Designing parallelism in surrogate-assisted multiobjective optimization based on decomposition

Nicolas Berveglieri, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, Qingfu Zhang, Kiyoshi Tanaka

Designing parallelism in Surrogate-assisted multiobjective optimization based on decomposition

Nicolas Berveglieri
Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, F-59000 Lille, France
nicolas.berveglieri@univ-lille.fr

Bilel Derbel
Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, F-59000 Lille, France
bilel.derbel@univ-lille.fr

Arnaud Liefooghe
JFLI - CNRS IRL 3527, University of Tokyo, Tokyo, Japan
arnaud.liefooghe@univ-lille.fr

Hernán Aguirre
Shinshu University, Faculty of Engineering, Nagano, Japan
ahernan@shinshu-u.ac.jp

Qingfu Zhang
City University Hong Kong Kowloon Tong, Hong Kong
qingfu.zhang@cityu.edu.hk

Kiyoshi Tanaka
Shinshu University, Faculty of Engineering, Nagano, Japan
ktanaka@shinshu-u.ac.jp

ABSTRACT

On the one hand, surrogate-assisted evolutionary algorithms are established as a method of choice for expensive black-box optimization problems. On the other hand, the growth in computing facilities has seen a massive increase in potential computational power, granted the users accommodate their approaches with the offered parallelism. While a number of studies acknowledge the impact of parallelism for single-objective expensive optimization assisted by surrogates, extending such techniques to the multi-objective setting has not yet been properly investigated, especially within the state-of-the-art decomposition framework. We first highlight the different degrees of parallelism in existing surrogate-assisted multi-objective evolutionary algorithms based on decomposition (S-MOEA/D). We then provide a comprehensive analysis of the key steps towards a successful parallel S-MOEA/D approach. Through an extensive benchmarking effort relying on the well-established `bbob-biobj` test functions, we analyze the performance of the different algorithm designs with respect to the problem dimensionality and difficulty, the amount of parallel cores available, and the supervised learning models considered. In particular, we show the difference in algorithm scalability based on the selected surrogate-assisted approaches, the performance impact of distributing the model training task and the efficacy of the designed parallel-surrogate methods.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**; *Gaussian processes*; Algorithm design techniques.

KEYWORDS

Multiobjective optimization, surrogates, parallelism benchmarking.

ACM Reference Format:
Nicolas Berveglieri, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, Qingfu Zhang, and Kiyoshi Tanaka. 2020. Designing parallelism in Surrogate-assisted multiobjective optimization based on decomposition. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3377930.3390202

1 INTRODUCTION

Context. Solving a multi-objective optimization problem (MOP) aims at optimizing simultaneously a number of objectives which are often conflicting with each other. This means that there is not one optimal solution, but a whole set of solutions offering different trade-offs between the objectives. Multi-objective evolutionary algorithms (MOEAs) have been proven particularly effective to solve black-box MOPs. However, a major drawback of MOEAs is the amount of solutions that need to be effectively evaluated in order to converge to a high-quality approximation set. In fact, it is very common to observe some MOEAs requiring a budget of few hundred of thousands up to millions of objective function evaluations. While this is acceptable for low-cost objectives, this can cause a serious bottleneck when tackling *expensive* MOPs, for which one evaluation requires a high computational effort. This drastically restricts the overall number of function evaluations the practitioner can afford for the problem to be solved in a reasonable and manageable amount of time. With respect to expensive MOPs, different studies investigate the design of novel MOEAs assisted by surrogate models, considered as a powerful tool to speed up convergence towards high-quality approximation sets. The main idea explored so far with respect to surrogates for MOPs is to build one or many data-driven meta-model(s) offering a cheap alternative to the expensive objective functions. It is hence possible for a standard evolutionary algorithm to perform at its best using the constructed models, thus allowing to pre-screen one or multiple promising solutions that can be evaluated subsequently using the original and expensive objective functions. For a recent survey on state-of-the-art surrogate-assisted approaches, the reader is referred to [2, 3, 5, 9, 11]. In this paper, we are specifically interested in coupling the benefits of *parallel* computing resources with the use of surrogate model for solving expensive MOPs.

Positioning and Related Work. On the one hand, the ever-increasing availability of computing resources, the advent of new computing facilities and of robust large-scale and massively parallel platforms opens tremendous research opportunities for pushing forward the development and uptake of evolutionary algorithms. A huge body of literature exists on the design of parallel and distributed optimization algorithms in general, e.g., [15, 20]. Although reviewing the literature is out-of-the-scope of this paper, let us comment that three main classes of parallel approaches are usually distinguished: (i) those exposing problem-dependent parallelism, typically for speeding up the cost of evaluating one single candidate solution, (ii) those exposing low-level parallelism with the goal of providing a substantial parallel speedup when deploying an optimization algorithm, for instance by evaluating multiple solutions in parallel, and (iii) those exposing high-level parallelism, typically referring to the situation where multiple, possibly different, search processes are executed in a cooperative and parallel manner, hence possibly improving the search quality and not only the computing time. On the other hand, although being particularly accurate for dealing with expensive problems, it is not clear how the recently proposed surrogate-assisted MOEAs can be adapted and scaled efficiently with respect to the available computing resources, while allowing to attain improved approximation quality.

Methodology and Contribution Overview. We are aware of relatively few investigations eliciting, in a comprehensive manner and through a systematic analysis approach, the challenges underlying the parallel design of surrogate-assisted MOEAs, together with the design options one can adopt. This is precisely the general goal we would like to contribute. Consequently, our aim is *not* to parallelize an existing surrogate-assisted approach nor to speedup a particular algorithm, but rather to investigate the different opportunities and design options that can lead to an effective parallel surrogate-assisted MOEA.

More precisely, and in order to focus more deeply on the optimization challenges, we leave behind the scene the characteristics of the intended computing platforms and the corresponding parallel and technological implementation issues. We instead consider an abstract setting where it is simply assumed that some processing units (PUs) are available and can communicate using some abstract communication medium. As such, we focus on enabling parallelism within the state-of-the-art MOEA based on decomposition (MOEA/D) [26]. As it will be discussed further in the rest of the paper, decomposition, and more generally aggregation-based MOEAs, are based on a divide-and-conquer principle which is by nature of high degree of parallelism [4, 24], as well as a high degree of flexibility in order to leverage existing surrogate-assisted algorithms from both single- and multi-objective optimization [2]. In this context, our contributions can be summarized as follows:

- In light of the recent taxonomies developed by the MOEA community for surrogate-assisted approaches [3, 9], we propose to identify and classify in a high-level manner the different levels of parallelism offered by decomposition approaches. Thereby, we discuss different classes of parallel designs for attacking expensive MOPs. This allows us to set up nine algorithm variants instantiated with different components and aiming to push a step further for a better understanding of what makes a surrogate-assisted approach effective in terms of approximation quality when enabling parallelism.
- Through extensive experiments including a range of bi-objective benchmark functions from the well-established `bbob-biobj` test suite covering different problem properties [18], we focus on the gain in approximation quality when using an increasing number of computing units.
- We investigate the convergence profile of the different approaches and we report insights into the ability of a parallel methodology to provide novel and improved surrogate-assisted algorithm design when tackling expensive MOPs. For instance, we found that parallelism can lead to an *ensemble* design allowing to deal in a natural manner with objectives having a different degree of difficulty.

Outline. In Section 2, we provide the necessary background on surrogate-assisted MOEAs. In Section 3, we discuss key parallel design. In Section 4, we report our empirical findings

2 BACKGROUND AND POSITIONING

In this section, we provide an overview of existing surrogate-assisted MOEAs with a focus on those based on decomposition. This will serve in the following section as a baseline for the proposed parallel designs.

2.1 Decomposition-based MOEAs

Let us assume a target MOP, defined as a vector function $F\colon \mathbb{R}^d \mapsto \mathbb{R}^m$, with d variables and m objectives to be minimized. Given two solutions $x, x' \in \mathbb{R}^d$, x' is dominated by x iff, for all $i \in \{1, \ldots, m\}$, $f_i(x) \leqslant f_i(x')$, and there is a $j \in \{1, \ldots, m\}$ such that $f_j(x) < f_j(x')$. A solution $x^\star \in \mathbb{R}^d$ is Pareto optimal if there does not exist any $x \in \mathbb{R}^d$ such that $x^\star$ is dominated by x. The set of all non-dominated solutions is the Pareto set. Its mapping in the objective space is the Pareto front.

A successful class of MOEAs, that includes MOEA/D [26], rely on the decomposition of the original MOP into a number of single-objective sub-problems that are expected to be easier to solve [24, 26]. Given a weight vector $w \in \mathbb{R}^m$, the scalarizing function $g(x \mid w)$ assigns a scalar value to any solution $x \in \mathbb{R}^d$. By generating multiple weight vectors for configuring the scalarizing function, multiple sub-problems are defined whose solution targets a different region of the Pareto front.

2.2 Surrogate-assisted MOEAs

When targeting expensive MOPs, we focus on two main classes of surrogate-assisted MOEA/D (S-MOEA/D) approaches: (i) approaches based on filtering (and to a lesser extent their extensions to substitution) [2, 11, 14], and (ii) approaches based on Gaussian processes [10, 12, 27]. They are briefly discussed below.

2.2.1 Filtering-based approaches. Surrogate-assisted approaches based on filtering [14] are perhaps the most obvious technique to tackle expensive problems. They are relatively simple and flexible enough to be plugged in a wide range of evolutionary algorithms. A filtering approach is usually divided into three sequential steps. Firstly, one or multiple regression models are built in order to fit

the different objective functions using a given training set. This step is repeated regularly, when new solutions are evaluated using the expensive functions, in order to maintain a surrogate model which is as accurate as possible. Secondly, they are based on the generation of a whole set of solutions in an iterative manner, for which the estimate fitness values is computed using the underlying surrogate. Thirdly, on the basis of these estimations, the most promising solution is selected and evaluated by means of the real (and costly) evaluation function. Hence, it becomes clear that a filtering approach acts as a pre-screening of offspring solutions and rely on the surrogate model to help for better-informed selection. Filtering methods have been paired with MOEAs and MOEA/D within, e.g., the MOEA/D-SVM algorithm [14]. Related approaches are also discussed [2].

2.2.2 Gaussian Process-based approaches. An alternative strategy for expensive optimization is the Efficient Global Optimization (EGO) approach [12]. EGO uses Gaussian Processes (Kriging) as a surrogate model to estimate the objective values. Gaussian processes are of particular interest since they provide both an estimate of solution quality and an uncertainty around this prediction. Hence, different alternative acquisition functions can be considered to sample a promising solution, such as the expected improvement (EI) or the maximum probability of improvement (MPI). As such, EGO does not specifically targets the solution with the best prediction, but it rather seeks for the solution that optimizes a particular acquisition function. This enables to balance the search between exploitation, when selected solutions have high expected values, and exploration, when selected solutions have high uncertainty. EGO has been coupled with MOEAs within, e.g., the ParEGO [13], Multi-EGO [10] and MOEA/D-EGO [27] approaches.

2.3 Overview of Parallel Approaches

We can basically distinguish between two categories of algorithms: (i) those that were developed to support an effective parallel execution, and (ii) those that were *not* initially designed with a parallel mindset, but that expose some natural degrees of parallelism. It should also be noted that surveys about parallel surrogate-assisted single-objective optimization can be found in [8, 21]

Looking at the first category, one can find relatively few specialized parallel surrogate-assisted algorithms. A remarkable approach is the so-called multi-point expected improvement approach (Q-EI) described in [22] and developed in [7]. The Q-EI approach is basically an extension of the EI acquisition function where, instead of searching for one single point optimizing the expected improvement in a Gaussian process, a whole set of q solutions is computed. The Q-EI is specifically designed to support the sampling and subsequent parallel evaluation of multiple solutions. This is designed in such a way that a similar exploration/exploitation trade-off than the well-established EI acquisition function can be attained, specifically in a parallel environment with *a priori* any scale, q being a user-defined parameter. We however remark that the Q-EI approach was initially designed for single-objective problems, and leveraging it to MOPs is in its own an open research question.

More recently, a parallel population-based approach is described in [1], where a classic filtering approach, based on a Radial Basis function (RBF), and a local candidate search around multiple points to balance exploration and exploitation, are combined in parallel with multiple solution evaluations. In [23], a filtering method is used where the offspring selection is adjusted to consider the error of the surrogate predictions. In particular, filters are built asynchronously based on the available computing resources. Despite their skillful design, such focused approaches do not elicit in a systematic manner the different degrees of parallelism that can be explored further.

Turning to the second category of approaches, which expose an implicit degree of parallelism, representative algorithms are MOEA/D-RBF [25] and MOEA/D-EGO [27]. Therein, a number of models of different nature are constructed using different data sets and trained in a sequential manner. Hence, a straightforward parallelization is to train the underlying models in parallel. Similarly, most S-MOEA/D approaches, such as [10, 14, 25], and more generally any standard filtering or substitution approach [2], compute a whole set of solutions at each iteration, from which some are selected for expensive evaluation. A standard sequential design choice consists in restricting the number of solutions that are effectively selected for evaluation in order to reduce the budget consumption. Hence, it is clear that parallelism can be supported very naturally by simply evaluating the promising solutions in parallel, in a way that it matches the resources available.

Considering these two examples, it should be clear that either the model training itself or the expensive evaluation step can be parallelized within existing surrogate-assisted approaches. This hence offers two obvious levels of parallelism: (i) constructing and training different models in parallel, and (ii) evaluating multiple promising solutions in parallel. Nevertheless, it remains unclear how these two levels can support massive parallelism, and how they influence solving quality and efficiency. For example, the previously-mentioned algorithms such as MOEA/D-RBF [25] and MOEA/D-EGO [27] train very few models, so that a typical setting where much more computing cores are available would not fully benefit from a straightforward parallel execution. Similarly, it is unclear how existing filtering approaches would manage an increasing number of parallel evaluations without biasing the search process or effectively attaining the best approximation quality. This can be attributed to the fact that, although they expose interesting levels of parallelism, such approaches were not designed nor properly analyzed and configured in order to support effective parallelism.

Interestingly, the class of decomposition-based algorithms, on top of which many of the surrogate-assisted approaches listed above are built, offers a high degree of parallelism. In fact, a number of studies have already investigated the distribution of the computational flow of MOEA/D based on the fact that the underlying sub-problems could be solved cooperatively in parallel given some extra communication effort; see, e.g., [4, 6, 21]. Accordingly, we also investigate the possibility of coupling the two aforementioned levels of parallelism, exposed in existing surrogate-assisted approaches, with the parallel nature of decomposition-based MOEAs.

3 PARALLEL S-MOEA/D APPROACHES

In the following, we propose a principled parallel design, and accordingly a number of representative surrogate-assisted algorithms. We first discuss some key design aspects, allowing us to later detail the proposed parallel S-MOEA/Ds as summarized in Table 1.

Table 1: S-MOEA/D techniques as instances of the general framework (PU is for processing unit).

Algorithm	Generation and evaluation of candidate solutions	Model training set	Model building	Model type and response
MOEA/D-Multistart	concurrent algorithm runs	–	–	–
S-MOEA/D-QEI	MOEA/D on Q-EI	fuzzy clustering	local models built on available PUs	Gaussian process for EI
S-MOEA/D-MEI	MOEA/D on EI	fuzzy clustering	local models built on available PUs	Gaussian process for EI
S-MOEA/D-LocalBatch	per sub-problem parallel evaluation	weight-centered clustering	local models built on a single PU	SVR for filtering
S-MOEA/D-GlobalBatch	per sub-problem parallel evaluation	fuzzy clustering	local models built on available PUs	SVR for filtering
S-MOEA/D-Multimodel	per sub-problem parallel evaluation	fuzzy clustering	local models built on available PUs	multiple models/kernels for filtering
MP-S-MOEA/D-W	parallel sub-problem solving	weight-centered clustering	local models built on sub-problem specific PU	SVR for filtering
MP-S-MOEA/D-F	parallel sub-problem solving	fuzzy clustering	local models built on available PUs	SVR for filtering
MP-S-MOEA/D-G	parallel sub-problem solving	full data set	global model built on a single PU	SVR for filtering

3.1 Key Design Issues for Parallel S-MOEA/D

3.1.1 Generation and Evaluation of Candidate Solutions. The most straightforward aspect is to enable the parallel evaluation of multiple solutions, since this is *a priori* the most critical and CPU-intensive operation. However, this can be achieved in several ways, impacting the algorithm design by itself. In fact, before even evaluating a pool of solutions, it is required to generate them. This can be done either in a sequential manner by one processing unit (PU), or distributively by a set of PUs. Different alternatives can be though, ranging from standard variation from evolutionary computation, to multiple models-guided pool generation.

3.1.2 Training Data and Model Building. Here we are concerned with both the amount of data (from solutions evaluated so far) to use for training, and how many models are to be used. Firstly, choosing the learning data set is critically important, since it can highly impact the outcome of the learning phase and hence the accuracy of the surrogate(s). A straightforward option is to use all the data available. However, this might lead to the construction of a global surrogate at a high computational cost, which can even dominate the expensive evaluation itself in some cases. This is particularly true when trying to scale the amount of available PUs, given that the amount of evaluated solutions should typically scale accordingly. Interestingly, in an attempt to reduce the computational cost of learning, common algorithms often consider to use a subset of the data available, such as: the most recently evaluated solutions, the best performing solutions, or even simply random solutions. Other algorithms (not necessarily parallel) cluster the data in order to both reduce the training cost and provide not solely one, but an ensemble of models that are jointly used to improve the prediction accuracy and then the candidate solution generation [2, 27]. Focusing on decomposition approaches, and since the aim is to optimize simultaneously different sub-problems, training data can be naturally split following the so-defined sub-problems [2, 10]. It should be clear that such considerations provide multiple opportunities when turning into a parallel design. Indeed, as for the evaluation step distributed among multiple PUs, the training phase can occur distributively or not, while clustering or not the learning data and using one or multiple models possibly trained in parallel as well.

3.1.3 Model Type and Response. Provided that multiple models are used, eventually using different subset of learning data, the choice of the model type is another important design issue. In most approaches, one model type and one specific kernel, when applicable, are selected before the search starts, and are subsequently used throughout the entire process. Given that we are considering black-box problems, no specific knowledge about the objective functions is available. As such, the choice of a particular surrogate can be critical, and a better option would be to use different model types in order to eventually better fit the unknown characteristics of the black-box functions. Furthermore, the objectives might typically expose different (unknown) characteristics, hence making a particular model more accurate for some functions. Such considerations are not well documented, and very little studies on the subject exist apart from [25].

3.2 Proposed Parallel S-MOEA/D Algorithms

We introduce, in this section, eight parallel S-MOEA/D algorithms based on the different components depicted in Table 1. First, let us define an *iteration* in the context of parallel S-MOEA/D: a full iteration is completed when all PUs have computed the objective values of one new solution using the true (costly) evaluation function. At each iteration, every algorithm: (1) constructs a training data set, (2) updates its surrogate models, (3) generates a pool of new solutions (offspring) for potential evaluation, and (4) selects p solutions from this pool for evaluation, p being the number of PUs.

For selecting the solutions to be evaluated from the pool of offspring, our parallel S-MOEA/D approaches discriminate between two algorithm designs, based on the decomposition strategy:

- **(Sa)** Considering a single sub-problem per iteration: at a given iteration, the entire set of solutions to be evaluated is made of p promising solutions targeting *one* specific sub-problem; i.e. one search direction in MOEA/D.
- **(Sb)** Considering the whole set of sub-problems at each iteration: at a given iteration, the set of solutions is made of p promising solutions for *all* sub-problems, then targeting different regions of the objective space.

In both cases listed above, a pool of candidate solutions is to be generated and selected based on variation operators and surrogate estimations. Here again, we consider two different designs:

- **(Ga)** Employing a filtering-based approach where, for a given sub-problem, candidate solutions are generated by means of variation operators, following the standard sub-problems cooperation from MOEA/D.
- **(Gb)** Employing an acquisition function from Gaussian processes, where solutions are selected based on a given criterion such as Expected Improvement (EI) and its parallel variants [7, 22].

Whatever the design choice, the issues of training data and model building remain essential. This is particularly true for the filtering-based approach (**Ga**), where the algorithm designer is not bound to any specific surrogate, in contrast with EGO which is stuck to Gaussian processes. This allows us to explore a third level of design choices for filtering-based approaches in terms of models and training data clustering. Notice that the baseline model that we consider for prediction in filtering-based approaches is support-vector regression (SVR) with a radial basis function (RBF) kernel [19]. At last, in terms of parallelism, we differentiate between the parallel evaluation, which is inherent to the per-iteration batch evaluation, and parallel model training, where models based on clustering or ensemble methods can be naturally trained on different PUs before being dispatched, if necessary.

Based on the aforementioned considerations, we sketch below the proposed parallel S-MOEA/Ds as summarized further in Table 1.

S-MOEA/D-QEI is based on a given number of sub-problems, set independently of the number of PUs available. At each iteration, a single sub-problem is considered (**Sa**): (1) the master process divides all solutions evaluated so far into c groups of similar size using the fuzzy clustering procedure from [27]; (2) each cluster is sent to one PU for fitting a Gaussian process for one objective, so as to balance the workload among PUs; (3) the master process runs a single-objective evolutionary algorithm [27] for identifying p solutions optimizing the Q-EI acquisition function (**Gb**) of the current sub-problem scalarizing function; (4) each PU computes the objective values of one solution identified by the evolutionary algorithm using the (costly) evaluation function, so that p solutions are evaluated in parallel on the p PUs, the evaluated solutions are then communicated to the master process.

S-MOEA/D-MEI follows a similar principle than S-MOEA/D-QEI, except that p sub-problems are considered at each iteration (**Sb**). The main difference appears at stage (3) where each PU runs its own single-objective evolutionary algorithm [27] for optimizing the EI acquisition function (**Gb**) of the scalarizing function of its own sub-problem.

S-MOEA/D-LocalBatch considers a single sub-problem per iteration (**Sa**). At each iteration: (1) the master process runs the weight sub-problem centered procedure from [10] to construct the training set for the current sub-problem; (2) one SVR model is constructed for the current sub-problem scalarizing function and deployed among all PUs; (3) each PU generates a pool of λ offspring by means of variation operator(s), where λ is the filter size, and selects one offspring to be evaluated based on the model estimations (**Ga**); (4) each PU evaluates the selected solution so that p solutions are evaluated in parallel on the p PUs, the evaluated solutions are then communicated to the master process.

S-MOEA/D-GlobalBatch also considers a single sub-problem per iteration (**Sa**). However, at each iteration: (1) c clusters are now constructed by the master process using fuzzy clustering [27]; and (2) each cluster is sent to one PU for model fitting of one objective based on SVR, so as to balance the workload among PUs, and the models are deployed among all PUs. The remaining steps follow the same mechanisms than S-MOEA/D-LocalBatch (**Ga**).

S-MOEA/D-Multimodel considers a single sub-problem per iteration (**Sa**) and employs a filtering-based approach (**Ga**) as well. The main difference with S-MOEA/D-GlobalBatch appears at stage (2) where not one, but multiple models are trained for each cluster and each objective; i.e. not only SVR with an RBF kernel is considered for the prediction, but also cubic, multi-quadratic or Gaussian kernels [25], as well as different models such as RBF neural networks and kernel ridge regressor (KRR) [19, 25]. Assuming we have m model types and c clusters, they are then dispatched among the p PUs. In case $p < c \cdot m$, p models are selected at random, and in case $p > c \cdot m$, some models are duplicated in order to fill all PUs.

MP-S-MOEA/D-W follows a fully distributed parallel design [4] and considers all p sub-problems at each iteration (**Sb**). At each iteration: (1) each PU constructs a training set based on all solutions that have been evaluated so far *locally*, that is on in this particular PU as well as the solutions shared by its neighbor during the algorithm; (2) each PU trains its own SVR models, one for each objective; (3) each PU generates a pool of λ offspring by means of variation operator(s), and selects one offspring to be evaluated based on the model estimations (**Ga**); (4) each PU evaluates the selected solution so that p solutions are evaluated in parallel on the p PUs, the evaluated solutions are then communicated to PUs mapping to its neighboring sub-problems MOEA/D [26].

MP-S-MOEA/D-F follows a workflow similar to the previous algorithm, by considering all sub-problems at each iteration (**Sb**) and a filtering-based approach (**Ga**). However, all solutions evaluated so far are here maintained on a master process that iteratively (1) build c clusters using fuzzy clustering [27], and (2) send each cluster to one PU for model fitting of one objective, so as to balance the workload among PUs. The models are then deployed among all PUs, and the remaining steps follow the same mechanisms than MP-S-MOEA/D-W. All solutions evaluated at a given iteration are then communicated to the master process.

MP-S-MOEA/D-G is the same than MP-S-MOEA/D-F (**Sb**, **Ga**), except that one single global model is trained per objective, using all solutions evaluated so far as the training set.

4 EXPERIMENTAL ANALYSIS

4.1 Experimental Setup and Methodology

As mentioned in the introduction, we consider an abstract parallel setting, where we simply assume that some processing units (PUs) are available to operate in parallel *without* targeting a particular computing platform. As such, our results are obtained following a *simulation* methodology. Thereby, we assume that each evaluation takes the same amount of CPU-intensive time, and we ignore the waiting and communication time between PUs, implying that all parallel algorithms are to be considered as operating synchronously in rounds or iterations. This is motivated by the hypothesis that in an expensive setting, objective evaluation time and learning time

Table 2: 8 considered bi-objective benchmark functions.

objective 1			objective 2			
			f_1	f_8	f_{14}	f_{20}
f_1	Sphere	separable	✓	✓	✓	✓
f_8	Rosenbrock	moderate		✓	✓	✓
f_{14}	Sum of powers	ill-conditioned				✓
f_{20}	Schwefel	weakly-structured				

are likely to dominate communication time. This allows us to focus on the quality approximation that can be attained by each designed algorithm when the budget affordable for *each* PU is restricted. In other words, compared to a non parallel setting, where each PU is given an overall budget, our simulation methodology attempt to elicit the impact on approximation quality where, not one but multiple parallel PUs can be afforded the same budget, i.e., we do not look for global speed-up, but for global quality.

For all algorithms, when the initial population size does not depend on the amount of PUs, i.e for all but the MP-MOEA/D variants, it is set to 50 individuals generated by means of Latin Hypercube Sampling [17]. We use the Chebyshev scalarizing function [26] whose reference point is updated at each iteration with the best objective values found so far. Multiple reference points (one per PU) are used when considering local approaches.

For filtering-based approaches, the number of offspring generated for pre-screening is 8 individuals *per* PU, among which the solution with the best predicted scalarizing value for the considered weight is evaluated on the current PU by means of the costly objectives. We also investigate the scalability of the parallel design with respect to the number of PUs available $p \in \{1, 5, 20, 100\}$. All existing algorithms and components are implemented as described in the original papers, including the parameters from MOEA/D [26].

For benchmarking, we rely on problems extracted from the bi-objective black-box optimization test suite `bbob-biobj` [18] Table 2. To observe the impact of problem dimensionality, all functions are considered with $d \in \{2, 5, 10, 20\}$ decision variables. Each algorithm is executed 10 independent times for each function, for a total of 11 520 runs.

Algorithm performance is computed under different scenarios in terms of budget, defined as the amount of evaluations performed *per PU*, from a low budget of 250 evaluations to a relatively high budget (for expensive MOPs) of 1 250 evaluations. The algorithms are compared in terms of hypervolume and hypervolume relative deviation w.r.t. the best-found approximation set.

4.2 Overall Algorithm Performance

Our results are summarized in Table 3, considering three main settings from a large panel of scenarios. We first differentiate the runs based on the global budget: an extremely tight budget of 250 evaluations, and a more moderate one of 1 250 evaluations *per* PU. We consider three different amount of available PUs, ranging from 5 through 20 up to 100. Finally, we consider two function dimensions: 2 and 10. Given that nine algorithms are compared, each algorithm is assigned a rank from 0 to 8, based on pairwise statistical testing. A lower rank is better and can be interpreted as the number of competing algorithms that significantly outperform the considered one

Table 3: Comparison of the competing algorithms with respect to for 250 (left) and 1 250 (right) calls to the evaluation function. The rank stands for the number of algorithms that statistically outperform the one under consideration w.r.t a Mann-Whitney test with a p-value of 0.05 and a Bonferroni correction (lower is better). Bold values correspond to the best algorithm for the problem under consideration.

		MOEA/D-Multistart	S-MOEA/D-QEI	S-MOEA/D-MEI	S-MOEA/D-LocalBatch	S-MOEA/D-GlobalBatch	S-MOEA/D-Multimodel	MP-S-MOEA/D-W	MP-S-MOEA/D-F	MP-S-MOEA/D-G	MOEA/D-Multistart	S-MOEA/D-QEI	S-MOEA/D-MEI	S-MOEA/D-LocalBatch	S-MOEA/D-GlobalBatch	S-MOEA/D-Multimodel	MP-S-MOEA/D-W	MP-S-MOEA/D-F	MP-S-MOEA/D-G
d = 2		#eval = 250									#eval = 1250								
#PUs = 5	f1-f1	5	**0**	**0**	2	2	2	2	2	2	2	1	1	**0**	**0**	**0**	1	1	1
	f1-f8	7	**0**	1	2	2	2	2	2	2	3	2	2	**0**	**0**	**0**	**0**	**0**	**0**
	f1-f14	6	**0**	**0**	3	3	3	2	3	3	3	**0**	1	1	**0**	**0**	**0**	**0**	**0**
	f1-f20	6	**0**	1	2	1	3	2	2	2	4	2	1	1	**0**	**0**	**0**	**0**	**0**
	f8-f8	7	**0**	1	2	1	2	2	2	3	4	1	2	**0**	1	1	**0**	0	1
	f8-f14	7	1	**0**	2	2	1	2	2	3	5	3	4	2	1	2	**0**	**0**	1
	f8-f20	7	**0**	1	2	1	1	1	2	3	5	3	2	2	1	2	**0**	**0**	**0**
	f14-f20	6	**0**	1	2	2	1	3	3	3	4	1	1	1	**0**	1	**0**	**0**	**0**
#PUs = 20	f1-f1	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	1
	f1-f8	5	**0**	**0**	1	1	2	1	2	2	2	1	1	**0**	**0**	**0**	**0**	**0**	**0**
	f1-f14	6	**0**	**0**	3	3	1	2	2	3	2	1	1	**0**	**0**	1	**0**	**0**	1
	f1-f20	4	**0**	1	4	3	2	3	3	4	4	3	2	**0**	2	**0**	**0**	1	1
	f8-f8	2	**0**	**0**	2	2	1	1	1	2	4	2	3	**0**	1	**0**	**0**	**0**	2
	f8-f14	5	**0**	1	3	3	1	3	3	4	4	2	2	1	1	1	**0**	**0**	**0**
	f8-f20	4	**0**	**0**	2	2	1	2	2	3	4	2	2	**0**	1	**0**	**0**	**0**	1
	f14-f20	4	**0**	1	3	2	1	2	2	2	3	1	1	**0**	**0**	**0**	**0**	1	1
#PUs = 100	f1-f1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	f1-f8	3	**0**	**0**	1	**0**	**0**	**0**	**0**	1	1	**0**	**0**	**0**	**0**	0	**0**	**0**	**0**
	f1-f14	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	f1-f20	1	**0**	**0**	**0**	1	**0**	**0**	1	2	4	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**
	f8-f8	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	f8-f14	3	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	1	1	2	1	**0**	**0**	**0**	**0**	1
	f8-f20	4	**0**	1	1	2	**0**	1	1	2	3	1	1	**0**	1	**0**	**0**	**0**	1
	f14-f20	3	**0**	**0**	1	2	1	2	3	2	3	1	1	**0**	**0**	**0**	**0**	**0**	**0**
d = 10		#eval = 250									#eval = 1250								
#PUs = 5	f1-f1	5	**0**	**0**	3	4	3	2	3	3	8	5	6	2	2	4	**0**	**0**	2
	f1-f8	6	**0**	**0**	3	3	3	2	2	3	7	5	5	2	2	3	1	**0**	1
	f1-f14	4	1	1	4	3	3	2	2	3	8	4	5	**0**	**0**	1	**0**	**0**	1
	f1-f20	3	1	1	3	4	**0**	3	4	4	8	5	6	1	1	**0**	**0**	1	1
	f8-f8	5	**0**	**0**	2	3	2	2	2	3	6	3	3	2	2	3	1	**0**	1
	f8-f14	4	**0**	**0**	4	4	3	2	3	4	8	5	4	2	2	2	**0**	**0**	2
	f8-f20	6	**0**	1	3	3	1	3	3	4	8	4	5	3	2	1	**0**	2	2
	f14-f20	4	1	1	3	4	**0**	3	4	4	8	3	5	1	1	**0**	**0**	**0**	1
#PUs = 20	f1-f1	5	**0**	**0**	2	2	3	1	2	2	6	4	4	2	1	3	**0**	**0**	1
	f1-f8	6	**0**	**0**	2	2	2	2	2	3	8	3	4	2	2	3	**0**	1	1
	f1-f14	8	**0**	**0**	3	4	4	2	2	3	7	5	5	1	2	3	**0**	**0**	1
	f1-f20	8	**0**	1	4	4	2	2	3	3	8	4	3	2	2	1	**0**	**0**	**0**
	f8-f8	6	**0**	**0**	2	2	2	1	1	2	6	2	3	**0**	**0**	1	**0**	**0**	1
	f8-f14	8	1	**0**	5	5	2	2	3	3	8	4	4	2	1	3	**0**	1	2
	f8-f20	8	**0**	2	4	5	1	3	3	4	8	3	4	3	3	**0**	1	1	2
	f14-f20	8	1	1	3	3	**0**	3	3	3	8	4	4	2	3	1	1	**0**	1
#PUs = 100	f1-f1	4	**0**	**0**	2	2	2	**0**	**0**	1	8	1	1	**0**	**0**	1	**0**	**0**	**0**
	f1-f8	5	**0**	**0**	3	3	4	1	2	3	8	3	2	2	2	3	**0**	**0**	1
	f1-f14	7	**0**	**0**	3	3	3	2	3	3	8	2	3	2	1	1	**0**	1	1
	f1-f20	8	1	**0**	3	3	2	3	3	3	8	3	2	2	2	**0**	**0**	1	1
	f8-f8	8	**0**	**0**	2	4	3	**0**	**0**	2	8	**0**	4	2	1	0	**0**	**0**	1
	f8-f14	8	**0**	1	2	3	3	1	1	2	8	4	4	2	2	2	1	**0**	1
	f8-f20	7	**0**	**0**	3	3	3	0	2	2	8	4	3	3	3	**0**	**0**	**0**	1
	f14-f20	8	1	2	3	3	**0**	2	1	3	8	3	4	2	2	**0**	1	**0**	1

for a given scenario. A multistart version of the original MOEA/D algorithm is used as a baseline for comparison purposes. We note first that all the strategies obtain a better rank than this multistart approach, which does not use any surrogate, independently of the parameters considered. Overall, we found that the approaches based on EGO (S-MOEA/D-QEI, S-MOEA/D-MEI) outperform all the others when considering an extremely tight budget, while the remaining filtering-based algorithms obtained similar ranks. By contrast, this observation does not hold when considering a larger budget, as both the rank of EGO-based algorithms increases and the different MP-SMOEA/D versions obtain a better average ranking than every other considered algorithm.

4.3 Clustering and Local vs Global Models

When considering parallel S-MOEA/D approaches, one should understand that the construction of the models depend on the solutions made available to the PU(s) that will be responsible for

the model training. Our comparison here focuses on three algorithms: MP-S-MOEA/D-G, where no clustering is performed and where the training set is made available on a master PU that builds a global model for each objective, MP-S-MOEA/D-F, where local models are built using fuzzy clustering [27], and MP-S-MOEA/D-W, where models are built locally and distributively using the current data available at each PU. This data clustering is actually similar to the one introduced in [10]. As shown in Table 3, we observe a clear difference in ranking within the different techniques, independently of the amount of PUs, the benchmark problem, and the budget. Overall, we see that the two approaches relying on clustering outperform the one building global (non-clustered) models. This kind of results where already highlighted for non-parallel approaches [2], but extending this to the parallel case has a seemingly important implication. In fact, as model building is a critical and computationally-intensive step for any S-MOEA/D, this emphasizes the viability of a distributed approach, where models are built locally in a parallel fashion such as in MP-S-MOEA/D-W.

4.4 Empirical Attainment Functions

Empirical attainment functions (EAF) [16] are particularly relevant when comparing different algorithms in order to grasp where an algorithm performs better than another one in the objective space. In this respect, they complement the hypervolume indicator, which alone does not always perfectly express the relative behavior of algorithms. The EAF provides the empirical probability distribution that an arbitrary objective vector is weakly dominated by a solution obtained by a single run. The difference between the EAFs for two different algorithms enables to identify the regions of the objective space where one algorithm outperforms the another. The magnitude of the difference in favor of one algorithm is plotted within a gray-colored graduation. While S-MOEA/D-Multimodel performs overall very similarly to S-MOEA/D-LocalBatch and S-MOEA/D-GlobalBatch, we can see a specific trend when considering problems using function f_{20} (Schwefel). For those problems, S-MOEA/D-Multimodel performs better in comparison to S-MOEA/D-LocalBatch. However, looking at Fig. 1, we clearly see that the multi-model approach is able to find a large amount of good-quality solutions toward objective f_{20}, that were not found with the other approach, while still covering relatively well the other area of the Pareto front. One hypothesis is that one model kernels, among multiple ones, is particularly well-suited for fitting objective f_{20} in particular. Another possible reason is that one kernel or model is actually performing better overall, and not specifically for objective f_{20}. In Fig. 1, we show similar EAF surfaces, but instead of using the described S-MOEA/D-Multimodel approach, we use an algorithm variant, S-MOEA/D-BestModel. The algorithm only uses the model (KRR) that was found to perform better at fitting f_{20}. The results are clearly worst compared to the multi-model technique, as a lot of solutions spanning the first objective are now missing. This allows us to confirm the initial assumption of having different kernels being well suited to fit different functions simultaneously in parallel.

In Fig. 2, we show the EAF difference between the best-performing filtering-based algorithm based on batch evaluation (S-MOEA/D-LocalBatch) and the best performing fully-distributed algorithm, MP-S-MOEA/D-W. The results are shown for problem f8_f14 with

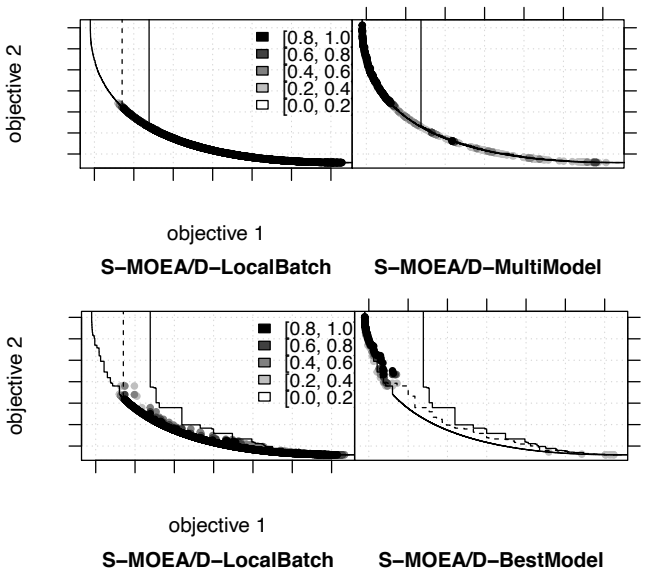

Figure 1: EAF on problem f1_f20 for S-MOEA/D-LocalBatch vs S-MOEA/D-Multimodel. d = 10, PUs = 100, eval = 1250

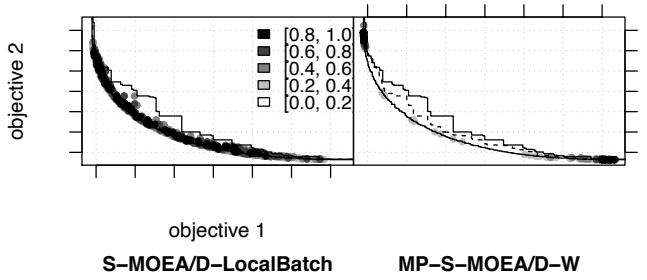

Figure 2: EAF on problem f8_f14 for S-MOEA/D-LocalBatch vs MP-S-MOEA/DW. d = 10, PUs = 100, eval = 1250

$d = 10$ and 100 PUs and a budget of 1250 evaluations per PU. Let us remind first that, as shown in Table 3, MP-S-MOEA/D-W obtains a better average ranking than S-MOEA/D-LocalBatch. On the one hand, We see in Fig. 2 that MP-S-MOEA/D-W is able to reach on the boundaries of the Pareto front that are not reached by S-MOEA/D-LocalBatch, while still covering the other regions of the Pareto front to a certain degree. On the other hand, we see that S-MOEA/D-LocalBatch focuses heavily on exploiting solutions converging toward the center of the Pareto front.

4.5 Convergence Profile and Scalability

When dealing with expensive problems, the convergence profile and thus the anytime performance becomes a critical issue. In Fig. 3, we show the convergence profile for S-MOEA/D-QEI, S-MOEA/D-LocalBatch and MP-S-MOEAD-W, respectively the best EGO-based, *single sub-problem* filtering-based, and *all sub-problems* filtering-based algorithms. Results are shown on a supposedly *easy* function (f1_f1), and a supposedly *difficult* function (f14_f20) with a dimension of $d = 10$, and using 5, 20 and 100 PUs. In these graphs,

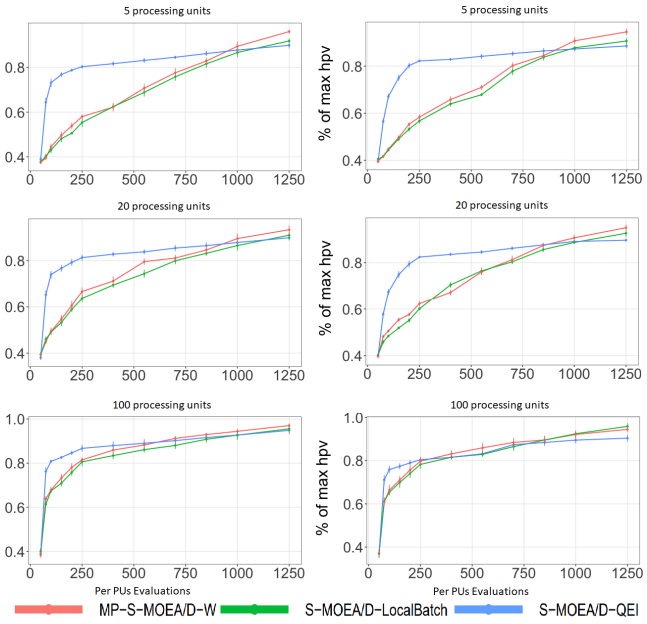

Figure 3: Convergence profile on problems f1_f1 (left) and f14_f20 (right) for different PUs, budgets and for $d = 10$.

note that the maximum hypervolume is different depending on the number of PUs, and the convergence profiles observed were similar for other problems and dimensions. Even though the convergence rate varies when scaling the amount of PUs, the profiles remain the same. On the one hand, we see that EGO-based approaches converge extremely fast, and are often stuck after a number of evaluations per PU ranging from 100 to 250. On the other hand, we see that filtering-based approaches have a similar convergence profile, starting slower than the EGO-based ones, and ultimately ending up outperforming them when the budget grows around a thousand evaluations. In terms of scalability, we show in Table 4 the amount of evaluations per PU needed to reach 60%, 80% and 95% of the maximum hypervolume value for the same three algorithms, using 5 and 100 PUs. While the performance of S-MOEA/D-QEI gets better, the difference in results between 5 and 100 PUs is mitigated. The first threshold of 60% is already reached on average for the lowest amount of evaluation considered. The second treshold of 80% is reached 150 evaluations quicker on average. The final treshold of 95% was never reached when using 5 PUs, it was reached 3 times with 100 PUs but was not reached before 1 250 evaluations on the average run. For both filtering approaches, the results are quite different: the number of evaluations needed to reach the first two thresholds decreases substantially from 400 evaluations to 75 evaluations for the first threshold and from 1 000 evaluations to 250 for the second. The hardest threshold of 95% is almost never hit for 5 PUs with S-MOEA/D-QEI, whereas it is hit 9 out of 10 times for S-MOEA/D-LocalBatch and in all runs for MP-S-MOEA/D-W. This clearly indicates that, depending on how restricted the budget is per PU, and how many PUs are available to operate in parallel, different design options of seemingly different nature are to be preferred.

Table 4: Bound on the maximum budget to hit a threshold percentage T of best hypervolume. The parenthesis indicate the ratio of runs where this threshold was hit with the indicated budget. The value ∞ indicates that the thredshold is never hit by any run.

# PUs	T	S-MOEA/D-QEI		S-MOEA/D-LocalBatch		MP-S-MOEA/D-W	
5	60%	75	(8/10)	400	(7/10)	400	(9/10)
	80%	400	(9/10)	1000	(6/10)	1000	(7/10)
	95%	∞	(0/10)	1250	(1/10)	1250	(2/10)
100	60%	75	(10/10)	75	(9/10)	75	(9/10)
	80%	250	(10/10)	250	(7/10)	250	(9/10)
	95%	1250	(3/10)	1250	(9/10)	1250	(10/10)

5 CONCLUSIONS

In this paper, we investigated parallel surrogate-assisted MOEAs based on decomposition for expensive multi-objective optimization. Following an extensive research on the current state-of-the art for parallel MOEA/D, surrogate-assisted MOEA/D and the combination of both, we were able to differentiate and instantiate existing approaches based on key parallel algorithm components. By combining components from non-parallel state-of-the-art algorithms, we built a set of representative parallel algorithms that are subsequently analyzed and benchmarked. Our analysis reveals the following findings. On the one hand, it was shown that the entire process of model building highly impacts the results of the different algorithms. Firstly, local data selection for training, clustering, and multiple model building were found to perform competitively, while offering a natural degree of parallelism compared to approaches based on a centralized model building using the full available data. Secondly, building models based on the aggregation of the objective functions showed to yield better results than model learning the objectives directly. Thirdly, model type and kernel selection have a high impact on algorithm performance. In particular, further investigations focusing on model selection is worth to be considered in the future, and is expected to lead to an improved design and performance. On the other hand, for filtering-based approaches, we found that algorithms evaluating in parallel a pool of solutions based on the entire set of sub-problems provide better results than algorithms performing parallel evaluations of a pool targeting solely one sub-problem. Interestingly, this does not hold when considering Gaussian process-based (EGO) approaches. At last, we were able to confirm that the convergence profile of Gaussian process-assisted approaches is appealing when only an extremely restricted budget is allowed. By contrast, when scaling the whole parallel system, the superiority of EGO-based approaches has a tendency to decrease, compared to filtering-based methods. This means that for an increasing number of PUs, and an increasing budget per PU, relatively simple evolutionary filtering methods can outperform pure EGO methods while being flexible and easy to deploy.

ACKNOWLEDGMENTS

This work was supported by the French national research agency (ANR-16-CE23-0013-01) and the Research Grants Council of Hong Kong (RGC Project No. A-CityU101/16).

REFERENCES

[1] Taimoor Akhtar and Christine Shoemaker. [n.d.]. Efficient Multi-Objective Optimization through Population-based Parallel Surrogate Search.

[2] Nicolas Berveglieri, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, and Kiyoshi Tanaka. 2019. Surrogate-Assisted Multiobjective Optimization Based on Decomposition: A Comprehensive Comparative Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 507–515. https://doi.org/10.1145/3321707.3321836

[3] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido. 2019. A Taxonomy for Metamodeling Frameworks for Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation* 23, 1 (Feb 2019), 104–116. https://doi.org/10.1109/TEVC.2018.2828091

[4] B. Derbel, A. Liefooghe, G. Marquet, and E. Talbi. 2015. A fine-grained message passing MOEA/D. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. 1837–1844. https://doi.org/10.1109/CEC.2015.7257110

[5] Alan Diaz-Manriquez, Gregorio Toscano Pulido, Jose Barron-Zambrano, and Edgar Tello-Leal. 2016. A Review of Surrogate Assisted Multiobjective Evolutionary Algorithms. *Computational Intelligence and Neuroscience* 2016 (06 2016), 1–14. https://doi.org/10.1155/2016/9420460

[6] Juan J. Durillo, Qingfu Zhang, Antonio J. Nebro, and Enrique Alba. 2011. Distribution of Computational Effort in Parallel MOEA/D. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION'05)*. Springer-Verlag, Berlin, Heidelberg, 488–502. https://doi.org/10.1007/978-3-642-25566-3_38

[7] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. 2010. *Kriging Is Well-Suited to Parallelize Optimization.* Springer Berlin Heidelberg, Berlin, Heidelberg, 131–162. https://doi.org/10.1007/978-3-642-10701-6_6

[8] Raphael T. Haftka, Diane Villanueva, and Anirban Chaudhuri. 2016. Parallel surrogate-assisted global optimization with expensive functions – a survey. *Structural and Multidisciplinary Optimization* 54, 1 (01 Jul 2016), 3–13. https://doi.org/10.1007/s00158-016-1432-3

[9] Daniel Horn, Tobias Wagner, Dirk Biermann, Claus Weihs, and Bernd Bischl. 2015. Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark. In *Evolutionary Multi-Criterion Optimization*, António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello (Eds.). Springer International Publishing, Cham, 64–78.

[10] Rayan Hussein and Kalyanmoy Deb. 2016. A Generative Kriging Surrogate Model for Constrained and Unconstrained Multi-objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, New York, NY, USA, 573–580. https://doi.org/10.1145/2908812.2908866

[11] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1 (2011), 61–70.

[12] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13, 4 (01 Dec 1998), 455–492. https://doi.org/10.1023/A:1008306431147

[13] J. Knowles. 2006. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10, 1 (Feb 2006), 50–66. https://doi.org/10.1109/TEVC.2005.851274

[14] Yung Siang Liau, Kay Chen Tan, Jun Hu, Xin Qiu, and Sen Bong Gee. 2013. Machine Learning Enhanced Multi-Objective Evolutionary Algorithm Based on Decomposition. In *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 553–560.

[15] Lakhdar Loukil, Malika Mehdi, Nouredine Melab, El-Ghazali Talbi, and Pascal Bouvry. 2009. A Parallel Hybrid Genetic Algorithm-Simulated Annealing for Solving Q3AP on Computational Grid. In *Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing (IPDPS '09)*. IEEE Computer Society, USA, 1–8. https://doi.org/10.1109/IPDPS.2009.5161126

[16] Manuel López-Ibáñez, Luis Paquete, and Thomas Stützle. 2010. Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization. In *Experimental Methods for the Analysis of Optimization Algorithms*, Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss (Eds.). Springer, Berlin, Germany, 209–222. https://doi.org/10.1007/978-3-642-02538-9_9

[17] Michael D. McKay. 1992. Latin Hypercube Sampling as a Tool in Uncertainty Analysis of Computer Models. In *Proceedings of the 24th Conference on Winter Simulation (WSC '92)*. Association for Computing Machinery, New York, NY, USA, 557–564. https://doi.org/10.1145/167293.167637

[18] O. Mersmann; T. Tušar; N. Hansen; A. Auger and D. Brockhoff. 2016. COCO: A platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* (2016).

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[20] Witold Pedrycz and Dirk Sudholt. 2013. Parallel Evolutionary Algorithms Chapter in the Handbook of Computational Intelligence.

[21] Frederik Rehbach, Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein. 2018. Comparison of Parallel Surrogate-Assisted Optimization Approaches. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 1348–1355. https://doi.org/10.1145/3205455.3205587

[22] Schonlau, Matthias. 1997. Computer experiments and global optimization. http://hdl.handle.net/10012/190

[23] Anna Syberfeldt, Henrik Grimm, Amos Ng, and Robert John. 2008. A Parallel Surrogate-Assisted Multi-Objective Evolutionary Algorithm for Computationally Expensive Optimization Problems. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 3177 – 3184. https://doi.org/10.1109/CEC.2008.4631228

[24] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh. 2017. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 21, 3 (June 2017), 440–462. https://doi.org/10.1109/TEVC.2016.2608507

[25] Saúl Zapotecas Martínez and Carlos A. Coello Coello. 2013. MOEA/D Assisted by Rbf Networks for Expensive Multi-Objective Optimization Problems. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. Association for Computing Machinery, New York, NY, USA, 1405–1412. https://doi.org/10.1145/2463372.2465805

[26] Q. Zhang and H. Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (Dec 2007), 712–731. https://doi.org/10.1109/TEVC.2007.892759

[27] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. 2010. Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model. *IEEE Transactions on Evolutionary Computation* 14, 3 (June 2010), 456–474. https://doi.org/10.1109/TEVC.2009.2033671