boldsymbol{p}_1^j$, $\boldsymbol{p}_2^j$, $\mathcal{D}$)
8: $\quad\quad(\boldsymbol{c}_1, \boldsymbol{c}_2) \leftarrow$ mutation($\boldsymbol{c}_1$, $\boldsymbol{c}_2$, η, $\mathcal{D}$)
9: $\quad\quad(\boldsymbol{c}_1, \boldsymbol{c}_2) \leftarrow$ compute_EI($(\boldsymbol{c}_1, \boldsymbol{c}_2)$, y_{min}, $\hat{y}()$, $\hat{s}()$)
10: $\quad\quad(\boldsymbol{p}_1^j, \boldsymbol{p}_2^j) \leftarrow$ replacement($\boldsymbol{p}_1^j$, $\boldsymbol{p}_2^j$, $\boldsymbol{c}_1$, $\boldsymbol{c}_2$)
11: $\quad\quad\mathcal{P}_{new} \leftarrow \mathcal{P}_{new} \cup (\boldsymbol{p}_1^j, \boldsymbol{p}_2^j)$
12: $\quad$**end for**
13: $\quad\mathcal{P} \leftarrow \mathcal{P}_{new}$
14: $\quad\boldsymbol{x}_{max} \leftarrow \text{argmax}_{\mathcal{P}}(\text{EI}(\boldsymbol{x}, y_{min}, \hat{y}(), \hat{s}()))$
15: **end for**
16: **return** $\boldsymbol{x}_{max}$

3. Parallel Batched BNN-assisted GA

3.1. Framework

The SAEA employed in this study is a Surrogate-Assisted Genetic Algorithm. In a GA, an initial population of N simulated solutions is iteratively evolved by reproduction, thanks to crossover and mutation operators, and elitist replacement [2]. From one generation to another, the population covers different regions of the search space with the goal of finding the best promising region.

EGO exhibits strong performances over a substantial number of applications. But, as stated by the No Free Lunch Theorem, there exists cases where EGO faces difficulties to optimize such as multi-modal landscapes and solution spaces of high dimension. In [37], a population of solutions is evolved to solve multi-objective problems. The reproduction step consists in several local searches starting from several points. The starting points are solutions extracted from the population and presenting a hypervolume contribution and a favorable Non-Dominated Rank. The method is compared to ParEGO, a multi-objective version of EGO, on eleven test problems of 8, 16 and 24 decision variables with an evaluation budget set to 400. The results indicate the superiority of the evolutionary-like method over ParEGO, especially in high dimensions. Another interesting comparison is the one between GA and BO methods realized in [38]. Tackling a combinatorial optimization problem, it is shown that GAs outperform the BO method.

In [39], we proposed a BNN-assisted GA and applied it successfully on a TBTC problem. The Evolution Controls implemented either focused on exploitation as the minimization of the predicted cost or on exploration as the maximization of the MCDropout-based BNN uncertainty and the maximization of the distance to the already simulated solutions. The Evolution Control using MCDropout-based BNN uncertainty appeared to be the most successful as it allows the surrogate to improve its accuracy. The coupling between a GA and a BNN has also been put forward in other scientific fields such as bioinformatics. In [40], the proposed coupling aims at selecting the best subset of a substantial amount of data and the best BNN classifier in order to perform species classification.

Involvement of long-running black-box simulator naively suggests for parallel simulations as it is the case in [18]. In this study ANN approximations and uncertainties are used to optimize a factory production planning maximizing devices utilization and minimizing tardiness. In a parallel multi-objective evolutionary algorithm, offsprings are evaluated and ranked according to the meta-model, the ones marked with the better ranks are re-evaluated in parallel with the original cost function before being inserted into the population. The ANN prediction error committed on the parents is incorporated into the offspring rank. Approaches presented previously such as in [37] also rely on parallelization. Here, on each CPU core, a local search is run and its resulting candidate solution is simulated. In [39], the batch of solutions to simulate, produced according to the Evolution Control, is simulated in parallel.

In Parallel Batched BNN-assisted GA, whose framework is described by Algorithm 4, the generational concept of GA is kept and batches of solutions are introduced in order to allow parallel simulations. A batch can be seen as a subset of the GA population. Besides, a database containing all the simulated solutions along with their respective cost is maintained to enable surrogate update. The initial database is constituted with N solutions sampled randomly from the solution space. The initial database is used as the initial population of the GA and as initial training set for the surrogate.

One GA generation is represented by lines 6 to 20 in Algorithm 4. At each generation, the population of parents of the GA is partitioned into M batches $(\mathcal{B}_{p1}, \ldots, \mathcal{B}_{pM})$ of n_{batch} parent solutions each. Each batch of parents is fed into the Acquisition Process which returns a batch $\mathcal{B}_{c,sim}$ made of $n_{batch} * p_{sim}$ children to be simulated and a batch $\mathcal{B}_{c,pred}$ made of $n_{batch} * (1 - p_{sim})$ children to be predicted. $p_{sim} \in [0, 1]$ is called the proportion of simulations. The batch $\mathcal{B}_{c,sim}$ is then simulated in parallel, assuming one simulation per CPU core, and the new simulations are added to the database. Once simulations and predictions are done, both batches $\mathcal{B}_{c,sim}$ and $\mathcal{B}_{c,pred}$ are added into the population of children $\mathcal{P}_c$. The surrogate is then updated, in an incremental way, thanks to a training set composed of the last N simulations from the database.

After treating the last batch of parents $\mathcal{B}_{pM}$ from a given generation, both the population and the population of children are merged and only the N best solutions according to elitist replacement are

kept to form the population for the next generation. This selection is represented by line 20 in Algorithm 4. Except for the very first generation, the population possibly embeds predicted individuals. Finally, the search ends when the allocated budget, usually expressed in execution time or number of simulations, is wasted.

The parallelization possibilities are restricted to the parallel simulation of multiple solutions. Indeed, in long-running black-box optimization, the simulator can last from several seconds to several hours [41]. Other operations as reproduction or prediction are negligible in comparison to simulation.

The reproduction operators, the concept of generation for a population of size N and the number of available computing cores n_{cores} imply some constraints when setting the number of batches per generation M, the number of solutions per batch n_{batch} and the proportion of simulations per batch p_{sim}. As the reproduction operators produce a couple of children generated from a couple of parents, n_{batch} is even:

$$n_{batch} = 2 * n_c \tag{9}$$

where n_c is the number of couples in one batch. As the population is to be partitioned into batches, the following equality must stand

$$N = n_{batch}.M \tag{10}$$

Let us define $q = n_{batch} * p_{sim}$, the number of candidates to be simulated in parallel. To reach the optimal use of the computational resource, the following property must stand

$$\exists m \in \mathbb{N}\backslash\{0\} \text{ such that } n_{cores} = m.q \tag{11}$$

The parameter $p_{sim} \in [0, 1]$ has to be fixed such that:

$$n_{batch}.p_{sim} \in \mathbb{N} \tag{12}$$

Setting p_{sim} to 1 corresponds to running the GA without surrogate.

3.2. Surrogate Building using MCDropout-based BNN

The surrogate chosen here is an Artificial Neural Network (ANN) trained with Dropout and that employs the MCDropout technique to provide both the prediction and the uncertainty measure around it. ANNs are often selected as surrogate for their approximation universality [42] and ease of update through incremental learning [43]. It is proven in [21] that training an ANN with Dropout and predicting from it with MCDropout amounts to training a Bayesian Neural Network. BNN learning consists in determining the probability distribution $p(\boldsymbol{W}|\boldsymbol{X_{train}}, \boldsymbol{y_{train}})$ over the weights and not directly their value as it is the case for traditional ANNs [12]. The main benefit of BNNs is the possibility to access uncertainty information about the prediction but the main drawback is the expensive computational cost of learning [44].

In [45], it is proposed to rely on an ANN with a Bayesian layer as output layer. This procedure consists in adaptive basis regression where the basis functions are the outputs of the last hidden layer. Since marginalization is realized by considering only the output weights of the net, training is faster than of traditional BNN. However, the training cost is still cubic in the number of basis functions and full incremental update is impossible because of the inversion of the kernel matrix.

The MCDropout technique is a computationally more efficient way of performing BNN learning [22] and retains the full incremental capability of ANNs.

The training method used in Parallel Batched BNN-assisted GA is described in Algorithm 5. At training time, Dropout is a regularization technique that drives each neuron to perform well independently from the remaining ones [46]. Let

$$\boldsymbol{W_2}.\boldsymbol{h}(\boldsymbol{W_1}.\boldsymbol{x}) \tag{13}$$

be the prediction for input $\boldsymbol{x}$ from a one-hidden-layer ANN with activation function $\boldsymbol{h}()$ and matrices of weights $\boldsymbol{W_1}, \boldsymbol{W_2}$. Training with Dropout consists in applying a Stochastic Gradient Descent algorithm [47] on the following network

$$diag(\boldsymbol{\epsilon_2}).\boldsymbol{W_2}.\boldsymbol{h}(diag(\boldsymbol{\epsilon_1}).\boldsymbol{W_1}.\boldsymbol{x}) \tag{14}$$

Algorithm 4 Framework of Parallel Batched BNN-assisted GA.

Input
simulator: real fitness function
N: population size
budget: budget for the search
M: number of batches per generation
n_{batch}: number of solutions per batch
p_{sim}: proportion of simulations per batch

database ← random_sampling(*simulator*, N)
surrogate ← training(*database*)
$\mathcal{P}$ ← *database* ▷ population
y_{min} ← get_best_cost(*database*)
while *budget*$\neq 0$ **do**
  $\mathcal{P}_p$ ← tournament(2, $\mathcal{P}$) ▷ population of parents
  $(\mathcal{B}_{p1}, \ldots, \mathcal{B}_{pM})$ ← partition($\mathcal{P}_p$, n_{batch}) ▷ batches of parents
  $\mathcal{P}_c \leftarrow \varnothing$ ▷ population of children
  for $0 \leqslant i \leqslant M$ & *budget* $\neq 0$ **do**
    $\mathcal{B}_{c,sim} \leftarrow \varnothing$ ▷ batch of children to simulate
    $\mathcal{B}_{c,pred} \leftarrow \varnothing$ ▷ batch of children to predict
    $(\mathcal{B}_{c,sim}, \mathcal{B}_{c,pred})$ ← acquisition_process($\mathcal{B}_{pi}$, *surrogate*, y_{min})
    $\mathcal{B}_{c,sim}$ ← parallel_simulation(*simulator*, $\mathcal{B}_{c,sim}$)
    $\mathcal{B}_{c,pred}$ ← prediction(*surrogate*, $\mathcal{B}_{c,pred}$)
    database ← *database* $\cup\, \mathcal{B}_{c,sim}$
    $\mathcal{P}_c \leftarrow \mathcal{P}_c \cup \mathcal{B}_{c,sim} \cup \mathcal{B}_{c,pred}$
    surrogate ← incremental_training(*database*,N,*surrogate*)
    $(\boldsymbol{x}_{min}, y_{min})$ ← get_best_cost(database)
  end for
  $\mathcal{P}$ ← elitist_replacement($\mathcal{P}$, $\mathcal{P}_c$, N)
end while
return $\boldsymbol{x}_{min}$, y_{min}

to minimize the error between the prediction and the target. $\boldsymbol{\epsilon_1}, \boldsymbol{\epsilon_2}$ are random vectors sampled, at each training iteration, from a Bernouilli law with probability p_{drop}. Let's assume $\epsilon_{1i} = 0$, then multiplying $\boldsymbol{W_1}$ by $\boldsymbol{\epsilon_1}$ zeros out the i-th row of $\boldsymbol{W_1}$. In other terms, the weights connected to the i-th neuron from the first layer are not updated during this training iteration. Dropout is represented by lines 4 to 7 in Algorithm 5. The training stops when the error between the predictions and the targets, computed over the training set has not decreased by at least δ during n_{ES} iterations. This process, called Early Stopping [48], corresponds to lines 8 to 10 in Algorithm 5.

Algorithm 5 Dropout training with one-hidden-layer ANN

Input

database: all simulated solutions
N: population size
p_{drop}: probability of dropping out neurons
SGD: Stochastic Gradient Descent algorithm
(δ, n_{ES}): Early Stopping parameters
$(\boldsymbol{W_1}, \boldsymbol{W_2})$: weights
$\boldsymbol{h}()$: activation function

1: $(\boldsymbol{X_{train}}, \boldsymbol{y_{train}}) \leftarrow$ get_last_samples(N, *database*)
2: *early_stopping* $\leftarrow$ *false*
3: **while** not *early_stopping* **do**
4: $\quad \boldsymbol{\epsilon_1}, \boldsymbol{\epsilon_2} \leftarrow$ Bernouilli_sampling(p_{drop})
5: $\quad \boldsymbol{preds} \leftarrow diag(\boldsymbol{\epsilon_2}).\boldsymbol{W_2}.\boldsymbol{h}(diag(\boldsymbol{\epsilon_1}).\boldsymbol{W_1}.\boldsymbol{X_{train}})$
6: $\quad MSE \leftarrow$ compute_MSE($\boldsymbol{preds}$, $\boldsymbol{y_{train}}$)
7: $\quad (diag(\boldsymbol{\epsilon_1}).\boldsymbol{W_1}, diag(\boldsymbol{\epsilon_2}).\boldsymbol{W_2}) \leftarrow$ backpropagation_update(*MSE*, *SGD*)
8: $\quad \boldsymbol{preds} \leftarrow \boldsymbol{W_2}.\boldsymbol{h}(\boldsymbol{W_1}.\boldsymbol{X_{train}})$
9: $\quad MSE \leftarrow$ compute_MSE($\boldsymbol{preds}$, $\boldsymbol{y_{train}}$)
10: $\quad$ *early_stopping* $\leftarrow$ update_early_stopping(*MSE*, δ, n_{ES})
11: **end while**
12: **return** $(\boldsymbol{W_1}, \boldsymbol{W_2})$

At prediction, MCDropout consists in sampling k sub-networks by dropping out neurons in the same way than Dropout training [21]. As described in Algorithm 6, the i-th sub-network predicts the output $\hat{y}_i$ from an input $\boldsymbol{x}$. Finally, the mean prediction $\hat{y}(\boldsymbol{x})$ and the approximated variance $\hat{s}(\boldsymbol{x})$ reflecting the BNN uncertainty are computed. Since only a limited number k of sub-networks are considered, the uncertainty measure is an approximated variance.

Algorithm 6 MCDropout prediction with one-hidden-layer ANN

Input

$\boldsymbol{x}$: input
k: number of sub-networks
p_{drop}: probability of dropping out neurons
$(\boldsymbol{W_1}, \boldsymbol{W_2})$: weights trained with Dropout
$\boldsymbol{h}()$: activation function

1: **for** $1 \leqslant i \leqslant k; i++$ **do**
2: $\quad \boldsymbol{\epsilon_1}, \boldsymbol{\epsilon_2} \leftarrow$ Bernouilli_sampling(p_{drop})
3: $\quad \hat{y}_i \leftarrow diag(\boldsymbol{\epsilon_2}).\boldsymbol{W_2}.\boldsymbol{h}(diag(\boldsymbol{\epsilon_1}).\boldsymbol{W_1}.\boldsymbol{x})$
4: **end for**
5: $\hat{y} \leftarrow \frac{1}{k}\sum_{i=1}^{k} \hat{y}_i$
6: $\hat{s} \leftarrow \frac{1}{k}\sum_{i=1}^{k} (\hat{y}_i - \hat{y})^2$
7: **return** $(\hat{y}, \hat{s})$

The incremental learning capability of the ANN grants keeping the information acquired at the beginning of the search and enhances the surrogate quality over the new regions of interest [49].

Incremental training appears in Algorithms 4 and 5 when passing the last known surrogate (or the known weight matrices $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$) as input to the training method. Training time increases when the number of neurons, the number of training inputs and/or the dimension of the inputs increases [48]. In the applications tackled in this paper, these quantities are relatively low and the training time is thus negligible compared to simulation.

3.3. Acquisition Process using EI-like Evolution Control

Once the surrogate trained, the Acquisition Process, described in Algorithm 7 begins. From a batch of parent solutions $\mathcal{B}_p$ composed of $n_c = n_{batch}/2$ couples of parents, a batch of n_{batch} child solutions $\mathcal{B}_c$ is generated. Each couple of parents $(\boldsymbol{p}, \boldsymbol{p'})$ generates by crossover and mutation two children $(\boldsymbol{c}, \boldsymbol{c'})$. The batch of children is then split into a batch of children to simulate $\mathcal{B}_{c,sim}$ and a batch of children to predict $\mathcal{B}_{c,pred}$. The $q = n_{batch} * p_{sim}$ children showing the greater Expected Improvement value, computed thanks to Formula 8, are assigned to $\mathcal{B}_{c,sim}$ and the remaining $n_{batch} * (1 - p_{sim})$ ones are assigned to $\mathcal{B}_{c,pred}$.

Algorithm 7 Acquisition Process in Parallel Batched BNN-assisted GA

Input
$\mathcal{B}_p$: current batch of parents
$(\hat{y}(), \hat{s}())$: surrogate prediction and variance functions
y_{min}: best simulated cost found so far
p_{sim}: proportion of simulations per batch

$n_c \leftarrow$ get_size$(\mathcal{B}_p)/2$
$\mathcal{B}_c \leftarrow \varnothing$ ▷ batch of children
for $1 \leqslant i \leqslant n_c; i++$ **do**
  $(\boldsymbol{p}, \boldsymbol{p'}) \leftarrow$ get_next_couple$(\mathcal{B}_p)$
  $(\boldsymbol{c}, \boldsymbol{c'}) \leftarrow$ crossover$(\boldsymbol{p}, \boldsymbol{p'})$
  $(\boldsymbol{c}, \boldsymbol{c'}) \leftarrow$ mutation$(\boldsymbol{c}, \boldsymbol{c'})$
  $\mathcal{B}_c \leftarrow \mathcal{B}_c \cup (\boldsymbol{c}, \boldsymbol{c'})$
end for
$\mathcal{B}_{c,sim} \leftarrow \varnothing$ ▷ batch of children to simulate
$q = n_{batch} * p_{sim}$ ▷ number of children to simulate
$\mathcal{B}_{c,pred} \leftarrow \varnothing$ ▷ batch of children to predict
for $1 \leqslant i \leqslant q; i++$ **do**
  $\mathcal{B}_{c,sim} \leftarrow \mathcal{B}_{c,sim} \cup \text{argmax}_{\boldsymbol{x} \in \mathcal{B}_c}(\text{EI}(\boldsymbol{x}, y_{min}, \hat{y}(), \hat{s}()))$
  $\mathcal{B}_c \leftarrow \mathcal{B}_c \backslash \mathcal{B}_{c,sim}$
end for
$\mathcal{B}_{c,pred} \leftarrow \mathcal{B}_c$
return $(\mathcal{B}_{c,sim}, \mathcal{B}_{c,pred})$

4. Experimentation using Benchmarks

4.1. Benchmark problems

In this section, the robustness comparison of the two approaches is proposed over different search landscapes. The landscapes considered are produced by benchmark functions known to be hard to optimize. Since the real-world applications tackled in the next section consist of 6 decision variables, it is decided to keep the same number when treating the artificial problems.

The Schwefel function, defined in Equation 15, produces a multi-modal plane [50].

$$\begin{aligned} F(x_1, \ldots, x_6) =& 418.9828872724338 * 6 - \sum_{i=1}^{6} x_i \sin\left(\sqrt{|x_i|}\right) \\ & \text{for } x_i \in [-500, 500] \\ F(x_1^*, \ldots, x_6^*) =& F(420.9687, \ldots, 420.9687) = 0 \end{aligned} \tag{15}$$

The Rastrigin function, defined in Equation 16, produces a noisy well-like surface [50].

$$\begin{aligned} F(x_1, \ldots, x_6) =& 60 + \sum_{i=1}^{6} x_i^2 - 10\cos(2\pi x_i) \\ & \text{for } x_i \in [-5.12, 5.12] \\ F(x_1^*, \ldots, x_6^*) =& F(0, \ldots, 0) = 0 \end{aligned} \tag{16}$$

The Rosenbrock function, defined in Equation 17, produces a valley-like surface [50].

$$\begin{aligned} F(x_1, \ldots, x_6) =& \sum_{i=1}^{5} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \\ & \text{for } x_i \in [-5, 10] \\ F(x_1^*, \ldots, x_6^*) =& F(1, \ldots, 1) = 0 \end{aligned} \tag{17}$$

The choice of benchmark functions is made to be representative of known issues encountered in global optimization techniques. Both noisy and multi-modal landscapes are difficult to optimize because of the high number of local optima. For a noisy landscape as Rastrigin the local noise amplitude and frequency is uniform whereas it is not the case for the multi-modal Schwefel problem.

4.2. Protocol

The proposed protocol considers a search budget limited to 1024 real fitness evaluations and an initial database made of 128 solutions sampled randomly from the search space. As evaluating a benchmark function is not time demanding, the budget is expressed as a limited number of real fitness evaluations.

The Genetic Algorithm from the Parallel Batched BNN-assisted GA approach is implemented in Pagmo [50] and is configured as follows: population size $N = 128$, SBX crossover with probability 0.9 and distribution index 10, polynomial mutation with probability 0.1 and distribution index 50 and a tournament replacement of size 2. The previous parameters are set according to [16] and more details about GA operators are given in [2].

The MCDropout-based BNN hyper-parameters are fixed in accordance to a hyper-parameters calibration step. The grid-search considers 270 settings obtained by combination of $\{1, 2, 3\}$ hidden layers, $\{3, 6, 12\}$ neurons per layer, {relu, sigmoid} activation functions, learning rates of $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $p_{drop} \in \{0.1, 0.5, 0.9\}$. The training set is composed of 100 samples and the setting providing the best MSE computed over a validation set of 10000 samples is reported in Table 1. In addition to these parameters, the Early Stopping parameters are fixed to $\delta = 10^{-4}$ and $n_{ES} = 56$, the Stochastic Gradient Descent relies on Nesterov momentum of 0.1 and the training set is normalized to lie into $[0, 1]$.

hyper-parameter \ problem	Rastrigin	Rosenbrock	Schwefel
number of layers	3	3	1
number of neurons	12	12	6
activation function	relu	relu	sigmoid
learning rate	0.1	0.1	0.1
p_{drop}	0.1	0.1	0.5

Table 1: Best MCDropout-based BNN configurations on the benchmark problems according to a grid-search hyper-parameters calibration involving 270 settings.

The proportion of simulations per batch p_{sim} is fixed to 25%, determined from preliminary experiments. Thus, in order to respect the constraints presented in Equations 9, 10 and 12, n_{batch} is set to $\{4, 32, 64, 128\}$. In this configuration, the number of simulated solutions per batch is $q \in \{1, 8, 16, 32\}$.

The number of solutions generated per Acquisition Process is $q \in \{1, 8, 16, 32\}$ for Parallel Kriging-based q-EGO. The implementation of the method relies on the R packages DiceKriging and DiceOptim from Roustant *et al.* [31]. The Kriging model is built with a linear trend, and a Matern$\frac{5}{2}$ covariance kernel as advised in [31]. A nugget effect is added to the meta-model to avoid ill-conditioning of the Kriging matrix. The nugget is automatically estimated during the fitting process of the Kriging thanks to the DiceKriging package. The maximization of EI is done thanks to a GA equipped with a linear crossover and a random mutation operator, the population size is set to 150 and the number of generations is limited to 15. The parameters of the GA have been manually tuned through informal experiments to give similar results in terms of time and quality of the solution than the default optimizer from the DiceOptim package. As mentioned in Sub-section 2.3, the choice of the GA for maximizing the Acquisition Function is motivated by the integration of constraints in the real-world application treated in the following. Indeed, the GA allows to take into account the constraint into the Acquisition Process and to propose only admissible candidate, as for the BNN-GA approach.

4.3. Results

Some High Performance Computing facilities rental companies fix the price for their rental based on the energy consumed by their customers' computations. Limiting the search budget to a fixed number of simulations reproduce artificially this situation. Indeed, assuming that an idle computational core does not consume energy, executing two simulations sequentially on one core costs as much as executing two simulations in parallel on two cores. The results obtained with the protocol described in the previous sub-section may then help practitioners constrained by energy consumption limitations.

Schwefel

The results presented in Figure 4 indicate that all the Parallel BNN-assisted GA methods outperform all the Parallel Kriging-based q-EGO variants for a budget greater than or equal to 600 real evaluations. Reversely, all the KRG q-EGO methods outperform all the BNN-GA variants for a budget less than or equal to 300 real evaluations. Once 200 real evaluations have been performed, the best identified cost provided by the KRG q-EGO methods hardly improves. KRG q-EGO improves rapidly its best known cost but has difficulty in identifying new promising regions once a stagnation point has been reached. This stagnation point occurs approximately at 200 real evaluations in Figure 4. BNN-GA improves its best known cost slower than KRG q-EGO at the beginning of the search but demonstrates a better behaviour at the end of the search by still reaching improvements.

Under a limited number of real evaluations, the number of parallel simulations q only influences the model update frequency. Higher is q, lower is the frequency of update. The surrogate updates are graphically represented in Figure 4, 5 and 6 by the crosses and the dots. Analyzing the relationship

between the update frequency and the improvement of the search allows to extract insights about the approximation capability of the surrogates on the landscape at hand.

For KRG q-EGO, a moderate update frequency ($q = 16$) is enough for the surrogate to approximate the landscape at the beginning of the search. Higher frequencies are preferred at the end of the search as it seems harder to identify new regions of improvement. When analyzing Kriging model update frequency, the updates implied by the Kriging Believer are not taken into account. For BNN-GA, moderate update frequencies ($q \in \{8, 16\}$) provide the best results. Multi-modality of the landscape could explain these results. Indeed, if there are a lot of disconnected promising regions, the approximated cost space can be explored more intensively before updating the surrogate.

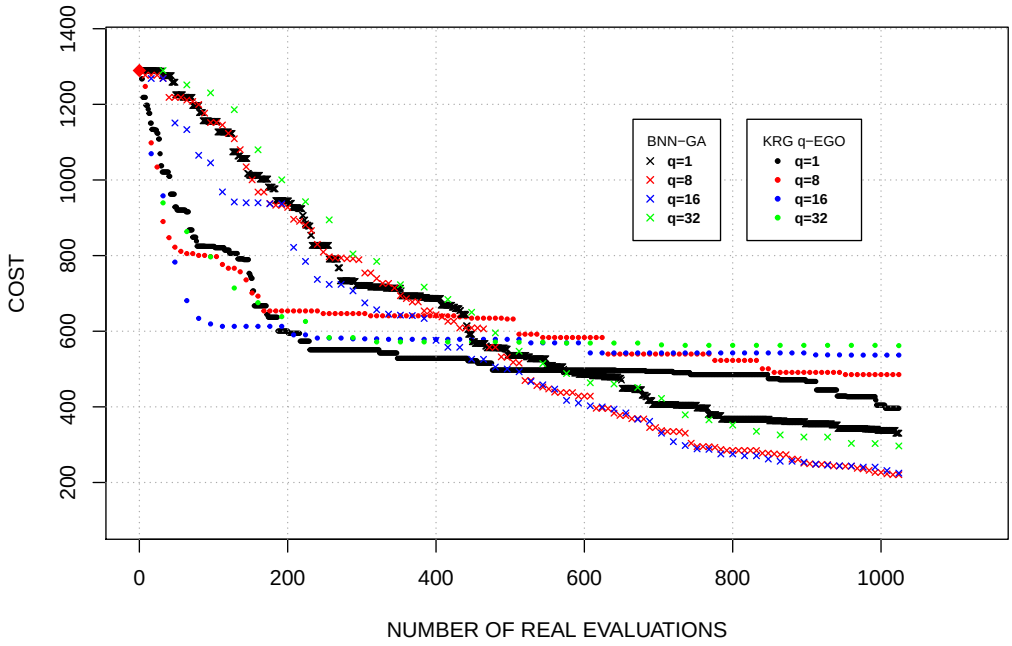

Figure 4: Mean of the best cost found according to the number of real evaluations performed for **Schwefel**. The mean is computed over 10 repetitions. The rhombus-shaped point represents the average of the initial best cost across the 10 initial samplings.

Rastrigin

The results obtained on the Rastrigin landscape are presented in Figure 5. The BNN-GA with $q = 1$ outperforms all other approaches for a budget greater or equal to 600 real evalutions. For a budget less than or equal to 300, all the KRG q-EGO methods outperform all the Parallel BNN-GA variants. The stagnation effect is also observed in this case with a stagnation point between 400 and 600 real evaluations for the KRG q-EGO variants. Observations about the improvement speed are the same as of the Schwefel case.

Regarding update frequency, it is better to consider high update frequency ($q = 1$) for BNN-GA. Moderate to high update frequencies ($q \in \{1, 8\}$) are preferred for KRG q-EGO particularly at the beginning of the search. Rastrigin landscape is formed by a general trend disturbed by homogeneous noise as demonstrates its 2-D representation in [50]. The few promising regions are rapidly identified and higher number of surrogate updates is required afterwards to further improve the quality of the search.

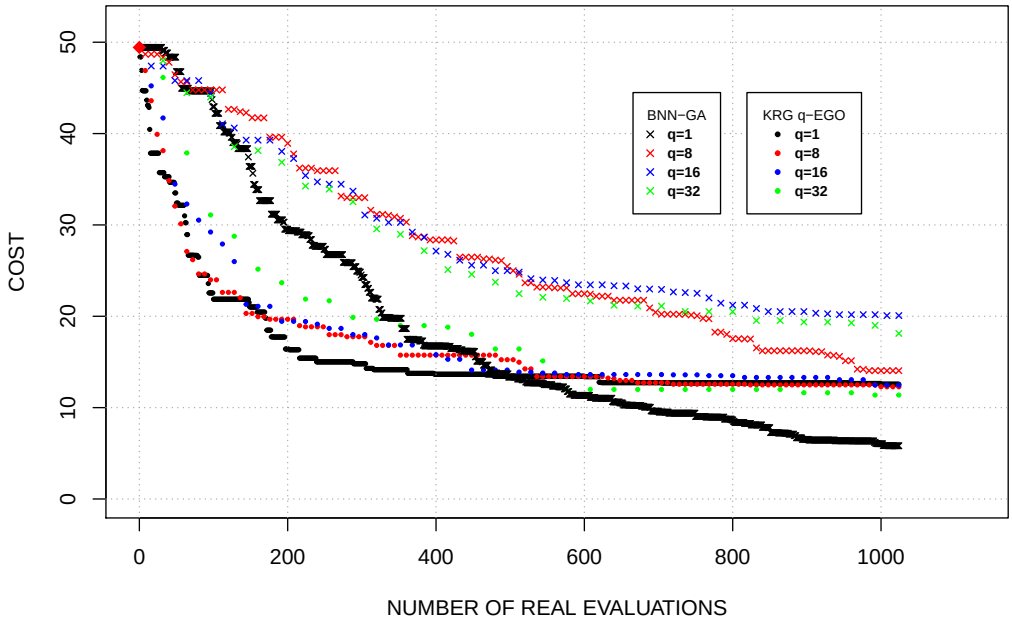

Figure 5: Mean of the best cost found according to the number of real evaluations performed for **Rastrigin**. The mean is computed over 10 repetitions. The rhombus-shaped point represents the average of the initial best cost across the 10 initial samplings.

Rosenbrock

All the variants of the KRG q-EGO outperform all the BNN-GA approaches when considering the Rosenbrock benchmark problem whose results are displayed in Figure 6. The stagnation effect for KRG q-EGO begins around 200 real evaluations and the observations about improvement speed are the same as of the two previous cases.

A high update frequency ($q = 1$) is shown to provide the best results for both approaches. The Rosenbrock landscape should exhibit few promising regions as suggests its 2-D representation in [50]. The observation regarding update frequency is then the same as of the Rastrigin case.

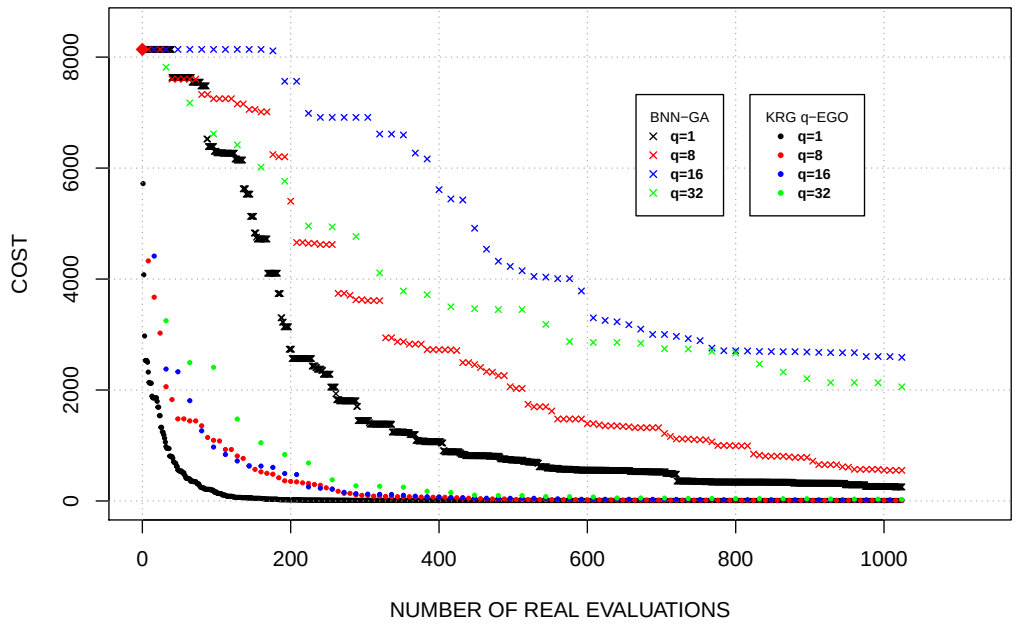

Figure 6: Mean of the best cost found according to the number of real evaluations performed for **Rosenbrock**. The mean is computed over 10 repetitions. The rhombus-shaped point represents the average of the initial best cost across the 10 initial samplings.

Conclusion

On the one hand, it has been observed on all the benchmark problems that KRG q-EGO improves rapidly its best identified cost before reaching a stagnation point at approximately 200 real evaluations. On the other hand, BNN-GA shows a slower improvement rate but a lower stagnation effect that allows it to outperform KRG q-EGO for higher budgets.

Regarding update frequency, the results exhibit a preference towards moderate update frequencies for multi-modal landscapes and high update frequencies for landscapes showing a low number of disconnected promising regions.

Global minimum is reached only for the Rosenbrock problem as it is the landscape with lowest roughness according to [50].

5. Application to TB Transmission Control

5.1. Problem description

Tuberculosis is an airborne disease that has been threatening mankind for thousands of years and still affects around 10 million individuals each year, killing around 1.7 million of them [51]. Previous works suggest that it will be impossible to reach the global elimination targets stated by WHO with the existing control tools [52]. In this context, main global health agencies and funders increasingly rely on mathematical modeling to design better tuberculosis control policies.

Mathematical models have the ability to simulate transmission within a population and to predict the impact of control interventions on future disease burden. In particular, the AuTuMN model is an ODE-based system designed to implement a mix of tuberculosis control programs and to estimate their effectiveness along with their programmatic costs [23]. In this report, we use two applications of the AuTuMN model to test the two optimization methods.

In a first exercise, the model is used to find the best strategy to allocate preventive treatments across different age groups given a limited number of treatments available each year. For this application we consider the Philippines, a high tuberculosis burden country with a disease prevalence of over 1% measured in 2016. The objective is to determine the allocation of preventive treatments that would minimize the estimated TB prevalence in 2035. Six age categories are considered and the optimization variables represent the number of treatments to be allocated to each of these sub-groups, considering a total number of preventive treatments of 600,000 per year from 2020 onwards. The optimization problem can be summarized as follows:

$$\min_{\mathbf{x}\in\mathbb{N}^6} f(\mathbf{x}), \tag{18}$$

$$\sum_{i=1}^{6} \mathbf{x}_i = 600000, \tag{19}$$

where f denotes the prevalence of tuberculosis in 2035 and $\mathbf{x}_i$ is the number of treatments allocated to the age category i every year.

Optimization of resource allocation is increasingly popular in the context of global health. Kelly *et al.* presented an estimation of how global HIV control resources could be redistributed between and within countries in order to maximize the epidemiological impact [53]. In [54] a similar analysis was performed, this time applied to Malaria disease in Nigeria. The complex optimization problems presented in these two studies were solved using a method based on adaptive stochastic descent [55].

The second problem tackled in this paper consists in calibrating a highly stratified version of the AuTuMN model. This model incorporates a high level of complexity in order to capture the important heterogeneity observed in tuberculosis epidemiology. That is, the simulated population can be stratified regarding age, risk factors (diabetes, HIV, smoking...), vaccination status, form of tuberculosis (smear-positive, smear-negative or extrapulmonary) and treatment history [25]. This complexity leads to long computational times which makes the optimization exercise laborious. Model calibration is performed by minimizing a sum-of-square distance between the model predictions and field observations for multiple disease indicators. The 6 calibrated parameters $\{p_i\}_{i\in\{1,...,6\}}$ represent the decision variables. If $SSD()$ denotes the calibration sum-of-square distance, the optimization problem is defined by:

$$\min_{p\in\mathbb{R}^6} SSD(p), \tag{20}$$

$$l_i \leq p_i \leq u_i, \forall i \in \{1, ..., 6\}, \tag{21}$$

where l_i and u_i are the lower and upper bounds associated with parameter p_i, respectively. This calibration problem is independent from the problem of preventive treatments allocation depicted above.

Calibration of models' parameters is a difficult optimization task encountered in the modelling of different phenomena. These problems are manageable by surrogate-assisted methods since they generally involve computationally expensive simulators or learning models tedious to train. In [28], parametrization of a predictive bending model is tackled. The design characteristics of the object to test and the characteristics of the load are given as inputs to the model and a value representing the deformation is provided as output. The objective function is the error observed between the model prediction and the physical measurements at hand. The watershed model examined in [37] is utilized worldwide for making practical water management decisions. Its parametrization consists in 15 decision variables and 2 or 3 objectives indicating the difference between simulation outputs and observed data. In [56], an ECG simulator is calibrated in order to produce realistic ECGs. Twenty one decision variables have to be optimally set to maximize the Pearson correlation coefficients between the measured and the simulated ECG signals at two different positions on the body surface.

5.2. Protocol

The computational resource at the disposal of the study is a cluster composed of 8 computational nodes, themselves composed of 2*16 AMD EPYC 7301 CPU cores. Both the parameter calibration

Problem	Initial database size	Budget	q for KRG q-EGO	$q = n_{batch} * p_{sim}$ for BNN-GA
TBTC-C	32	1 hour	1, 8, 16, 32	1, 2, 4, 8
TBTC-V	128	30 minutes	1, 8, 16, 32	1, 8, 16, 32

Table 2: Protocols features for the parameter calibration problem (TBTC-C) and the treatment distribution problem (TBTC-V).

problem, referred to as TBTC-C, and the treatment distribution problem, named TBTC-V, are tackled on this cluster. One evaluation of the TBTC-V model lasts 6 to 20 seconds on one core while one simulation with the TBTC-C simulator is about 3 to 4 minutes. The initial database size, the time budget for the search and the batch sizes are as presented in Table 2.

The population size of the Parallel Batched BNN-assisted GA is fixed to $N = 32$ for TBTC-C and $N = 128$ for TBTC-V. For TBTC-V, the crossover and mutation operators are designed specifically to take into account the constraint of the problem. From two parent solutions $(\boldsymbol{p_1}, \boldsymbol{p_2})$ a first child is generated by allocating the same number of treatments as $\boldsymbol{p_1}$ for categories sampled randomly. The remaining budget is distributed on the remaining categories according to the proportion of $\boldsymbol{p_2}$. The second child is generated in a similar way by reversing the roles of $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$. The mutation operator transfers some number of treatments from one category to another. All remaining GA parameters are fixed as presented in Sub-section 4.2.

The MCDropout-based BNN hyper-parameters are fixed arbitrarily to 2 hidden layers, 12 neurons per layer, relu activation function, a learning rate of 0.3 and $p_{drop} = 0.1$.

In order to respect the constraints presented in Equations 9 and 10 n_{batch} is set to $\{4, 8, 16, 32\}$ for TBTC-C and to $\{4, 32, 64, 128\}$ for TBTC-V. The proportion of simulations per batch p_{sim} is fixed to 25% for both problems. The third constraint presented in Equation 11 is not respected for the TBTC-V problem, which means that the number of idling cores is not optimal.

The hyper-parameters for the Parallel Kriging-based q-EGO approach are very similar to the one chosen for the benchmark problems, except for the crossover and mutation operators for the TBTC-V problem. Indeed, within the Acquisition Process of the method, the GA must take into account the linear constraint of the TBTC-V problem. Both operators are identical to the ones used in BNN-assisted GA. Dealing with the constraint directly through the GA in the Acquisition Process allows to propose only admissible candidates to the simulator.

5.3. Results

TBTC-C

The results of the experiments led on the TBTC-C problem are reported in Figure 7. The colorful curves of the same color indicate a same batch size ($n_{batch} \in \{8, 16, 32\}$). The batch size is the number of new candidates issued by the Acquisition Process. The number of parallel simulations per batch is $q = n_{batch}$ for KRG q-EGO and $q = n_{batch} * p_{sim}$ for BNN-GA.

It can be observed in Figure 7 that the number of simulations performed during the one-hour search increases along with q for both methods. On the one hand, the constraint of BNN-GA given in Equation 11 limits its highest number of parallel simulations to $q = 8$. On the other hand, KRG q-EGO reaches to occupy the full 32-cores computational node with $q = 32$. The corresponding curves displayed in green in Figure 7 demonstrate the superiority of KRG q-EGO in terms of scalability by reaching about 190 simulations in one hour while BNN-GA only performs 70 simulations. Relaxing the constraints of the BNN-GA method, for instance by increasing the population size as soon as more candidate points are identified, should improve considerably its scalabity.

A measure of scalability is introduced to compare both methods for the same value of q. The scaling factor η is defined as the ratio between the number of achieved simulations and the optimal number of simulations. The optimal number of simulations is set to be the number of simulations performed by the sequential algorithm ($q = 1$) multiplied by the number of cores. An optimal parallelization is characterized by a scaling factor equals to 1. The results presented in the left part of Table 3 show

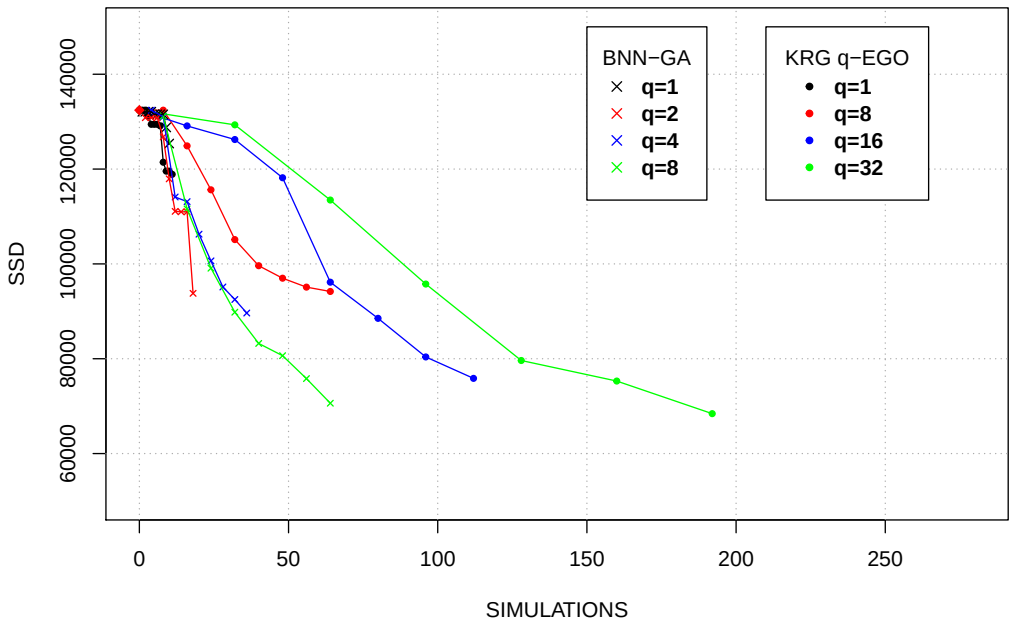

Figure 7: Mean of the best SSD value found according to the number of simulations performed for the **TBTC-C**. SSD value is the sum of square distance between AuTuMN model predictions and field observations. The mean is computing over 50 repetitions. The rhombus-shaped point represents the average of the initial best cost across the 50 initial samplings.

a similar scalability of both methods for $q = 8$. This conclusion is confirmed by the red dotted curve and the green crossed curve of Figure 7, both terminating at 70 simulations.

Table 3: Scaling factor for TBTC-C problem (on the left) and for the TBTC-V problem (on the right). A higher value indicates a higher scalability.

q	2	4	8	16	32
η_{KRG}	/	/	0.73	0.64	0.55
η_{BNN}	0.90	0.90	0.80	/	/

q	8	16	32
η_{KRG}	0.39	0.24	0.14
η_{BNN}	0.63	0.57	0.53

Figure 7 indicates that increasing the number of parallel simulations q causes an increase of the total number of simulations that in turns provides a great improvement of the SSD for both methods.

BNN-GA reaches equivalent SSD values than KRG q-EGO with a lower number of simulations. The green curves indicate that 190 simulations are required by KRG q-EGO while only 70 simulations are needed by BNN-GA to reach the same SSD of 70,000. A similar observation is obtained by focusing on the red curves: 70 simulations are required by KRG q-EGO when only 25 are needed by BNN-GA. For a fixed number of simulations, the results indicate that BNN-GA outperforms KRG q-EGO for all the parallel variants. Indeed, all the colorful crossed curves lay underneath all the colorful dotted curves. In particular, for $q = 8$, after 70 simulations BNN-GA reaches a SSD of 70,000 whereas KRG q-EGO reaches a SSD of 95,000.

For the benchmark problems tackled in Section 4, the KRG q-EGO used to outperform BNN-assisted GA especially at the beginning of the search. It is not the case for the TBTC-C problem. Hence, the search landscape of the TBTC-C problem should present a very different roughness than the benchmark landscapes. A landscape scattered with large plateaus regions may possibly explain these results.

Regarding the time budget, both methods achieve close results as demonstrates the last point of both green curves. When analyzing the time spent by the Acquisition Process, it is deduced that the Acquisition Process of BNN-GA is more time-efficient than in KRG q-EGO. In fact, for a given batch size, BNN-GA performs more Acquisition Processes than KRG q-EGO during the allocated time. This information is perceptible through the number of dots or crosses of the curves. For $n_{batch} = 32$, corresponding to green curves in Figure 7, BNN-GA realizes 8 Acquisition Processes while KRG q-EGO only realizes 6. Indeed, in this case, the Acquisition Process of KRG q-EGO involves 32 updates of the Kriging model and 32 maximization of EI. The high duration of the Acquisition Process for the q-EGO methods hampers its search capabilities.

TBTC-V

The results of the experiments led on the TBTC-V problem are reported in Figure 8. The curves show for both methods the increase of the number of simulations performed during the 30-minutes-search when q increases.

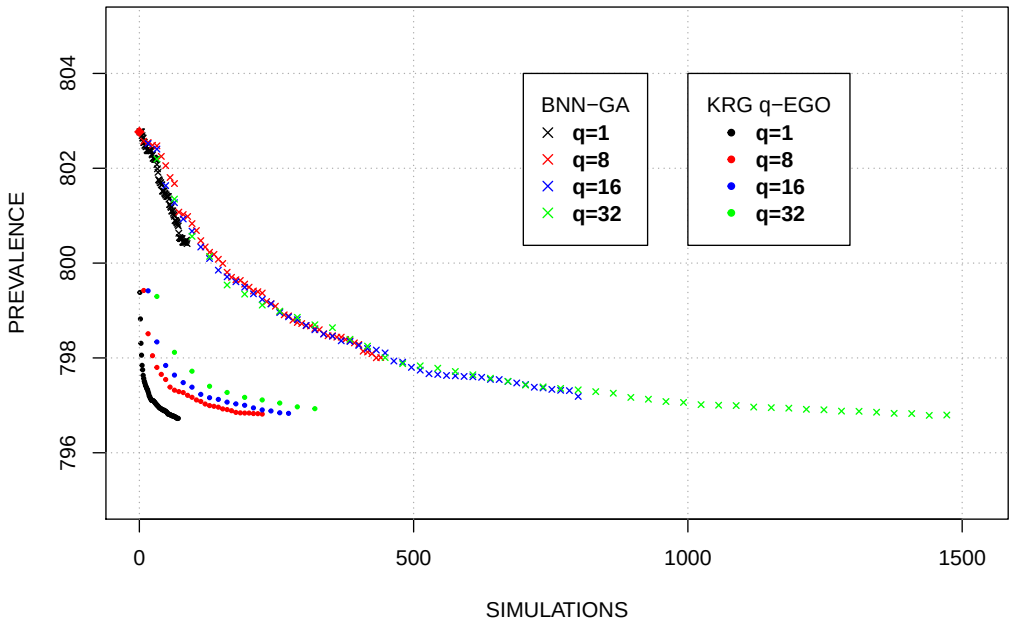

Figure 8: Mean of the best prevalence value found according to the number of simulations performed for the **TBTC-V**. The mean is computing over 50 repetitions. The rhombus-shaped point represents the average of the initial best cost across the 50 initial samplings.

A significant observation is that the constraint on the BNN-assisted GA does not restrain the number of parallel simulations as of the previous protocol. Since the initial database is bigger, BNN-GA reaches to fully use the available computing power by setting $q = 32$. The scaling factor presented in the right part of Table 3 shows higher scalability potential for BNN-GA than KRG q-EGO for all values of q. This analysis is confirmed by comparing the colorful curves of the same color. In particular, the green curves indicate that BNN-GA performs 1,500 simulations in 30 minutes while KRG q-EGO only performs 350 simulations. The poor scalability of KRG q-EGO is due to the long duration of its Acquisition Process as exhibited, for instance, by the low number of green dots compared to the

number of green crosses.

Figure 7 indicates that increasing the number of parallel simulations q causes an increase of the total number of simulations. The BNN-GA method clearly benefits from this increase as it causes the prevalence to decrease. Furthermore, since the curves presenting the results of BNN-GA are placed on top of each other in Figure 8, it indicates that raising the value of q does not deteriorate the quality of the search as it is for the competing approach. For the Kriging-based q-EGO method, this increase is not favorable, as the prevalence raises. Indeed, among the KRG q-EGO results, a clear preference is given to the classical EGO algorithm ($q = 1$). For $q \in \{8, 16, 32\}$ the Kriging Believer-based Acquisition Process implies to update the Kriging model with predicted solutions as explained in Algorithm 2. Possibly poorly predicted solutions — appearing because of a low surrogate update frequency on a landscape difficult to approximate — hamper the Acquisition Process to propose new promising candidates.

KRG q-EGO reaches equivalent prevalences than BNN-GA with a striking lower number of simulations. The green curves indicate that 1,500 simulations are required by BNN-GA while only 350 simulations are needed by KRG q-EGO to reach the same prevalence of 797. For a fixed number of simulations, the results indicate that KRG q-EGO outperforms BNN-GA for all the variants. Indeed, all the dotted curves lay underneath all the crossed curves. In particular, for $q = 1$, after 100 simulations KRG q-EGO reaches a prevalence of 797 whereas BNN-GA with $q = 1$ reaches a prevalence of 801. The observations made in Section 4 are recovered in the case of TBTC-V, indicating a search landscape similar with the benchmark landscapes tackled.

Regarding the time budget, BNN-GA with $q = 32$ and KRG q-EGO with $q = 1$ achieve close results. When analyzing the time spent in the Acquisition Process, it is deduced that the Acquisition Process of BNN-GA is more time-efficient than in KRG q-EGO as in the TBTC-C application. Besides, the Acquisition Process duration of KRG q-EGO increases as q increases. Indeed, the scaling factor η_{KRG}, presented in the right part of Table 3, decreases fast when q increases which means that the sequential part of the algorithm (*i.e.* the Acquisition Process) is prominent. On the contrary, η_{BNN} remains acceptable and shows that the sequential part of BNN-GA does not become prohibitive even when providing 128 new candidates as for $q = 32$.

It is to be noted that the prevalence reduction from 803 to 797 /100,000 population as observed with the BNN-assisted GA approach represents a significant improvement in the epidemiological situation of the Philippines. Indeed, given that the population of this country is over 100 million, the reduction induced by optimization is equivalent to more than 6,000 cases of tuberculosis prevented in 2035.

Conclusion

KRG q-EGO outperforms BNN-GA in terms of scalability on the TBTC-C problem because of the constraint imposed on q for BNN-GA. Nevertheless, for $q = 8$, both methods show the same performance. Increasing q for BNN-GA by increasing the population size should be considered when more simulated solutions become available. In the TBTC-V case, BNN-GA outperforms KRG q-EGO in terms of scalability for two reasons. On the one hand, the constraint on q for BNN-GA does not apply here since the initial population is large enough. On the other hand, the Acquisition Process of KRG q-EGO is too time-demanding. One explanation for this latter behavior is the ratio between the duration of the simulation and the duration of the Acquisition Process. Indeed for short-duration simulations like TBTC-V, the duration of the Acquisition Process highly exceeds the simulation duration. Therefore the algorithm spends most of its time to search for candidates, based on a meta-model that is not often updated. The Acquisition Process in KRG q-EGO is mainly suitable for more time-consuming simulations like in the TBTC-C case. These explanations are supported by the results of Table 3 where a decrease in the scaling factor is observed for KRG q-EGO from the TBTC-C to the TBTC-V.

For a budget expressed as a limited time in a computational node, both approaches produce similar results for both TBTC-C and TBTC-V. However, when the budget is expressed as a limited number of simulations, BNN-GA outperforms KRG q-EGO for the TBTC-C and KRG q-EGO outperforms BNN-GA for the TBTC-V. The roughness of the search landscape is most likely responsible for these

results.

6. Conclusions and Future Works

In this paper, we propose a new Evolution Control based on the Expected Improvement criterion within a Parallel Surrogate-Assisted Genetic Algorithm. To validate the approach, a comparison with a Parallel Kriging-based q-EGO method is driven on benchmark problems as well as on real-world problems related to Tuberculosis Transmission Control.

The MCDropout-based Bayesian Neural Network is chosen as surrogate model to integrate the Expected Improvement as Evolution Control in a Surrogate-Assisted Genetic Algorithm. The MCDropout-based BNN provides both the uncertainty information about predictions and an incremental cheap training. It takes benefit from both the advantages of Gaussian Processes and those of Artificial Neural Networks.

The Parallel Batched BNN-assisted GA outperforms the Parallel Kriging-based q-EGO for moderate budgets — expressed as a limited number of simulations — for the majority of the problems tackled in this paper. Even if EGO-like methods improve rapidly the objective value at the beginning of the search, a stagnation effect hinders further improvement. Besides, BNN-GA outperforms Kriging-based q-EGO even for very restricted budgets on one of the real-world problems.

Regarding scalability, Parallel Batched BNN-GA does not reach optimal use of available computational units when the initial database size is not large enough. Increasing the population size when more simulated individuals become available should bypass this restriction. When the simulation duration is not long enough, Parallel Batched KRG q-EGO does not benefit from an increase of the computational power.

The increasing computational capacity of energy-aware supercomputers from the TOP500 ranking [57] may allow to increase the budget allocated to optimization based on time-demanding simulations. In this context, the Parallel Batched BNN-GA is the most suitable method.

In order to prevent the stagnation effect affecting q-EGO approaches, a hybrid method will be investigated in our future works. The idea is to rely on a Kriging EGO-like Acquisition Process at the beginning of the search and to switch to a BNN-GA Acquisition Process once the stagnation point has been reached. For larger data bases and therefore a time-expensive Acquisition Process in the EGO-like approach, it is possible to alleviate the cost by splitting the search space. Doing so allows one to propose distinct candidate solutions in parallel.

Besides, other Evolution Controls should be investigated when considering BNN-GA. Actually, Expected Improvement has been designed specifically for EGO and surrogates that provide an exact variance around the predictions. However, the uncertainty information provided by MCDropout-based Bayesian Neural Network is an approximated variance.

Acknowledgment

We want to thank Dr. James Trauer, Prof. Emma McBryde and Dr. Tan Doan who developed the AuTuMN software in collaboration with Dr. Romain Ragonnet.

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] J. M. Trauer, J. T. Denholm, and E. S. McBryde. Construction of a mathematical model for tuberculosis transmission in highly endemic regions of the asia-pacific. *Journal of Theoretical Biology*, 358:74 – 84, 2014.

[2] E. G. Talbi. *Metaheuristics: From Design to Implementation.* Wiley Series on Parallel and Distributed Computing. Wiley, 2009.

[3] R. Bolze et al. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *The International Journal of High Performance Computing Applications*, 20(4):481–494, 2006.

[4] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 2018.

[5] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido. A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):104–116, Feb 2019.

[6] R. T. Haftka, D. Villanueva, and A. Chaudhuri. Parallel surrogate-assisted global optimization with expensive functions – a survey. *Structural and Multidisciplinary Optimization*, 54(1):3–13, Jul 2016.

[7] L. Shi and K. Rasheed. *A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms*, pages 3–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[8] A. Diaz-Manriquez, G. Toscano Pulido, J. Barron-Zambrano, and E. Tello-Leal. A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*, 2016:1–14, 06 2016.

[9] S. Rojas-Gonzalez and I. Van Nieuwenhuyse. A survey on kriging-based infill algorithms for multiobjective simulation optimization. *Computers & Operations Research*, 116:104869, 2020.

[10] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Fluids Engineering*, 86(1):97–106, 03 1964.

[11] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998.

[12] C. E. Rasmussen. *Gaussian processes for machine learning.* MIT Press, 2006.

[13] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104:148–175, 2016.

[14] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, Oct 2002.

[15] A. Díaz-Manríquez, G. Toscano, and C. A. Coello Coello. Comparison of metamodeling techniques in evolutionary algorithms. *Soft Computing*, 21(19):5647–5663, Oct 2017.

[16] K. Deb and P. Nain. An Evolutionary Multi-objective Adaptive Meta-modeling Procedure Using Artificial Neural Networks. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51, chapter 13, pages 297–322. Springer, Berlin, Heidelberg, 2007. doi: `http://dx.doi.org/10.1007/978-3-540-49774-5\_13`.

[17] C. Poloni, A. Giurgevich, L. Onesti, and V. Pediroda. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynam ics. *Computer Methods in Applied Mechanics and Engineering*, 186(2):403 – 420, 2000. doi: `https://doi.org/10.1016/S0045-7825(99)00394-1`.

[18] A. Syberfeldt, H. Grimm, A. Ng, and R. I. John. A parallel surrogate-assisted multi-objective evolutionary algorithm for computationally expensive optimization problems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3177–3184, June 2008. doi: `https://doi.org/10.1109/CEC.2008.4631228`.

[19] A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *International Journal of Computers, Systems and Signals*, 6:18–36, 01 2005.

[20] G. Vicario, G. Craparotta, and G. Pistone. Meta-models in computer experiments: Kriging versus artificial neural networks. *Quality and Reliability Engineering International*, 32(6):2055–2065, 2016.

[21] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[22] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[23] J. M. Trauer, R. Ragonnet, T.N. Doan, and E. S. McBryde. Modular programming for tuberculosis control, the "AuTuMN" platform. *BMC Infectious Diseases*, 17(1):546, Aug 2017.

[24] J. M. Trauer, J. T. Denhol, S. Waseem, R. Ragonnet, and E. S. McBryde. Scenario Analysis for Programmatic Tuberculosis Control in Western Province, Papua New Guinea. *American Journal of Epidemiology*, 183(12):1138–1148, 05 2016.

[25] R. Ragonnet, F. Underwood, T. Doan, E. Rafai, J. M. Trauer, and E. S. McBryde. Strategic planning for tuberculosis control in the republic of fiji. *Tropical Medicine and Infectious Disease*, 4(2), 2019.

[26] R. Ragonnet, J. M. Trauer, E. S. McBryde, R. M. G. J. Houben, J. T. Denholm, A. Handel, and T. Sumner. Is ipt more effective in high-burden settings? modelling the effect of tuberculosis incidence on ipt impact. *The international journal of tuberculosis and lung disease*, 21(1):60–66, 2017.

[27] D. Ginsbourger, R. Le Riche, and L. Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Technical report, March 2008.

[28] H. Wang, Y. Fan, E. Li, and Guangyao. A comparative study of expected improvement- assisted global optimization with different surrogates. *Engineering Optimization*, 01 2016.

[29] D. G. Krige. A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52(6):119–139, 1951.

[30] M. Georges. Principles of geostatistics. econ geol (lancaster). *Economic Geology*, 1963.

[31] O. Roustant, D. Ginsbourger, and Y. Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software, Articles*, 51(1), 2012.

[32] N. A. C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, 1993.

[33] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In Yoel Tenne and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, Springer series in Evolutionary Learning and Optimization, pages 131–162. springer, 2010.

[34] J. Liu, Z. H. Han, and W. Song. Comparison of infill sampling criteria in kriging-based aerodynamic optimization. *28th Congress of the International Council of the Aeronautical Sciences 2012, ICAS 2012*, 2:1625–1634, 01 2012.

[35] U. Noe and D. Husmeier. On a new improvement-based acquisition function for bayesian optimization. *ArXiv*, abs/1808.06918, 2018.

[36] P. Hennig and C. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13, 12 2011.

[37] T. Akhtar and C. A. Shoemaker. Efficient multi-objective optimization through population-based parallel surrogate search. *CoRR*, abs/1903.02167, 2019.

[38] N. Mori, M. Takeda, and K. Matsumoto. A comparison study between genetic algorithms and bayesian optimize algorithms by novel indices. pages 1485–1492, 01 2005.

[39] G. Briffoteaux, R. Ragonnet, M. Mezmaz, N. Melab, and D. Tuyttens. Evolution control for parallel ann-assisted simulation-based optimization, application to the tuberculosis transmission control. *Future Generation Computer System*, 2019.

[40] E. Correa and R. Goodacre. A genetic algorithm-bayesian network approach for the analysis of metabolomics and spectroscopic data: application to the rapid identification of bacillus spores and classification of bacillus species. *BMC Bioinformatics*, 12(1):33, Jan 2011.

[41] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[42] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. doi: `https://doi.org/10.1016/0893-6080(89)90020-8`.

[43] R. Polikar, L. Udpa, S. Udpa, S. Member, and Vasant V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, 03 2002.

[44] A. Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

[45] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. *Statistics*, 02 2015.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[47] Sebastian Ruder. An overview of gradient descent optimization algorithms., 2016. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam.

[48] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[49] M. Andrade, E. Gasca, and E. Rendón. Implementation of incremental learning in artificial neural networks. In Christoph Benzmüller, Christine Lisetti, and Martin Theobald, editors, *GCAI 2017. 3rd Global Conference on Artificial Intelligence*, volume 50 of *EPiC Series in Computing*, pages 221–232. EasyChair, 2017.

[50] F. Biscani, D. Izzo, and M. Märtens. esa/pagmo2: pagmo 2.6, November 2017. `https://esa.github.io/pagmo2/docs/cpp/cpp_docs.html#implemented-problems`.

[51] Global tuberculosis report 2018. Technical Report WHO/CDS/TB/2018.20, World Health Organization, Geneva, 2018.

[52] R. M. G. J. Houben and al. Feasibility of achieving the 2025 WHO global tuberculosis targets in South Africa, China, and India: a combined analysis of 11 mathematical models. *Lancet Glob Health*, 4(11):e806–e815, 11 2016.

[53] S. L. Kelly, R. Martin-Hughes, R. Stuart, X. F. Yap, D. J. Kedziora, K. L. Grantham, S. A. Hussain, I. Reporter, A. J. Shattock, L. Grobicki, et al. The global optima hiv allocative efficiency model: targeting resources in efforts to end aids. *The Lancet HIV*, 5(4):e190–e198, 2018.

[54] N. Scott, S. A. Hussain, R. Martin-Hughes, F. J. I. Fowkes, C. C. Kerr, R. Pearson, D. J. Kedziora, M. Killedar, R. M. Stuart, and D. P. Wilson. Maximizing the impact of malaria funding through allocative efficiency: using the right interventions in the right locations. *Malaria journal*, 16(1):368, 2017.

[55] C. C. Kerr, S. Dura-Bernal, T. G. Smolinski, G. L. Chadderdon, and D. P. Wilson. Optimization by adaptive stochastic descent. *PloS one*, 13(3):e0192944, 2018.

[56] B. Filipič, M. Depolli, J. Zupančič, J. Gmys, M. Gobert, N. Melab, and D. Tuyttens. Ecg simulator tuning: A parallel multiobjective optimization approach. In *Proceedings OLA'2018 International Workshop on Optimization and Learning: Challenges and Applications*, pages 25–28, 02 2018.

[57] Top500 the list. `https://www.top500.org/`.