# Evolution Control for parallel ANN-assisted simulation-based optimization application to Tuberculosis Transmission Control

Guillaume Briffoteaux, Romain Ragonnet, Mohand Mezmaz, Nouredine Melab, Daniel Tuyttens

# Evolution Control for Parallel ANN-assisted Simulation-based Optimization Application to Tuberculosis Transmission Control

G. Briffoteaux[a,c], R. Ragonnet[b], M. Mezmaz[a], N. Melab[c], D. Tuyttens[a]

[a] *Mathematics and Operational Research Department, University of Mons, Belgium*
[b] *School of Public Health and Preventive Medicine, Monash University, Australia*
[c] *Univ. Lille., CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France*

**Abstract**

In many optimal design searches, the function to optimise is a simulator that is computationally expensive. While current High Performance Computing (HPC) methods are not able to solve such problems efficiently, parallelism can be coupled with approximate models (surrogates or meta-models) that imitate the simulator in timely fashion to achieve better results. This combined approach reduces the number of simulations thanks to surrogate use whereas the remaining evaluations are handled by supercomputers. While the surrogates' ability to limit computational times is very attractive, integrating them into the overarching optimization process can be challenging. Indeed, it is critical to address the major trade-off between the quality (precision) and the efficiency (execution time) of the resolution. In this article, we investigate Evolution Controls (ECs) which are strategies that define the alternation between the simulator and the surrogate within the optimization process. We propose a new EC based on the prediction uncertainty obtained from Monte Carlo Dropout (MCDropout), a technique originally dedicated to quantifying uncertainty in deep learning. Investigations of such uncertainty-aware ECs remain uncommon in surrogate-assisted evolutionary optimization. In addition, we use parallel computing in a complementary way to address the high computational burden. Our new strategy is implemented in the context of a pioneering application to Tuberculosis Transmission Control. The reported results show that the MCDropout-based EC coupled with massively parallel computing outperforms strategies previously proposed in the field of surrogate-assisted optimization.

*Keywords: Surrogate-assisted Optimization, Simulation, Evolution Control, Artificial Neural Network, Massively Parallel Computing.*

*Email addresses:* guillaume.briffoteaux@umons.ac.be (G. Briffoteaux), romain.ragonnet@monash.edu (R. Ragonnet), mohand.mezmaz@umons.ac.be (M. Mezmaz), nouredine.melab@univ-lille.fr (N. Melab), daniel.tuyttens@umons.ac.be (D. Tuyttens)

## 1. Introduction

Many engineering problems consist of finding optimal designs. These problems are formalized as optimization problems where the function to optimize (fitness function) is generally a computationally expensive black-box simulator. In [1], the real-world problem treated consists in finding the intervention plan to purify contaminated groundwater at minimum cost. The black-box simulator takes up to 4 minutes to solve the system of nonlinear partial differential equations representing the fluid dynamic phenomena involved. In [2], the authors attempt to optimize the aerodynamic properties of an airfoil based on a 3D flow simulator. In [3], optimization of a factory production planning is achieved by maximizing devices utilization while minimizing tardiness, using a fitness function that relies on a software simulating the factory production. In [4], simulation times of 6 hours are reported when optimizing the shape of helicopter rotor blades to minimize vibrations.

Evolutionary Algorithms (EAs) such as Genetic Algorithms (GAs) are well suited to address such problems based on black-box simulators, as they do not require the design space to satisfy any property. However, EAs need a large number of fitness function evaluations to provide high-quality results.

The main obstacle to evolutionary optimization based on a computationally expensive black-box simulator is execution time. We aim to improve the search quality under a limited computational time budget. Recent developments of hybrid supercomputers emphasized by the TOP500 ranking [5], show the emergence of supercomputers exceeding 100 petaflops and with them the expectation to tackle bigger and bigger computational tasks. The computationally expensive simulators could be parallelized to run on supercomputers. The systems captured by expensive simulators are often ruled by differential equations that may be solved using big matrix operations. Such tasks can be performed efficiently using parallel computing [6], including on Graphics Processing Units (GPUs) [7]. Another approach consists in parallelizing the EA [8]. In [9] several EAs are run in parallel on GPUs, each of them handling the exploration of a portion of the design space.

An alternative approach to address the issue of long computational times is the use of surrogate models [10] [11]. Surrogate models, also called metamodels or approximate models, aim to emulate an original model, making its evaluation faster but at the cost of a coarser accuracy. During the optimisation task, the original simulator is replaced with the surrogate at a specific frequency. The alternation between the surrogate and the simulator is controlled by a process called Evolution Control (EC). In this paper we propose two new ECs based on Artificial Neural Networks (ANNs) surrogate models and MCDropout [12], a deep learning technique producing uncertainty information about ANNs predictions. MCDropout consists in sampling multiple ANN sub-networks to allow the computation of the variance over multiple predictions.

The first new EC selects for simulation the candidate solutions with the

highest uncertainty around their ANN-predicted fitness. Incorporating the uncertainty information allows to improve the surrogate accuracy on dubious regions of the design space, reducing the risk for the search to be misled. With the second new EC, the candidate solutions are non-dominated sorted based on uncertainty maximization and predicted fitness minimization to determine which candidates will be simulated. Combining uncertainty and predicted fitness allows to balance surrogate improvement and search improvement.

Parallelization presents multiple advantages for surrogate-assisted simulation-based optimization. Combining several surrogates is interesting to implement space decomposition or to combine diverse surrogate features. The underlying computational and memory costs of an ensemble of surrogates can be borne by a supercomputer. Another example is the use of GPUs to speed up training of large deep learning models [13] working as surrogates. In this paper, the ECs are conceived to treat batches of candidate solutions to enable parallel simulations. At a crossroad between deep learning, optimization and time-intensive simulations, future hybrid exascale machines are expected to be the key component to solve big optimal design problems arising from many areas.

In this paper, we consider a real-world problem of Tuberculosis (TB) Transmission Control (TBTC). TB is one of the most lethal infectious diseases with 10 million cases of active TB recognized in 2017 in the world which resulted in 1.6 million deaths [14]. In this context, the World Health Organization (WHO) proposed to end the global TB epidemic [15], with targets to reduce TB deaths by 95% and to cut new cases by 90% between 2015 and 2035.

The AuTuMN simulator [16] implements a TB transmission dynamic model and facilitates predictions of future TB epidemic trajectories across diverse scenarios. Besides, the coupling with an economic model allows to compare different control policies, which then helps decision makers in their effort to decrease local TB burden. Indeed, under a given scenario, the simulator can be optimized to produce efficient intervention plans. This problem falls consequently into the category of simulation-based optimal design problems.

The main contributions of this paper are the following:

- Thanks to MCDropout, we integrate the ANN prediction uncertainty into new ECs for surrogate-assisted evolutionary optimization to tackle computationally expensive simulation-based optimization problems. The MCDropout user-dependent parameter is studied on five benchmark problems and the method is compared to several ECs previously proposed in the field.
- A new EC is derived from the non-dominated sorting of two criteria: estimated fitness and uncertainty.
- We study the parallelization of the ECs on two supercomputers through batch-simulation of multiple candidate solutions.

- Finally, we tackle the TBTC problem in the Republic of Fiji.

The remainder of this paper is composed of six sections. In Section 2, related works are presented. In Section 3, the first new batch MCDropout-based EC is presented. Section 4 introduces the experimental protocol and the algorithm configurations. In Section 5, we present results obtained by tackling artificial benchmark problems. In Section 6, the new batch EC based on non-dominated sorting according to uncertainty maximization and prediction minimization is presented. Both new ECs are applied to the real-world problem of TBTC in the Republic of Fiji. Finally we present the study's conclusions in Section 7.

## 2. Related works

The articles cited hereafter present interesting methods related to the selection of candidate solutions to simulate and the parallelization strategy [17].

In [18], Sobester *et al.* propose a parallel strategy based on Radial Basis Functions (RBFs) surrogates. The candidate solutions producing the best expected improvement according to the RBF-predicted fitness are selected and simulated in parallel. Optimization of a spoked structure subjected to a load is treated in order to reduce the Von Mises stress. Six design variables representing the geometry of the structure are involved.

In [19], Akhtar & Shoemaker suggest five rules to perform the selection of candidate solutions to be simulated in parallel. Each rule is based on RBF predictions and brings its own degree of exploitation and exploration of the design space. Their method is applied to a ground-water remediation problem consisting in minimizing the contaminant concentration and the cost. The search is carried out by an EA and is limited to 500 simulations.

A similar ground-water remediation problem of 12 design variables is tackled in [20]. The RBF-predicted fitness and the distance to already simulated solutions are sequentially optimized to produce a set of candidate solutions that are simulated in parallel.

Within an EA, in [21], multiple surrogate-assisted local searches are carried on in parallel in order to propose multiple candidate solutions for simulation. The starting points of the local searches are the simulated solutions from the current EA population. RBF surrogates are built using the nearest neighbors of the starting solutions. The aerodynamic wing design problem considered is characterized by 11 design variables and 4 non-linear constraints. The execution time of the simulator used to compute the drag (quantity to minimize) is 11 minutes.

In [22], two criteria are used to select candidate solutions for simulation. First, the local expected improvement is maximized to produce candidate solutions with a promising predicted fitness. At the same time, the global expected improvement is maximized to yield candidate solutions with a high surrogate uncertainty. Even though parallelization of simulations is straightforward in

this study, it is not applied to the optimization of packing profile treated by the proposed method.

In [1], a combination of different surrogates is proposed to predict candidate solutions randomly sampled from the design space. The best predicted candidate solution is simulated and compared to the best simulated solution found so far. The approach is validated by solving a 12-dimensional ground water remediation problem. The simulator takes up to 4 minutes to solve a system of nonlinear partial differential equations reflecting both the water flows and the chemical reactions.

Zhan *et al.* propose in [23] a batch version of the Efficient Global Optimization (EGO) method [24] to allow parallel simulations. The first candidate solution selected for simulation is determined by optimizing the expected improvement based on Kriging predictions. Without simulation nor Kriging update, the next candidate solutions are obtained by optimizing a new acquisition function called pseudo-expected improvement. The candidate solutions are then simulated in parallel and the Kriging surrogate is updated. The approach is tested on artificial benchmark problems with up to 6 design variables.

In [4], the surrogate is first built on historical data and then used to replace the simulator completely in the optimization process. Only the best predicted candidate solution found by the end of the surrogate-based search is simulated. The authors treat a real-world problem optimizing helicopter rotor blade shapes to reduce vibration in flight conditions. The rotor blades are represented by 17 design variables and the reported simulation time is 6 hours.

In the study presented in [25], the authors approximate both the fitness function and the constraints of the optimization problem. The next candidate solution to simulate is chosen by considering the predicted fitness, the predicted constraints and the distance from already simulated solutions thus enhancing exploitation, feasibility and exploration, respectively. An automotive problem of minimizing the mass of a vehicle modeled by 124 design variables and subject to 68 performance constraints is addressed.

The coupling of nature-inspired algorithms as GA and ANN has been extensively investigated in the field of surrogate-assisted evolutionary optimization.

An ANN-assisted GA is used in [3] to optimize a factory production planning by maximizing device utilization and minimizing tardiness. In a GA, the new candidate solutions (offsprings) are generated by reproducing already evaluated solutions (parents). Here, the ANN prediction error committed on the parents is incorporated into the ANN-predicted fitness of the offsprings. The best predicted candidate solutions are then simulated in parallel.

In [2], an ANN-assisted GA is set up to optimize the aerodynamic properties of an airfoil thanks to a 3D flow simulator. The optimization procedure only relies on the computationally expensive simulator until a sufficient amount of simulations is available to train the ANN. Then, the GA relies on the ANN only and the best predicted candidate solutions are simulated at the end of the generation.

A cooperation strategy between the multi-objective NSGA-II and an ANN

is proposed in [26]. The distribution of simulations and surrogate predictions during the search is fixed *a priori* by the user. The strategy is applied to the optimization of a cantilever plate where the authors aim to minimize the weight and the maximum deflection of the plate.

In [27], we study the EC named Hypersphere Confident Region (HCR) that is based on the distance to already simulated solutions. If a candidate solution is close enough to the surrogate training set, the surrogate prediction is trusted. Otherwise, the candidate solution is simulated. HCR appears to outperform the EC presented in [26] on the multi-modal benchmark problem ZDT4.

The next section focuses on the Evolution Controls taxonomy and the first new batch EC based on MCDropout is presented.

## 3. A new batch Adaptive Evolution Control based on MCDropout

### *3.1. Background on Fitness Replacement*

When using surrogate-assisted population-based EAs, incorporating the surrogate model into the EA is not straightforward. In particular, one has to design an integration strategy to determine whether to call the simulator or the surrogate at each step of the evolution.

L. Shi & K. Rasheed define two categories of integration in [28]: Direct Fitness Replacement (DFR) and Indirect Fitness Replacement (IFR). In the former, the surrogate replaces the original fitness function at the evaluation step of the EA. Predicted candidate solutions can thus be embedded into the EA population. In the latter, the surrogate intervenes at the initialization or at the reproduction step. When considering IFR, the EA population embeds exclusively simulated solutions.

A common issue with DFR and IFR is that the search may be misled or convergence may be reached prematurely because of an inaccurate surrogate, although this drawback may be of lower importance in IFR since the EA population is composed of simulated solutions only. Setting up an alternation between the surrogate and the simulator should allow to improve the surrogate and thus to circumvent these issues. This alternation is controlled by the Evolution Control (EC) [29] [28].

In DFR, three categories of ECs can be defined as illustrated in [29]: No Evolution Controls (NECs), Fixed Evolution Controls (FECs) and Adaptive Evolution Controls (AECs).

With the NEC approach, the EA only relies on the surrogate predictions [30] [31] [32] [33] [34] [35] [36]. The surrogate is built using a historical database of simulations and only the best predicted solution returned by the EA is simulated. The guarantees about surrogate accuracy should therefore be strong when opting for a NEC approach.

When there is no strong guarantee about surrogate accuracy, the surrogate should be updated during the search. The alternation between the original and

the approximated fitness functions during the search allows the surrogate to be updated and to adapt to the new regions of interest dynamically detected by the EA. When using FECs [26] [37] [38], the alternation is fixed before the EA execution while in AECs [39] [40] the alternation process is designed at running time using information acquired during the search such as surrogate prediction and prediction uncertainty. Accessing the uncertainty information is challenging in the field of surrogate-assisted optimization.

In IFR, the surrogate generally filters out non-promising candidate solutions at the reproduction step of the population-based EA. At the reproduction step, a reproduction operator is in charge of generating offspring solutions based on parent solutions. A reproduction operator relying on a surrogate is called Informed Operator. AECs defined in the DFR paradigm can be used to perform the filtering with Informed Operators.

In Section 6, all the DFR ECs and an Informed Operators are applied to the TBTC problem. The next subsection focuses on the design of the first new batch AEC based on MCDropout.

### *3.2. Batch Adaptive Evolution Controls*

In this study, we propose new batch AECs to tackle the TBTC problem. To conceive a batch AEC, one needs to choose the surrogate type (Artificial Neural Network, Gaussian Processes...) [41], the number of surrogates (single or ensemble) [1] [40], the surrogate confident region (local or global) [42], the surrogate update policy, the batch size and the decision mode.

First, the TBTC problem relies on a black-box simulator and no historical simulation database is available. Consequently an ANN is chosen as surrogate. ANNs are selected for their approximation universality [43] and the possibility for incremental updates. Gaussian Processes (GPs) as Kriging models or RBFs are commonly used as they provide uncertainty information about their predictions but their training cost is cubic to the number of training points [44]. An ANN with MCDropout retains the best of both ANNs and GPs. MCDropout is further explained in Subsection 3.3.

Second, we choose to rely on a unique ANN. This decision is motivated by the fact that an ANN is already an ensemble of neural sub-networks from the MCDropout's point of view.

Third, it does not seem relevant *a priori* to decompose the design space into multiple reduced sub-spaces to divide the search, since the surrogate confidence region is expected to cover the whole design space.

Fourth, it is sought to obtain a high-quality surrogate. The surrogate update policy includes the calibration of hyper-parameters and incremental updates. More details about the surrogate update policy are given in Subsection 4.3.

Fifth, the batch size $n_{batch} \in \mathbb{N}^*$ is the number of candidate solutions that are treated simultaneously by the AEC. When $n_{batch} = 1$, every candidate solution is compared to a user-calibrated threshold to decide whether to simulate

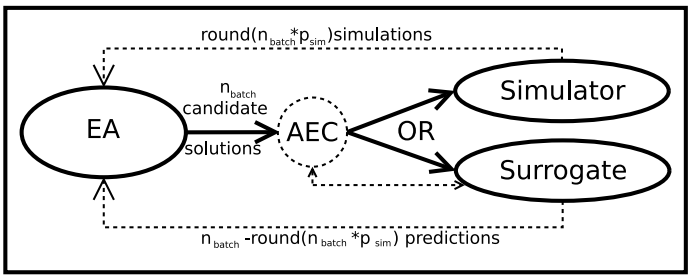


Figure 1: Batch Adaptive Evolution Control in surrogate-assisted Evolutionary Algorithm

it or not. Alternatively, when $n_{batch} > 1$ the $n_{batch}$ candidate solutions are compared with one another. A user-defined proportion $p_{sim} \in [0, 1]$ of the batch is simulated. Hence $round(p_{sim} * n_{batch})$ simulations can be performed in parallel. Different values of $n_{batch}$ and $p_{sim}$ offer different surrogate update frequencies. Figure 1 depicts the batch AEC in surrogate-assisted EA.

Finally, we need to specify the decision mode, which is the criteria used to select the candidates to simulate from the batch. Five decision modes are considered in this study. To define them, let

- $N \in \mathbb{N}^*$ be the population size.
- $f()$ be the function returning the fitness attached to a solution. This fitness could either be a simulated fitness or a predicted fitness.
- $\{\boldsymbol{p_1}, \ldots, \boldsymbol{p_N}\}$ be the population of parent solutions ordered such that $f(\boldsymbol{p_1}) \leqslant \cdots \leqslant f(\boldsymbol{p_N})$
- $\{\boldsymbol{c_1}, \ldots, \boldsymbol{c_N}\}$ be the population of candidate solutions such that $(\boldsymbol{c_{2i+1}}, \boldsymbol{c_{2i+2}})$ are fathered by $(\boldsymbol{p_{2i+1}}, \boldsymbol{p_{2i+2}})$ $\forall i \in \{0, \ldots, \frac{N}{2} - 1\}$.
- $M \in \mathbb{N}^*$ be the number of batches per generation. $n_{batch}$ must verify the following equality:
$$N = M.n_{batch} \tag{3.1}$$
- $round()$ be the round function returning the integral value that is nearest to the input, with halfway cases rounded away from zero.
- $n_{sim}$ be the number of simulated candidates per batch
$$n_{sim} = round(p_{sim} * n_{batch}) \tag{3.2}$$
- $\mathcal{B}_i$ be the $i$-th batch of solutions for a given generation ($i \in \{1, \ldots, M\}$).
- $\mathcal{S}_i$ be the subset of $\mathcal{B}_i$ made of candidate solutions to be simulated.
- $\mathcal{P}_i$ be the subset of $\mathcal{B}_i$ made of candidate solutions to be predicted.

Since all the batches are treated in the same way, let us focus on the first batch $\mathcal{B}_1 = \{\boldsymbol{c_1}, \ldots, \boldsymbol{c_{n_{batch}}}\}$ to simplify the notations.

The first decision mode, referred to as *mod*, simulates the first $n_{sim}$ candidates from the batch. Since the decision is made *a priori* the EC is a FEC. This FEC is included into the study for comparison purposes. For *mod*-based FECs,

$$\begin{aligned} \mathcal{S}_1 &= \{\boldsymbol{c_1}, \ldots, \boldsymbol{c_{n_{sim}}}\} \\ \mathcal{P}_1 &= \{\boldsymbol{c_{n_{sim}+1}}, \ldots, \boldsymbol{c_{n_{batch}}}\} \end{aligned} \tag{3.3}$$

The second decision mode, referred to as *rand*, simulates $n_{sim}$ candidates randomly sampled from the batch. This naive AEC is included into the study for comparison purposes. For *rand*-based AECs,

$$\begin{aligned} \mathcal{S}_1 &= \{\boldsymbol{c_{p(1)}}, \ldots, \boldsymbol{c_{p(n_{sim})}}\} \\ \mathcal{P}_1 &= \{\boldsymbol{c_{p(n_{sim}+1)}}, \ldots, \boldsymbol{c_{p(n_{batch})}}\} \end{aligned} \tag{3.4}$$

where $p()$ is a random permutation of $\{1, \ldots, n_{batch}\}$

The third decision mode, referred to as *hcr* [27], simulates the $n_{sim}$ candidates that are the farthest from the surrogate training set $\mathcal{T}$. For *hcr*-based AECs,

$$\begin{aligned} \mathcal{S}_1 &= \{\boldsymbol{c_{q(1)}}, \ldots, \boldsymbol{c_{q(n_{sim})}}\} \\ \mathcal{P}_1 &= \{\boldsymbol{c_{q(n_{sim}+1)}}, \ldots, \boldsymbol{c_{q(n_{batch})}}\} \end{aligned} \tag{3.5}$$

where $q()$ is the permutation that classifies the candidates in descending order according to the euclidean distance $d()$ to $\mathcal{T}$

$$d(\boldsymbol{c_{q(1)}}, \mathcal{T}) \geqslant \cdots \geqslant d(\boldsymbol{c_{q(n_{batch})}}, \mathcal{T}) \tag{3.6}$$

The fourth decision mode, referred to as *bp* [1], simulates the $n_{sim}$ candidates yielding the best predicted fitness. For *bp*-based AECs,

$$\begin{aligned} \mathcal{S}_1 &= \{\boldsymbol{c_{r(1)}}, \ldots, \boldsymbol{c_{r(n_{sim})}}\} \\ \mathcal{P}_1 &= \{\boldsymbol{c_{r(n_{sim}+1)}}, \ldots, \boldsymbol{c_{r(n_{batch})}}\} \end{aligned} \tag{3.7}$$

where $r()$ is the permutation that classifies the candidates in ascending order according to the predicted fitness $\hat{f}()$ (minimization is assumed)

$$\hat{f}(\boldsymbol{c_{r(1)}}) \leqslant \cdots \leqslant \hat{f}(\boldsymbol{c_{r(n_{batch})}}) \tag{3.8}$$

In this paper, we propose the decision mode that simulates the $n_{sim}$ candidates showing the highest prediction uncertainty obtained from MCDropout. This decision mode is referred to as *mcd* in the following. For *mcd*-based AECs,

$$\begin{aligned} \mathcal{S}_1 &= \{\boldsymbol{c_{s(1)}}, \ldots, \boldsymbol{c_{s(n_{sim})}}\} \\ \mathcal{P}_1 &= \{\boldsymbol{c_{s(n_{sim}+1)}}, \ldots, \boldsymbol{c_{s(n_{batch})}}\} \end{aligned} \tag{3.9}$$

where $s()$ is the permutation that classifies the candidates in descending order according to the prediction uncertainty $V()$

$$V(\boldsymbol{c_{s(1)}}) \geqslant \cdots \geqslant V(\boldsymbol{c_{s(n_{batch})}}) \tag{3.10}$$

In the next subsection, we provide details about the computation of the prediction uncertainty $V()$ thanks to MCDropout.

### *3.3. MCDropout prediction uncertainty*

Originally, Dropout is a regularization technique stemming from deep learning models and aiming at improving the generalization capacity of the model [45]. Ensembles of such models have demonstrated to perform well on diverse applications [46] [1]. Moreover, ensembles allow to retrieve a prediction uncertainty computed as the variance over the predictions produced by the members. Nevertheless, the cost of maintaining several surrogates is critical when a limited time budget is available. MCDropout offers a compromise by considering a single ANN as an ensemble of sub-networks obtained by randomly dropping out neurons [45]. Although MCDropout has already been used in a Bayesian Optimization approach [47], it has never been used in an Evolutionary Optimization approach and it has never been applied to a computationally expensive real-world problem.

During training, the stochastic gradient descent algorithm updates the weights connecting the neurons of the network in order to reduce the errors between the predictions and the targeted values. With Dropout, at each training iteration a random set of neurons is deactivated and the weights attached to these neurons are not updated. Each neuron is deactivated with a probability $p_{drop}$. Let us consider a neural network made of one hidden layer $\hat{\boldsymbol{f}} = \boldsymbol{W_2}.\boldsymbol{h}(\boldsymbol{W_1}.\boldsymbol{x} + \boldsymbol{b_1}) + \boldsymbol{b_2}$ where $\boldsymbol{W_1}, \boldsymbol{W_2}$ are the weight matrices, $\boldsymbol{b_1}, \boldsymbol{b_2}$ the bias vectors, $\boldsymbol{h}()$ the activation function and $\boldsymbol{x}$ a training input vector. Sampling vectors $\boldsymbol{\epsilon_i}$ from a Bernouilli distribution with probability $p_{drop}$ and multiplying $\boldsymbol{W_i}$ by $diag(\boldsymbol{\epsilon_i})$ randomly drops out some weights during the update step. This procedure forces each neuron to adapt to the training data individually and independently from the other neurons. Indeed, when trained all together neurons may co-adapt, as some of them compensate errors committed by others. Dropout prevents this co-adaptation. At inference time, the Monte-Carlo approach consists in predicting a candidate solution with each possible sub-network and to average the results. This method, referred to as MCDropout, has the benefit to provide a

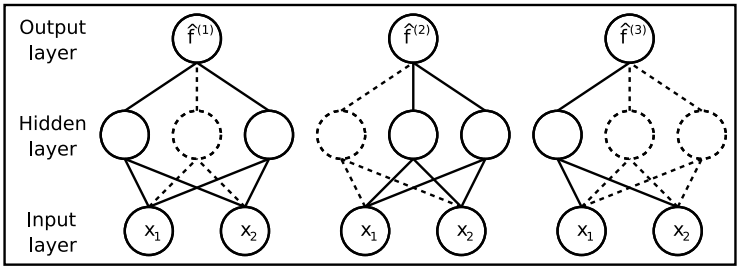


Figure 2: Prediction of a candidate solution $\boldsymbol{x} = (x_1, x_2)$ by $k = 3$ sub-networks allowing to compute the mean predicted cost $\overline{\hat{f}} = \frac{1}{k}\sum_{i=1}^{k} \hat{f}^{(i)}$ and the variance around it $V = \frac{1}{k}\sum_{i=1}^{k}(\hat{f}^{(i)} - \overline{\hat{f}})^2$.

variance around the mean, therefore quantifying uncertainty around the prediction.

Bayesian Neural Networks (BNNs) are machine learning models which provide confidence intervals around the prediction. Training a BNN consists in determining a probability distribution over the weights of the network and not the value of the weights directly. After assigning prior distributions to the weights $p(\boldsymbol{W})$, the Bayes theorem is used to compute the posterior distribution $p(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{W}$ is the weight matrix and $(\boldsymbol{X}, \boldsymbol{Y})$ is the training data set. The predictive distribution $p(\boldsymbol{y^*}|\boldsymbol{x^*}, \boldsymbol{X}, \boldsymbol{Y})$ for a new data point $\boldsymbol{x^*}$ is then obtained by inference. The practical difficulty of performing Bayesian inference has been partially alleviated by Y. Gal who proves theoretically in [12] that performing MCDropout amounts to achieve approximate Bayesian inference.

In the batch *mcd*-based AEC defined in the previous subsection, the prediction uncertainty $V()$ is an approximated variance. $V()$ is obtained by randomly sampling a limited number $k$ of sub-networks as depicted in Figure 2. The main advantage of this *mcd*-based AEC is to dynamically prevent the search from being misled by a poorly trained surrogate. However, *mcd*-based AEC may over-favor exploration when the surrogate is accurate enough. To remove this drawback, we propose in Section 6 another batch *mcd*-based AEC that combines both the prediction uncertainty and the predicted fitness.

In the next section, we present our experimental protocol and the algorithm configurations.

## 4. Experimental protocol and algorithms configurations

### *4.1. Batch Evolution Controls*

We consider 79 batch ECs in the experiments reported in Sections 5 and 6. Each EC is represented by a dot in Figure 3. The red dots represent NECs where the GA only relies on the surrogate model (*surrogate_only*) or only relies on the

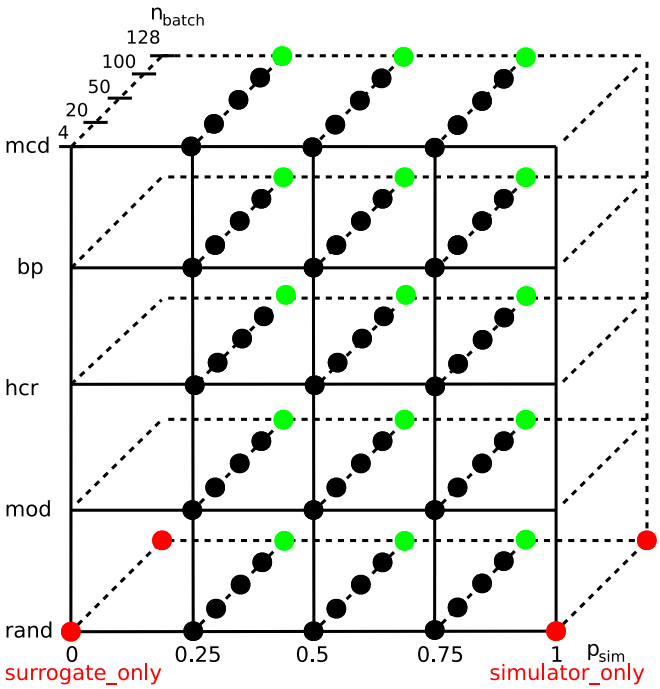


Figure 3: Batch-ECs obtained by combination of decision modes, proportions of simulations $p_{sim}$ and batch sizes $n_{batch}$. Red dots represent NEC baselines where only simulations (*simulator_only*) or only predictions (*surrogate_only*) are performed. Green dots as well as *surrogate_only* and *simulator_only* where $n_{batch} = 128$ represent ECs assuming optimal use of computational resource.

simulator (*simulator_only*). The remaining 75 batch ECs are batch FECs or batch AECs obtained by combining the decision modes (*rand*, *mod*, *hcr*, *bp*, *mcd*) with different proportions of simulations per batch ($p_{sim} \in \{0.25, 0.5, 0.75\}$) and different batch sizes ($n_{batch} \in \{4, 20, 50, 100, 128\}$).

### *4.2. Parallelism*

One experience consists in 79 searches (one per EC) considering a same initial GA population. The searches of the same experience are run sequentially on a same computational node made of multiple CPU cores. The batch simulations are distributed between the cores.

As the GA is a stochastic algorithm, we need to perform multiple repetitions of the experience in order to compare the ECs statistically. We execute 100 repetitions in parallel on different nodes of the supercomputer to obtain all the results presented in Sections 5 and 6.

Table 1: Number of idling Intel Xeon E5-2630 cores per batch in a node of 20 cores for $n_{batch} \in \{4, 20, 50, 100\}$.

| $p_{sim}$ \ $n_{batch}$ | 4 | 20 | 50 | 100 |
|---|---|---|---|---|
| 0.25 | 19 | 15 | 7 | 15 |
| 0.5 | 18 | 10 | 15 | 10 |
| 0.75 | 17 | 5 | 2 | 5 |

Let $n_{cores}$ be the number of available CPU cores in each computational node and $round()$ be the round function returning the integral value that is nearest to the input, with halfway cases rounded away from zero. In order to minimize computing units idling, $n_{batch}$ and $p_{sim}$ have to be fixed so that

$$\exists R \in \mathbb{N}^* \text{ such that } n_{cores} = R * round(p_{sim} * n_{batch}) \tag{4.1}$$

The first supercomputer exploited, proceeding from Grid5000 [48], is composed of 15 computational nodes of 20 Intel Xeon E5-2630 cores each. It is used to run the ECs where $n_{batch} \in \{4, 20, 50, 100\}$, depicted by black dots in Figure 3. In this configuration, resource idling is never optimally minimized as shown in Table 1. ECs minimizing the number of idling cores should *a priori* present better results. One simulation of the simulator involved in the TBTC problem lasts approximately 8 seconds on an Intel Xeon E5-2630 core while an ANN prediction time is around 0.01 second.

The second supercomputer employed is made of 8 computational nodes of 32 AMD EPYC 7301 cores each [48]. It is utilized to run the ECs where $n_{batch} = 128$, depicted by green dots in Figure 3. In this configuration resource idling is always optimally minimized. One simulation of the simulator involved in the TBTC problem lasts approximately 10 seconds on an AMD EPYC 7301 core while an ANN prediction time is around 0.02 second.

Only the simulations are parallelized since these are the tasks that are computationally expensive. The running times of GA operations and surrogate training can be neglected.

### *4.3. Surrogate training*

The surrogate update policy, named *update_effort* includes hyper-parameter calibration and incremental updates.

The calibration of the hyper-parameters is realized before the ANN initialization. 270 configurations are compared through a parallel grid-search considering $\{1, 2, 3\}$ hidden layers, $\{6, 12, 18\}$ neurons per hidden layer, {relu,sigmoid} activation functions, $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ learning rates and $\{0.1, 0.5, 0.9\}$ probability of Dropout $p_{drop}$. Training ends when the training Mean Squared Error (MSE) does not improve by at least $10^{-4}$ during 56 iterations. We select the configuration producing the best MSE computed over a validation set of 10000 points.

The incremental ANN updates occur at the end of each batch with a training set made of the last $N$ simulations, where $N$ is the GA population size. The training update is stopped by early stopping when the MSE computed on the training set does not improve by at least $10^{-4}$ during 56 iterations.

Even if efforts are focused on properly training the surrogate, high prediction accuracy is not totally guaranteed. Indeed, it could be tedious for the surrogate to follow the promising regions of the design space dynamically determined by the GA. Moreover, the hyper-parameter parallel grid-search does not offer guarantees of accuracy. For these reasons, and in view of testing the ECs in different conditions, we include an opposite surrogate update policy called *no_update*.

The second surrogate update policy, referred to as *no_update*, does not imply any specific effort to reach high quality ANN predictions. The ANN hyper-parameters are fixed arbitrarily to the following: 2 hidden layers formed of 20 sigmoid neurons each, 0.3 for the stochastic gradient descent learning rate, 0.1 for Nesterov momentum and 0.5 for Dropout probability $p_{drop}$. Training duration is controlled by early stopping which terminates the training process when the MSE computed on the training set does not improve by at least 0.1 during 8 iterations. More importantly, the surrogate is never updated during the search.

For both update policies, the initial training set is obtained from random sampling of $N$ solutions that are simulated in parallel. This sample corresponds to the initial GA population.

Dropout is enabled for training on hidden layers for all ECs requiring an ANN and MCDropout is enabled for prediction for batch *mcd*-based AECs only. A slight modification in the Keras implementation of ANN [49] is performed to allow for MCDropout at prediction.

As batch *mcd*-based AECs aim at preventing the search to be misled by inaccurate surrogate predictions, they are expected to produce interesting results under the *no_update* policy. Conversely, when the surrogate succeeds to follow the region of interest, batch *bp*-based AECs should perform best. Under the *no_update* policy, high values of $p_{sim}$ should be favored to prevent the search from being misled. Otherwise, lower values should be preferred to make more efficient use of the surrogate.

### *4.4. Genetic Algorithm*

The artificial benchmark problems are only tackled by ECs where $n_{batch} \in \{4, 20, 50, 100\}$ on the supercomputer equipped with 20 cores per computational node. The GA population size is set to $N = 100$ and the initial GA population is the initial ANN training set. The GA crossover operator is a simulated binary crossover with probability 0.9 and distribution index 10. The mutation operator is polynomial with probability 0.1 and distribution index 50. More details about GA operators can be found in [50].

The GA configuration used to treat the TBTC is detailed in Subsection 6.2.

For the benchmark problems as well as for the TBTC problem, the selection operator is a tournament selection of size 2 and the replacement operator is an elitist replacement. The GA is implemented in Pagmo [51].

The next section presents the experimental results obtained using the artificial benchmark functions.

## 5. Experimental results on benchmark functions

### *5.1. Benchmark functions*

Five artificial benchmark problems are considered to investigate the robustness of the ECs considering different search landscapes. The benchmark problems include six design variables as six interventions are considered in the TBTC problem. Hereafter, we present the fitness functions defining five box-constrained continuous single-objective problems.

- Ackley

$$F(x_1, \ldots, x_6) = 20 + e - 20 \exp\left(-\frac{1}{5\sqrt{6}}||x||_2\right) - \exp\left(\frac{1}{6}\sum_{i=1}^{6}\cos(2\pi x_i)\right)$$
$$\text{for } x_i \in [-15, 30] \tag{5.1}$$

- Griewank

$$F(x_1, \ldots, x_6) = \frac{1}{4000}\sum_{i=1}^{6} x_i^2 - \prod_{i=1}^{6}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$
$$\text{for } x_i \in [-600, 600] \tag{5.2}$$

- Rastrigin

$$F(x_1, \ldots, x_6) = 60 + \sum_{i=1}^{6} x_i^2 - 10\cos(2\pi x_i)$$
$$\text{for } x_i \in [-5.12, 5.12] \tag{5.3}$$

- Rosenbrock

$$F(x_1, \ldots, x_6) = \sum_{i=1}^{5} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$$
$$\text{for } x_i \in [-5, 10] \tag{5.4}$$

- Schwefel

$$F(x_1, \ldots, x_6) = 418.9828872724338 * 6 - \sum_{i=1}^{6} x_i \sin(\sqrt{|x_i|})$$
$$\text{for } x_i \in [-500, 500] \tag{5.5}$$

Table 2: Best mean fitness scores for the MCDropout parameter $k$. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{4, 20, 50, 100\}$ the $k$ value producing the best mean fitness (computed over 100 searches) is granted a point. Best scores appear in bold.

| $k$ \ problem | Ackley | Griewank | Rastrigin | Rosenbrock | Schwefel |
|---|---|---|---|---|---|
| 4 | **6** | **8** | **8** | **8** | **8** |
| 20 | **6** | 4 | 4 | 4 | 4 |

The 2D variants of the previous fitness functions can be visualized in [52]. The resulting search landscapes are either multi-modal, noisy or valley-shaped, making thus the optimization tedious. To reproduce a computationally expensive fitness function, the budget limiting the search is fixed to 200 fitness evaluations.

### *5.2. MCDropout parameter calibration*

The MCDropout parameter $k$, illustrated in Figure 2, is calibrated considering two values: 4 and 20. The ECs are compared based on the benchmark problems exhibited in the previous subsection. The considered ECs are characterized by $p_{sim} \in \{0.25, 0.5, 0.75\}$, $n_{batch} \in \{4, 20, 50, 100\}$ and the *no_update* policy. The search is limited to 200 fitness evaluations.

According to the best mean fitness scores presented in Table 2, it is enough to sample 4 sub-networks from the entire ANN to retrieve an informative prediction uncertainty. This value is used from now on in all batch *mcd*-based AECs.

### *5.3. Decision modes performances on different landscapes*

The decision modes presented in Subsection 3.2 are compared on the benchmark problems with a budget limited to 200 fitness evaluations and the *no_update* policy. The different values of the batch sizes and the proportions of simulations per batch yield different distributions of the original fitness evaluations during the search. Applying the *no_update* surrogate update policy allows one to evaluate the capability of the ECs to prevent the misleading of the search.

In order to identify the ECs in the following tables, the following naming convention is applied:

$$\text{decision-mode}\_p_{sim}\_n_{batch}$$

and the NECs are named *simulator_only* and *surrogate_only*.

In Table 3 and Table 4 are reported respectively the best mean fitness scores and the best overall ECs for each benchmark problem. According to the results, *mcd*-based AECs seem to be the best for 3 benchmark problems over 5. The ability of the *mcd* decision mode to prevent the search from being misled by an inaccurate surrogate is consequently validated. The performance of *hcr*-based AECs already investigated in [27] is confirmed. Nevertheless, further works should be conducted to explain the particularly great performance of the *hcr* decision mode on the Schwefel benchmark problem.

Table 3: Best mean fitness scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{4, 20, 50, 100\}$ the decision mode producing the best mean fitness better than NECs strategies (computed over 100 searches) is granted a point. Best scores appear in bold.

| problem / decision mode | Ackley | Griewank | Rastrigin | Rosenbrock | Schwefel |
|---|---|---|---|---|---|
| *rand* | 2 | 2 | 2 | 2 | 1 |
| *mod* | 3 | 1 | 2 | 2 | 0 |
| *hcr* | 2 | **7** | **4** | **4** | **11** |
| *bp* | 0 | 0 | 0 | 0 | 0 |
| *mcd* | **5** | 2 | **4** | **4** | 0 |

Table 4: Best ECs for each benchmark problem according to the mean fitness (computed over 100 searches).

| Ackley | Griewank | Rastrigin | Rosenbrock | Schwefel |
|---|---|---|---|---|
| *mcd*_0.25_20 | *mcd*_0.25_50 | *rand*_0.25_4 | *mcd*_0.25_20 | *hcr*_0.25_20 |

It can be seen from Table 4 that a low value of $p_{sim}$ is preferred. Actually, the budget is expressed in number of fitness evaluations so a low value of $p_{sim}$ concedes a higher number of predictions hence allowing the search to progress further. The surrogate imprecision, in this case, enhances exploration.

In the next section, we describe the TBTC problem and report the associated experimental results.

## 6. Application to Tuberculosis Transmission Control

### *6.1. Problem description*

Global health policy makers now rely extensively on mathematical predictions to design disease control guidelines and to plan budgets. In particular, mathematical models are frequently used to estimate the effects and the costs associated with disease control interventions [53, 54]. These models can also be used to optimize resource allocation between several candidate interventions and under financial constraints [55]. In such optimization problems, the budget allocation proportions of the total funding between the different programs are the optimization variables while a disease burden indicator is chosen as the objective to minimize.

Most infectious disease modeling studies employ a compartmental transmission model governed by ordinary differential equations (ODEs) [56]. Under this approach, the simulated population is stratified into several categories (termed compartments) according to their infection state, and often additional characteristics such as age-groups or factors associated with differential risks of disease.

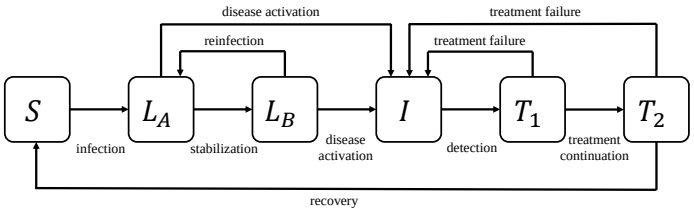


Figure 4: Simplified representation of the compartmental model used to simulate TB transmission. The compartments represent the different stages of infection. Susceptible individuals ($S$) may develop latent TB infection ($L_A$, $L_B$) through transmission. We included two sequential latent compartments ($L_A$, $L_B$) in our model to simulate the increased risk of progression to active disease in the months immediately following initial infection. That is, progression towards active TB disease ($I$) occurs at a higher rate early after infection (from $L_A$) than later after exposure (from $L_B$). Detected individuals receive treatment ($T_1$) and are assumed to become non-infectious soon after treatment initiation ($T_2$). Treatment may result in recovery (back to $S$) or failure (back to $I$). The model is further stratified by age (all compartments), diabetes status (all compartments), vaccination status ($S$), type of TB ($I$, $T_1$, $T_2$) and treatment history (all compartments).

Despite the relative simplicity and the deterministic nature of the ODE-based approach, disease models often require a high level of complexity to be able to produce realistic representations of the population and to simulate the disease dynamics accurately.

Such complexity of the disease models leads to extended computation times for the numerical solving of the ODE systems, making the optimization introduced above challenging, as it necessitates evaluating the disease simulator repeatedly. There is therefore a critical need for a reliable optimization solution that would significantly reduce computation time and thus increase the capacity of mathematical modeling to assist health policies efficiently.

In this paper, we consider the optimization of the strategic planning of tuberculosis control in the Republic of Fiji. The model structure presented in Figure 4 is used to simulate the local TB transmission dynamics. We implement the optimization of resource allocation between six control interventions considered by the Fiji TB program. Two interventions consist in increasing the provision of preventive treatment to contacts of TB patients (0-5 years old contacts in one case and >5 years old contacts in the other case). Another program under consideration is to offer enhanced medical support for patients under treatment in order to improve therapy outcomes. The national TB program also considered changing their detection approach through two additional interventions: sending van- and ferry-based screening units equipped with a more sensitive diagnostic tool (GeneXpert) to detect previously unrecognized cases [57]; and implementing decentralized care to improve access to diagnostic and treatment for the most remote communities. Finally, a program aiming to raise awareness about TB within the Fijian population was considered in order to improve health care seeking behavior. The aim of the optimization exercise is to identify

the amount of funding to be allocated to each of the six interventions described above that would yield the lowest level of TB prevalence in 2035, under the constraint that the combined spending should not be more than USD 1 million per year. The optimization problem can be formalized as follows:

$$\min_{\mathbf{x}\in\mathbb{N}^6} F(\mathbf{x}) \tag{6.1}$$

$$\sum_{i=1}^{6} \mathbf{x}_i = 1000000 \tag{6.2}$$

where $F(\boldsymbol{x})$ is the predicted TB prevalence in 2035 associated with a given budget allocation $\boldsymbol{x}$. In this application of AuTuMN to the TBTC, we use the model previously presented in Ragonnet *et al.* [58].

To the best of our knowledge, no existing works from surrogate-assisted optimization has been applied to epidemiological problems.

*6.2. GA initialization and reproduction*

To approach the TBTC optimization problem, the initialization of the GA population as well as the GA reproduction operators have to be defined to take the constraint into account.

An initial solution $\mathbf{x}$ is created by adding a random amount of money to the successive interventions $x_i$ until the financial budget is consumed. In this application, the population size is fixed to $N = 100$ for ECs where $n_{batch} \in \{4, 20, 50, 100\}$ and $N = 128$ where $n_{batch} = 128$. The initial population is also the ANN initial training set.

The crossover operator is illustrated by an example in Figure 5. Let $\mathbf{x}$ and $\mathbf{y}$ be the parent solutions and $I, J$ a random partition of $\{1, 2, 3, 4, 5, 6\}$ that represents the set of interventions. Let also $\mathbf{z}$ be the first offspring solution created based on $\mathbf{x}$ and $\mathbf{y}$. For the interventions in $I$, $\mathbf{z}$ receives the corresponding amount from $\mathbf{x}$ ($z_i = x_i$ for $i \in I$). The remaining budget at this step is $r = b_{fin} - \sum_{i\in I} x_i$. For the interventions in $J$, the remaining budget is shared out according to the proportion of the corresponding interventions in $\mathbf{y}$. In other terms, for $j \in J$, $z_j = \left\lfloor \frac{r * y_j}{\sum_{j\in J} y_j} \right\rfloor$. Finally, in case the remaining budget is not null, it is randomly added to an intervention from $J$. A second offspring solution is generated from the parent solutions $\mathbf{x}$ and $\mathbf{y}$ with a similar procedure where the roles of the parents are reversed. Crossover probability is set to 0.9.

The mutation operator applied with probability 0.1 randomly selects two different interventions from a same solution and transfers some budget from the one to the other.

*6.3. Performance evaluation considering the no_update policy: decision modes and parallelism*

For this experiment, the TBTC optimization problem is tackled with an execution time budget set to 1 minute for one search using 20 cores. The

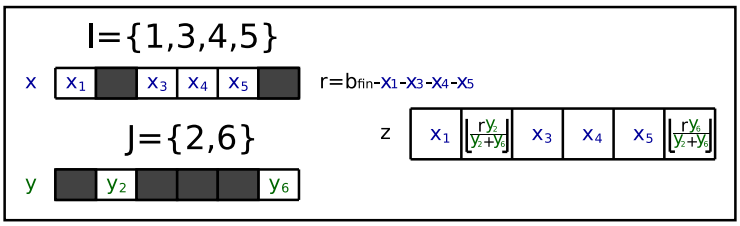


Figure 5: Example of crossover for the TBTC problem. Parents $\mathbf{x}$ and $\mathbf{y}$ father the offspring $\mathbf{z}$ respecting the amount of available financial budget $b_{fin}$.

Table 5: Case of resource idling and *no_update* policy. **Left:** best mean prevalence scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{4, 20, 50, 100\}$ the decision mode producing the best mean prevalence better than NEC strategies (computed over 100 searches) is granted a point. **Right:** Best ECs according to the mean prevalence.

| Decision mode | Best mean prevalence score |
|---|---|
| *rand* | 2 |
| *mod* | 2 |
| *hcr* | 1 |
| *bp* | 1 |
| *mcd* | **3** |

| EC | Mean prevalence |
|---|---|
| *mcd*_0.75_50 | **24.8889** |
| *mcd*_0.75_100 | 24.8983 |
| *mod*_0.75_50 | 24.9013 |
| *rand*_0.75_100 | 24.9033 |
| *hcr*_0.75_100 | 24.9100 |

different values of simulation proportions and batch sizes allow one to produce different levels of computational resource idling as shown in Table 1.

The *mcd* decision mode performs the best according to the results presented in Table 5. Besides, several FECs and AECs over-perform the *simulator_only* NEC even if this latter optimally uses the computational resources. This result proves the benefits of relying on surrogates.

It is shown through the right part of Table 5 that $(p_{sim}, n_{batch}) \in \{(0.75, 50), (0.75, 100)\}$ allow the production of the best mean prevalence values. Indeed, these pairs of values allow one to minimize the number of idling cores per batch as shown in Table 1. The pair of values $(0.75, 20)$ is less efficient since the total number of treated batches is greater, implying consequently a higher total number of idling cores during the search.

The TBTC is now tackled on the 32-cores-per-node supercomputer with population size and batch size fixed to 128 to ensure the optimal usage of the CPU cores. The results shown in Table 6 demonstrate the superiority of the *mcd*-based AEC according to the mean prevalence. Besides, the graphic displayed in Figure 6 also expresses the superiority of the *mcd* decision mode regarding median and variance of the prevalence values.

In Table 6, the results obtained for ECs where $n_{batch} = 128$ are better than those obtained in Table 5 for ECs where $n_{batch} \in \{4, 20, 50, 100\}$ for two reasons.

Table 6: Case of minimal resource idling and *no_update* policy. **Left:** best mean prevalence scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{128\}$ the decision mode producing the best mean prevalence better than NEC strategies (computed over 100 searches) is granted a point. **Right:** Best ECs according to the mean prevalence.

| Decision mode | Best mean prevalence score |
|---|---|
| *rand* | 0 |
| *mod* | 0 |
| *hcr* | 0 |
| *bp* | 0 |
| *mcd* | **3** |

| EC | Mean prevalence |
|---|---|
| *mcd*_0.25_128 | **24.7523** |
| *mcd*_0.50_128 | 24.7848 |
| *mod*_0.75_128 | 24.8137 |
| *hcr*_0.75_128 | 24.8201 |
| *rand*_0.50_128 | 24.8508 |

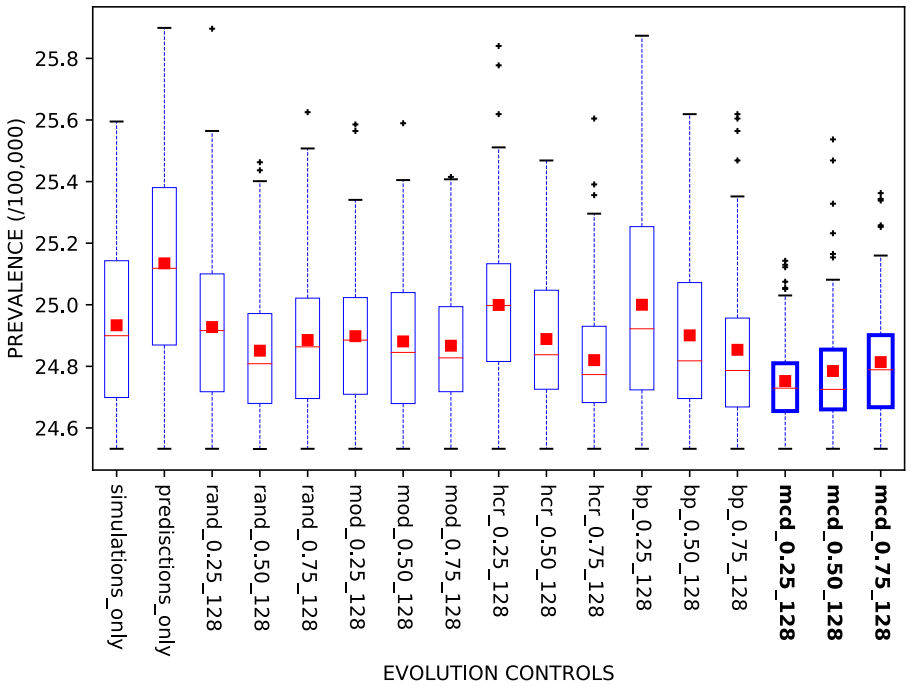


Figure 6: Distribution of the 100 best prevalence values found by the GA for each of the 100 searches for the *no_update* policy and minimal resource idling. Mean values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

First, the optimal usage of the CPU cores allows to ameliorate the budget consumption. Second, the population size increased to 128 (when $n_{batch} = 128$) enhances the GA exploration capability compared to a population size fixed to 100 (when $n_{batch} \in \{4, 20, 50, 100\}$).

### *6.4. Performance evaluation of the decision modes considering the update_effort policy*

Under the *update_effort* policy, the calibration of the ANN hyper-parameters is realized by a parallel grid-search involving 270 ANN hyper-parameters settings. The relative small size of the training set allows each setting to be trained on one core. The results reported in Table 7 define the ANN hyper-parameters values used from now on.

Table 7: Best ANN configuration for the TBTC according to a parallel grid-search hyper-parameters calibration involving 270 settings.

| | |
|---|---|
| number of layers | 1 |
| number of neurons | 18 |
| activation function | relu |
| learning rate | 0.5 |
| $p_{drop}$ | 0.1 |

In order to highlight the differences in terms of prediction accuracy between the two surrogate update policies, the surrogate error is monitored during the search on the TBTC. The graphic displayed in Figure 7 represents the surrogate error, computed as the absolute value of the difference between the simulated fitness and the predicted fitness, for each simulation performed during the search. During the first generation, the error committed by the surrogate trained under the *no_update* policy is higher than that under the *update_effort* policy. This result proves the beneficial impact of the hyper-parameters calibration task. Using the *no_update* policy, the surrogate error increases during the next generation, as the surrogate is not updated and the region of interest detected by the GA evolves. This phenomenon does not appear when the *update_effort* policy is used and demonstrates the contribution of performing incremental updates at the end of each generation.

The TBTC is now tackled by the 79 ECs on the 32-cores-per-node supercomputer with population size and batch size fixed to 128 to ensure the optimal usage of the CPU cores. The results shown in Table 8 demonstrate the superiority of the batch *bp*-based AECs according to the mean prevalence. Indeed, since the surrogate adapts dynamically to the regions of interest revealed during the search, the predicted fitness can be trusted. The *mcd* decision mode performs slightly worse than the *bp* decision mode and better than the remaining decision modes. Besides, the graphic displayed in Figure 8 confirms the superiority of *bp*

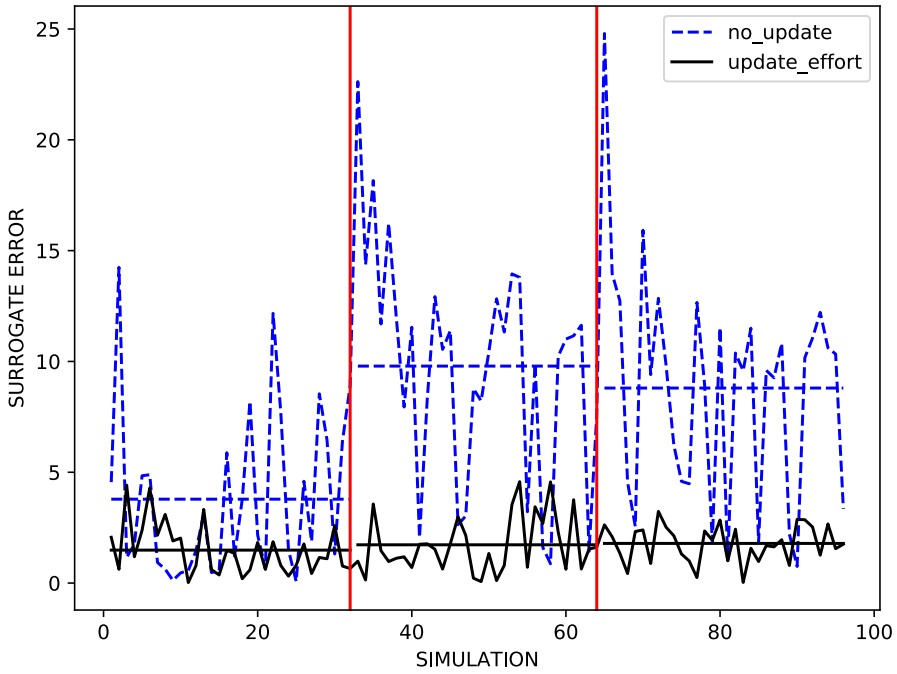


Figure 7: Surrogate error committed during the search for the two update policies. The blue dashed lines refer to the *no_update* policy while the **black** plain lines refer to the *update_effort* policy. Vertical red lines delimit the generations and horizontal blue and **black** lines represent the mean error computed for each policy and each generation.

and the rather good performance of the *mcd* and *hcr* decision modes regarding median and variance of the prevalence values.

Lower values of $p_{sim}$ induce a higher number of surrogate updates during the search. Conversely, higher values of $p_{sim}$ imply a lower number of surrogate updates but with a training set enriched with a higher number of new solutions. According to Table 8 a high update frequency is preferred. The update duration is moderate (approximately 1 second) in comparison to the initial ANN training (approximately 15 seconds). The difference in training time is due to the incremental way of update and the repetition of several training samples from one update to another.

### *6.5. Comparison with IFR-AECs and threshold-based MCDropout-AECs*

A crossover Informed Operator is proposed in order to compare DFR and IFR. From two parent solutions, four offspring solutions are generated using the same stochastic crossover operator. The offspring solutions are predicted and the best two solutions according to *bp*, *hcr* or *mcd* decision modes are retained for simulation. For IFR AECs, the batch of solutions exclusively contains solutions to be simulated and the GA population is exclusively composed

Table 8: Case of minimal resource idling and *update_effort* policy. **Left:** best mean prevalence scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{128\}$ the decision mode producing the best mean prevalence better than NEC strategies (computed over 100 searches) is granted a point. **Right:** Best ECs according to the mean prevalence.

| Decision mode | Best mean prevalence score |
|---|---|
| *rand* | 0 |
| *mod* | 0 |
| *hcr* | 0 |
| *bp* | **2** |
| *mcd* | 1 |

| EC | Mean prevalence |
|---|---|
| *bp*_0.25_128 | **24.6149** |
| *mcd*_0.25_128 | 24.6721 |
| *hcr*_0.25_128 | 24.6838 |
| *bp*_0.50_128 | 24.6917 |
| *rand*_0.50_128 | 24.7360 |

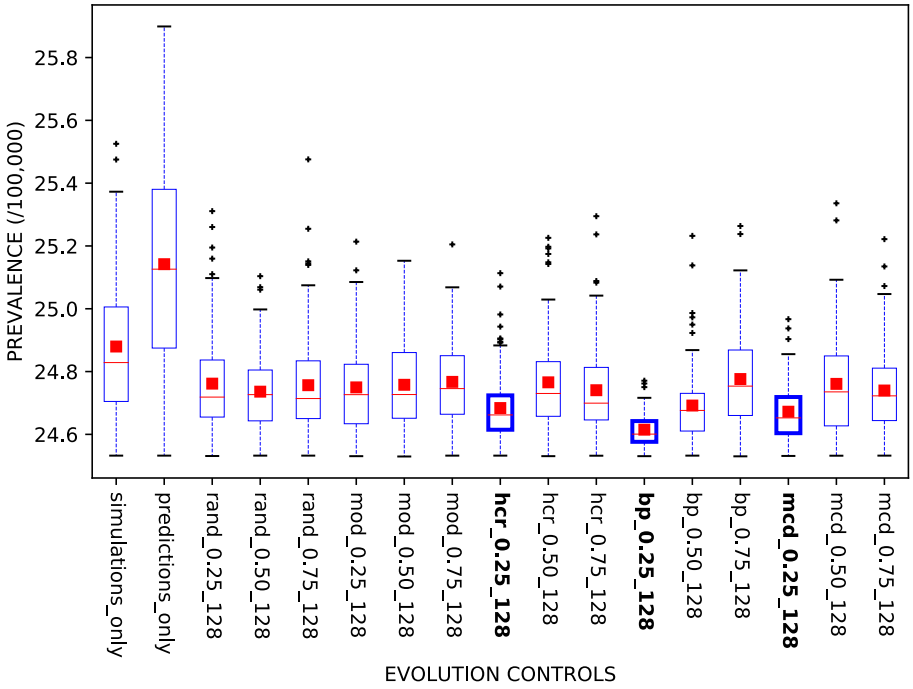


Figure 8: Distribution of the 100 best prevalence values found by the GA for each of the 100 searches considering the *update_effort* policy and minimal resource idling. Mean values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

of simulated solutions. IFR AECs with 32 and 64 solutions per batch are contemplated and produce the same surrogate update frequency than DFR AECs with $p_{sim} = 0.25$ and $p_{sim} = 0.5$ respectively.

The threshold variant of the *mcd* decision mode, named *THR-mcd*, is also investigated. *THR-mcd* consists in comparing the prediction uncertainty to a

threshold value $t$ that is updated at each surrogate update. The threshold value is defined as $t = V_{min} + (1 - p_{sim})(V_{max} - V_{min})$ where $V_{min}$ and $V_{max}$ are respectively the minimum and the maximum prediction uncertainty obtained on a set of $N$ solutions sampled randomly. If the prediction uncertainty of the current candidate solution is higher than the threshold, the candidate is simulated, otherwise the candidate is predicted.

Figure 9 proves the superiority of DFR AECs over the IFR AECs and the *THR-mcd* AECs regarding mean, median and variance of the prevalence values. Nevertheless, a broader study should be dedicated to investigate these two approaches, as there are numerous ways to design IFR and threshold-based AECs.

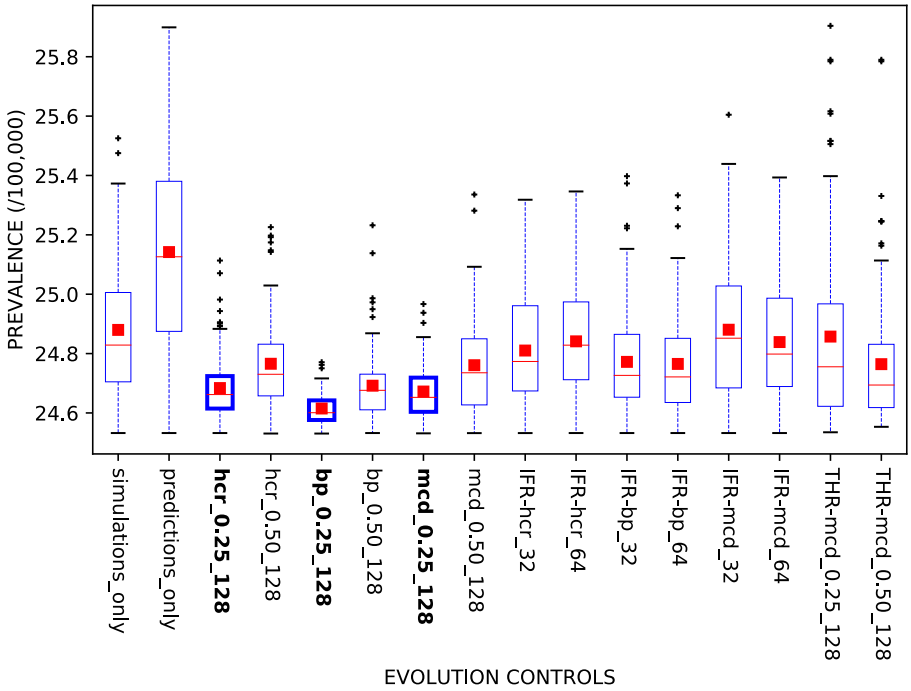


Figure 9: Distribution of the 100 best prevalence values from the 100 repetitions considering DFR, IFR and threshold-variant ECs, the *update_effort* policy and minimal resource idling. Mean values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

### *6.6. Comparison with combined ECs*

The observations drawn from Subsection 6.3 and 6.4 suggest to design a new combined AEC where the *bp* and *mcd* decision modes cooperate. It is proposed to rely on the Pareto dominance comparison according to the simultaneous minimization of the predicted cost $\hat{f}()$ and maximization of the prediction

Table 9: Case of minimal resource idling and *no_update* policy. **Left:** best mean prevalence scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{128\}$ the decision mode producing the best mean prevalence better than NEC strategies (computed over 100 searches) is granted a point. **Right:** Best ECs according to the mean prevalence.

| Decision mode | Best mean prevalence score |
|---|---|
| *bp* | 0 |
| *mcd* | **3** |
| *biobj-max* | 0 |
| *biobj-min* | 0 |

| EC | Mean prevalence |
|---|---|
| *mcd*_25_128 | **24.7523** |
| $biobj - min$_25_128 | 24.7722 |
| $biobj - max$_25_128 | 24.7841 |
| *mcd*_50_128 | 24.7848 |
| *mcd*_75_128 | 24.8137 |
| $biobj - max$_50_128 | 24.8179 |
| $biobj - min$_75_128 | 24.8181 |
| $biobj - max$_75_128 | 24.8256 |
| *bp*_75_128 | 24.8537 |
| $biobj - min$_50_128 | 24.8575 |
| *bp*_50_128 | 24.9005 |
| *bp*_25_128 | 25.0000 |

uncertainty $V()$. Let $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ be two candidate solutions. It is defined that $\boldsymbol{c_1}$ dominates $\boldsymbol{c_2}$ if and only if $\boldsymbol{c_1}$ is at least better than $\boldsymbol{c_2}$ regarding one metric and as good as $\boldsymbol{c_2}$ regarding the potential remaining metric.

The solutions dominating each candidate $\boldsymbol{c}$ from the batch are counted. The $n_{sim}$ candidates demonstrating the lowest number of dominating solutions are selected for simulation. The possible selection between candidates presenting the same number of dominating solutions is made randomly. This combined decision mode, named *biobj-max*, allows one to simulate candidate solutions presenting both a promising predicted fitness and a high prediction uncertainty.

The *biobj-max* decision mode is compared to the criterion proposed in [59] where the Pareto dominance is made according to minimization of the predicted fitness and minimization of the prediction uncertainty. This latter decision mode, named *biobj-min*, offers less exploitation than *biobj-max* since solutions with a high predicted fitness and a low prediction uncertainty can be considered for simulation.

Considering the *no_update* policy, the results presented in Table 9 confirm the superiority of the *mcd* decision mode compared with *bp*, *biobj-max* and *biobj-min*. The combined decision modes offer a tradeoff between the *mcd* and *bp* decision modes. *biobj-min* outperforms *biobj-max* as it offers less exploitation.

Considering the *update_effort* policy, the results exhibited in Table 10 show the superiority of *bp* decision mode over *mcd*, *biobj-max* and *biobj-min*. Nevertheless, the *biobj-max* and *biobj-min* decision modes perform consistently over different surrogate update frequencies. Indeed for $p_{sim} = 0.25$, *bp* provides the best result, followed by *biobj-max*, *biobj-min* and finally *mcd*. By contrast, for

Table 10: Case of minimal resource idling and *update_effort* policy. **Left:** best mean prevalence scores. For each pair $(p_{sim}, n_{batch}) \in \{0.25, 0.5, 0.75\} \times \{128\}$ the decision mode producing the best mean prevalence better than NEC strategies (computed over 100 searches) is granted a point. **Right:** Best ECs according to the mean prevalence.

| Decision mode | Best mean prevalence score |
|---|---|
| *bp* | **2** |
| *mcd* | 1 |
| *biobj-max* | 0 |
| *biobj-min* | 0 |

| EC | Mean prevalence |
|---|---|
| *bp*_25_128 | **24.6149** |
| $biobj - max$_25_128 | 24.6574 |
| $biobj - min$_25_128 | 24.6600 |
| *mcd*_25_128 | 24.6721 |
| *bp*_50_128 | 24.6917 |
| $biobj - min$_50_128 | 24.7035 |
| $biobj - max$_50_128 | 24.7173 |
| *mcd*_75_128 | 24.7396 |
| $biobj - min$_75_128 | 24.7530 |
| *mcd*_50_128 | 24.7607 |
| $biobj - max$_75_128 | 24.7657 |
| *bp*_75_128 | 24.7762 |

$p_{sim} = 0.75$, *mcd* produces the best result followed by *biobj-min*, *biobj-max* and finally *bp*. Increasing $p_{sim}$ decreases the number of surrogate updates which consequently deteriorates the surrogate accuracy. The *biobj-max* decision mode outperforms the *biobj-min* decision mode for a sufficiently high surrogate update frequency. The combined decision modes appear to be a reliable compromise when no indication is available about the surrogate accuracy. Our further studies will be dedicated to investigate the combined decision modes more deeply.

## 7. Conclusions and future works

In this paper, we successfully apply parallel surrogate-assisted evolutionary optimization to design control policies for the fight against tuberculosis. Emergence of tuberculosis transmission dynamic models coupled with economic models and parallel surrogate-assisted optimization assist decision makers in their effort to reach the End TB targets fixed by the World Health Organization.

A fundamental question arising when resorting to surrogate models in optimization subject to a limited running time budget is how to decide whether to simulate or to predict a candidate solution. To provide part of the answer, we propose two new batch Adaptive Evolution Controls based on MCDropout. MCDropout is a deep learning technique that provides uncertainty information about Artificial Neural Networks predictions. With the first new batch Adaptive Evolution Control, the candidate solutions with the highest prediction uncertainty are simulated. As reported by the experiment outcomes, a small number of predictions $k = 4$ is sufficient to provide an informative prediction uncertainty. The second batch Adaptive Evolution Control proposed considers non-dominated sorting based on prediction uncertainty maximization

and predicted fitness minimization.

The new batch Adaptive Evolution Controls are tested on benchmark problems and on the design of an intervention plan to reduce tuberculosis prevalence in the Republic of Fiji by 2035. The results indicate the superiority of the first new batch Adaptive Evolution Control when few effort is engaged in the surrogate training. When some efforts are engaged in the surrogate training, the Adaptive Evolution Control favoring the best predicted solutions shows to perform well. The second new batch Adaptive Evolution Control combines both the predicted fitness and the prediction uncertainty. This latter performs consistently over the problems considered and outperforms a similar combined Evolution Control for high surrogate update frequencies.

Finally, cluster-based parallel computing has been considered at three levels: ANN hyper-parameters calibration, parallel batch simulation of multiple candidate solutions and repetition of experiments. In the future, we also plan to complexify the model considering other parameters that influence TB transmission. As a consequence, the computational burden will be increased at least at two levels: the simulation will be more time-demanding and the size of the deep neural network will be larger. This will make the use of extreme-scale GPU-powered computers necessary. Different challenges will therefore be raised including scalability and heterogeneity (multi-core+GPU-accelerated computing).

## 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. Müller and C.A. Shoemaker. Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box glo bal optimization problems. *Journal of Global Optimization*, 60(2):123–144, Oct 2014. doi: `https://doi.org/10.1007/s10898-014-0184-0`.

[2] R. Duvigneau and M. Visonneau. Hybrid genetic algorithms and artificial neural networks for complex design optimization in cfd. *International Journal for Numerical Methods in Fluids*, 44(11):1257–1278, 2004. doi: `http://dx.doi.org/10.1002/fld.688`.

[3] 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.

[4] B. Glaz, P.P. Friedmann, and L. Liu. Surrogate based optimization of helicopter rotor blades for vibration reduction in forward flight. *Structural and Multidisciplinary Optimization*, 35(4):341–363, Apr 2008.

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

[6] J.J. Dongarra, L.S. Duff, D.C. Sorensen, and H.A.V. Vorst. *Numerical Linear Algebra for High Performance Computers*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[7] S. Patil and G. Kale. Survey on gpu based linear solver. *International Journal of Science Technology & Engineering*, 2, 05 2016.

[8] E. Alba, G. Luque, and S. Nesmachnow. Parallel metaheuristics: Recent advances and new trends. *International Transactions in Operational Research*, 20:1–48, 01 2012.

[9] T. Van Luong, N. Melab, and E.G. Talbi. Gpu-based island model for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1089–1096. ACM, 2010.

[10] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61 – 70, 2011. doi: https://doi.org/10.1016/j.swevo.2011.05.001.

[11] K. Khac Vu, C. D'Ambrosio, Y. Hamadi, and L. Liberti. Surrogate-based methods for black-box optimization: Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24, 04 2016.

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

[13] M. Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[14] Global tuberculosis report 2018., 2018.

[15] Who end tb strategy. global strategy and targets for tuberculosis prevention, care and control after 2015., 2015.

[16] J.M.C 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.

[17] 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.

[18] A. Sóbester, S.J. Leary, and A.J. Keane. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27(5):371–383, Jul 2004.

[19] T. Akhtar and C.A. Shoemaker. Multi objective optimization of computationally expensive multi-modal functions with rbf surrogates and multi-rule selection. *Journal of Global Optimization*, 64(1):17–32, Jan 2016.

[20] R.G. Regis and C.A. Shoemaker. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing*, 21(3):411–426, 2009.

[21] Y. Ong, P. Nair, and A. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 04 2003.

[22] C. Li, F.L. Wang, Y.Q. Chang, and Y. Liu. A modified global optimization method based on surrogate model and its application in packing profile optimization of injection molding process. *The International Journal of Advanced Manufacturing Technology*, 48(5):505–511, May 2010.

[23] D. Zhan, J. Qian, and Y. Cheng. Pseudo expected improvement criterion for parallel ego algorithm. *Journal of Global Optimization*, 68(3):641–662, Jul 2017.

[24] 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.

[25] R.G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.

[26] K. Deb and P. Nain. An Evolutionary Multi-objective Adaptive Metamodeling 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`.

[27] G. Briffoteaux, N. Melab, M. Mezmaz, and D. Tuyttens. An adaptive evolution control based on confident regions for surrogate-assisted optimization. In *HPCS 2018 - International Conference on High Performance Computing & Simulation*, Orléans, France, July 2018.

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

[29] A. Díaz-Manríquez, G. Toscano, J.H. Barron-Zambrano, and E. Tello-Leal. A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*, 2016:14, 2016. doi: `https://doi.org/10.1155/2016/9420460`.

[30] M.S. Innocente, S.M. Bastos Afonso, J. Sienz, and H.M. Davies. Particle swarm algorithm with adaptive constraint handling and integrated surrogate model for the management of petroleum fields. *Applied Soft Computing*, 34:463 – 484, 2015.

[31] T. Chugh, N. Chakraborti, K. Sindhya, and Y. Jin. A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. *Materials and Manufacturing Processes*, 32(10):1172–1178, 2017.

[32] S. Choi, J.J. Alonso, and H.S. Chung. Design of a low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method. *Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 2692–2706, 04 2004.

[33] Y. Lian and M.S. Liou. Multi-objective optimization of transonic compressor blade using evolutionary algorithm. *Journal of Propulsion and Power, vol. 21, no. 6*, page 979–987, 2005.

[34] L. Gonzalez, K. Srinivas, J. Periuax, and E. Whitney. A generic framework for the design optimisation of multidisciplinary uav intelligent systems using evolutionary computing. *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, page 1–19, 2006.

[35] T. Goel, R. Vaidyanathan, R.T. Haftka, W. Shyy, N.V. Queipo, and K. Tucker. Response surface approximation of pareto optimal front in multi-objective optimization. *Computer Methods in Applied Mechanics and Engineering*, 196(4):879 – 893, 2007.

[36] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35(6):561–569, Jun 2008.

[37] I. Voutchkov and A. Keane. *Multi-Objective Optimization Using Surrogates*, pages 155–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[38] L.G. Fonseca, H.J.C. Barbosa, and A.C.C. Lemonge. *On Similarity-Based Surrogate Models for Expensive Single- and Multi-objective Evolutionary Optimization*, pages 219–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[39] J. Sreekanth and B. Datta. Multi-objective management of saltwater intrusion in coastal aquifers using genetic programming and modular neural network based surrogate models. *Journal of Hydrology*, 393(3):245 – 256, 2010.

[40] A. Rosales-Pérez, C.A. Coello Coello, J.A. Gonzalez, C.A. Reyes-Garcia, and H.J. Escalante. A hybrid surrogate-based approach for evolutionary multi-objective optimization. In *2013 IEEE Congress on Evolutionary Computation*, pages 2548–2555, June 2013.

[41] 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.

[42] H. Wang, Y. Jin, and J. Doherty. Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. *IEEE Transactions on Cybernetics*, 47(9):2664–2677, Sep. 2017.

[43] 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`.

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

[45] 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.

[46] R. Maclin and D.W. Opitz. Popular ensemble methods: An empirical study. *CoRR*, abs/1106.0257, 2011.

[47] X. Lin, H.L. Zhen, Z. Li, Q. Zhang, and S. Kwong. A batched scalable multi-objective bayesian optimization algorithm. 2018. `http://export.arxiv.org/pdf/1811.01323`.

[48] 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.

[49] F. Chollet et al. Keras. `https://keras.io`, 2015.

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

[51] F. Biscani et al. esa/pagmo2: pagmo 2.10, January 2019.

[52] List of problems available in pagmo/pygmo. `https://esa.github.io/pagmo2/docs/problem_list.html`.

[53] L.J. Abu-Raddad et al. Epidemiological benefits of more-effective tuberculosis vaccines, drugs, and diagnostics. *Proceedings of the National Academy of Sciences*, 106(33):13980–13985, 2009.

[54] J.M.C. Trauer, J.T. Denholm, 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.

[55] N. Scott et al. Maximizing the impact of malaria funding through allocative efficiency: using the right interventions in the right locations. *Malaria Journal*, 16(1):368, Sep 2017.

[56] M. Keeling, P. Rohani, and B. Pourbohloul. Modeling infectious diseases in humans and animals. *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America*, 47:864–865, 10 2008.

[57] C.C. Boehme et al. Rapid molecular detection of tuberculosis and rifampin resistance. *New England Journal of Medicine*, 363(11):1005–1015, 2010. PMID: 20825313.

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

[59] J. Tian, Y. Tan, J. Zeng, C. Sun, and Y. Jin. Multiobjective infill criterion driven gaussian process-assisted particle swarm optimization of high-dimensional expensive problems. *IEEE Transactions on Evolutionary Computation*, 23(3):459–472, June 2019.