On the Design of a Master-Worker Adaptive Algorithm Selection Framework

Christopher Jankee, Sébastien Verel, Bilel Derbel, Cyril Fonlupt

On the Design of a Master-Worker Adaptive Algorithm Selection Framework

Christopher Jankee[1], Sébastien Verel[1] Bilel Derbel[2], and Cyril Fonlupt[1]

[1] Université du Littoral Côte d'Opale, LISIC
[2] Université Lille 1, LIFL – CNRS – INRIA Lille

Abstract. We investigate the design of a master-worker schemes for adaptive algorithm selection with the following two-fold goal: (i) choose accurately from a given portfolio a set of operators to be executed in parallel, and consequently (ii) take full advantage of the compute power offered by the underlying distributed environment. In fact, it is still an open issue to design online distributed strategies that are able to optimally assign operators to parallel compute resources when distributively solving a given optimization problem. In our proposed framework, we adopt a reward-based perspective and investigate at what extent the average or maximum rewards collected at the master from the workers are appropriate. Moreover, we investigate the design of both homogeneous and heterogeneous scheme. Our comprehensive experimental study, conducted through a simulation-based methodology and using a recently proposed benchmark family for adaptive algorithm selection, reveals the accuracy of the proposed framework while providing new insights on the performance of distributed adaptive optimization algorithms.

1 Introduction

The selection of an accurate algorithm from a given portfolio, as well as the effective choice of the relevant algorithmic components of a general-purpose search heuristic, are among the major issues that one has to face in practice when tackling an optimization problem; in particular, in a black-box optimization scenario when no problem-specific properties can be known beforehand [3]. In fact, from a theoretical point of view, several parallel compute ressources, possibly distributed over a large scale environment, are provided, it is even more challenging to design an efficient distributed cooperative strategy, since the algorithmic design space gets huge and we still lack knowledge on the optimal mapping of the implied search computational flows to the available resources. The motivation of this paper is precisely to investigate these issues by proposing a master-worker algorithm selection framework and precisely analyzing the impact of its different possible design components. On the one hand, algorithm selection (or the related topic of parameter setting), although being one of the oldest research topic in evolutionary computation [14], is attracting more and more attention [17] due to its crucial importance and the difficult, and yet unsolved, challenges it implies

in practice. In this work, we are interested in *adaptive algorithm selection*. Indeed, there are two main and tightly related methodologies that are commonly adopted to select an algorithm [9]. In the offline setting, usually called *tuning*, an algorithm is first selected, and only then it is executed from scratch on the target and unseen problem instance. In the online setting, called *control*, an algorithm is selected all along the optimization process (see for example [11, 3]). An online selection scheme is typically and continuously getting feedback from the optimization algorithm being executed, and deciding accordingly on the next choice. Hence, online algorithm selection can be viewed as an adaptive optimization algorithm which follows the multi-armed bandit framework where the arms are the algorithms of the portfolio [5]. The adaptive algorithm selection is then performed as follows. A reward is computed according to the performance observed when previously executing an algorithm. Then, in every iteration, a reinforcement machine learning is applied in order to select from the portfolio the next algorithm to execute, typically according to some exploration-exploitation rules.

On the other hand, numerous real wold optimization problems, such as engineering design which are often based on numerical simulation, are computationally expensive, e.g., one fitness function evaluation can take several minutes [19]. Besides, the advent of new compute facilities and the establishment of robust and large scale massively parallel platforms, such as grids and pay-as-you-go clouds, open tremendous research opportunities for pushing forward the development and uptake of parallel and distributed evolutionary optimization algorithms. In this context, a number of evolutionary optimization models have been investigated [20], e.g., from centralized to fully decentralized, from finedgrained (cellular model) to coarse grained (island model). In this work, we adopt the centralized Master-Worker (M/W) architecture, where each worker process is basically responsible of executing locally in parallel the actions scheduled for him by the master (e.g., evaluate a candidate solution), whereas the master process is responsible of collecting the local results from the workers (e.g., the fitness values) and deciding on the next actions to send them (e.g., next candidate solutions to evaluate). It is worth-noticing that this framework is often adopted in practice, not only due to the simplicity of deploying it over a real test-bed, but also due to its high accuracy when dealing with computationally expensive optimization problems [6].

In this context, we argue that a master-worker approach to adaptive algorithm selection requires specific coordination mechanisms in order to achieve optimal performances. In a sequential setting, the observed rewards are in fact updated according to the performance of the algorithm executed previously in the last round by one single process. In a M/W approach, one can benefit from the set of performances observed by several parallel processes, i.e., the workers. However, switching to such a scenario requires to carefully define the aggregated reward with respect to a selected algorithm given a set of observed performance values instead of just a single one. Additionally, one can adopt either a homogenous strategy in which all workers execute the same algorithm at each iteration (e.g., the best rewarded one so-far) or instead a heterogeneous strategy where

the workers can execute different algorithms. Several existing machine learning technics have previously been used and studied in the sequential setting [11], as well as in the decentralized island model [7, 15]. However, to our best knowledge, the design and analysis of online selection strategies have not been investigated within a M/W framework. We argue that the M/W scheme make it more convenient, as a first step, to reason about the optimal distributed decisions to make since the master has the ability of acquiring a global view of the whole distributed system before selecting the most accurate algorithms to execute in parallel by the workers. This allows us to focus on the critically important selection strategy at the master level. To summarize, we propose a M/W algorithm selection framework contributing to the solving of the following questions:

- How to define a reward function on the master based on the performance of the algorithm(s) executed by the workers?
- How the master can decide on the set of algorithms to be executed next by the workers based on the reward function?
- What is the relative quality that can be achieved by different algorithm selection strategies ?

Our M/W framework is evaluated using a tunable benchmark family and a simulation-based experimental procedure in order to abstract away the technical implementation issues, and instead provide a fundamental and comprehensive analysis of the expected empirical parallel performance of the underlying adaptive algorithm selection. The rest of the paper is organized as follows. In Section 2, we review some related works. In Section 3, the design components of our M/W adaptive framework is described in details. In Section 4, we report our main experimental findings. In Section 5, we conclude the paper and discuss future research directions.

2 Related works

In the following, we provide an overview of related studies on the algorithm selection problem in the sequential and distributed setting, as well as a brief summary of exiting optimization benchmark problems designed at the aim of evaluating their dynamics and behavior.

2.1 Sequential Adaptive Algorithm Selection

In the sequential setting, a number of reinforcement machine learning technics have been proposed for the online and adaptive selection of algorithms from a given portfolio. Back to the early works of Grefenstette [14], one standard technique consists in predicting the performance of a set of operators using a simple linear regression and the current average fitness of the population, which then allows to select the best operator to be chosen according to the prediction given by the regression. However, recent works embeds this selection

problem into a multi-armed bandit framework dealing more explicitly with the tradeoff between the exploitation of the best so far identified algorithm, and the exploration of the remaining potentially under-estimated algorithms.

A simple strategy is the so-called ϵ-greedy (ϵ-G) strategy which consists in selecting the algorithm with the best estimated performance at rate $(1 - \epsilon)$, and a random one at rate ϵ. In that case, the performance of an operator i is estimated with the empirical mean $\hat{\mu}_i$ of rewards on a sliding window where only the W previous reward observations are considered. The Upper Confidence Bound (UCB) strategy [2] is a state-of-the-art framework in machine-learning which consists in estimating the upper confidence bound of the expected reward of each arm by $\hat{\mu}_i + C \cdot e_i$; where $\hat{\mu}_i$ is the estimated (empirical) mean reward, and e_i is the standard error of the prediction. It then selects the algorithms with the higher bound (for maximization problem). The parameter C allows to tune the exploitation/exploration trade-off. In the context of algorithm selection [11] where the arms could be neither independent nor stationary, the estimation of the expected reward is refined using a sliding window of size W. The Adaptive Pursuit (AP) strategy [22] is another technique using an exponential recency weighted average to estimate the expected reward with a parameter α to tune the adaptation rate of the estimation. This is used to define the probability p_i of selecting every algorithm from the portfolio. At each iteration, these probability values are updated according to a learning rate β, which basically allows to increase the selection probability for the best algorithm, and to decrease it for the other ones.

One key aspect to design a successful adaptive selection strategy is the estimation of the quality of an algorithm based on the observed rewards. Some authors showed that the maximum reward over a sliding window improves the performance compared to the mean on some combinatorial problems [11, 4]; but no fundamental analysis of this result was given. In genetic algorithms, the reward can be computed not only based on the quality but also on the diversity of the population [18]. In the context of parallel adaptive algorithm selection, the estimation of quality of each available algorithm is also a difficult question since not only one but many algorithms instances could be executed in each iteration.

2.2 Parallel Adaptive Algorithm Selection

The Master-Worker (M/W) architecture has been extensively studied in evolutionary computation (e.g., see [8]). It is in fact simple to implement, and does not require sophisticated parallel operations. Two communication modes are usually considered. In the synchronous mode, the distributed entities operate in rounds, where in each round the master communicates actions to the workers and then waits until receiving a response from every worker before starting a new round, and so on. In the asynchronous mode, the master does not need to wait for all workers; but instead can initiate a new communication with a worker, typically when that worker has terminated executing the previous action and is idle. When the evaluation time of the fitness function can vary substantially during the course of execution, the asynchronous mode is generally preferred [24] since

it can substantially improve parallel efficiency. However, the synchronous mode can allow to have a more global view of the distributed system which can be crucially important to better coordinate the workers [23].

Adaptive selection approaches designed to operate in a distributed setting are not new. The island model, which is considered as inherently distributed, has been investigated in the past. To cite a few, in [21, 12], it is also demonstrated that a randomly setting the parameters at each iteration in a heterogeneous manner can outperforms static homogeneous parameter settings. Nonetheless, embedding a reinforcement machine learning technique instead of random selection can improve the performance of the adaptive distributed system. In [4], a dynamic island model is proposed to select online the relevant algorithm. Each island is associated to one algorithm, and the migration rates of solutions between islands are controlled by the operators performance of each island. As commented by the authors, this technique is not designed to fit directly in a scalable distributed system and requires some further adaptations. In [7, 15], a distributed adaptive metaheuristic selection framework is proposed which can be viewed as a natural extension of the island model that was specifically designed to fit the distributed nature of the target compute platforms. The adaptive selection is performed locally by selecting the best rewarded metaheuristic from the neighboring nodes (islands) or a random one with small probability like in ϵ-greedy strategy. Notice however that we are not aware of any in-depth analysis addressing the design principles underlying a M/W adaptive algorithm selection approach. In this work, we propose and empirically analyze the behavior of such an approach in an attempt to fill the gap between the existing sequential algorithm selection methods and the possibility to deploy them in a parallel compute environment using a simple, yet effective, parallel scheme like the M/W one.

2.3 Benchmarks: The Fitness Cloud Model

The understanding of the dynamics of a selection strategy according to the problem at hand is a difficult issue. A number of artificial combinatorial problems have been designed and used in the literature. We can distinguish between two main benchmark classes. In the first one, a well-known combinatorial problem in evolutionary algorithm is used, such as oneMax or long-path problems, with basic operators, such as bit-flip, embedded in a $(1 + \lambda)$-EA [5]. This however can only highlight the search behavior according to few and problem-specific properties. In the second class of benchmarks, the problem and the stochastic operators are abstracted. The performance of each available operator is then defined according to the state of the search [22, 11, 13, 16]. This allows to study important black-box (problem independent) features such as the number of operators, the frequency of change of the best operators, the quality difference between operators, etc.

In this work, we use a tunable benchmark, called the Fitness Cloud Model (FCM), introduced recently in [16]. The FCM is a benchmark from the second class where the state of the search is given by the fitness of the solution. The fitness of a solution after applying a search operator is modeled by a random variable for which the probability distribution depends on the fitness of the

current solution. A normal distribution with tunable parameters is typically used. More specifically, given the fitness $z = f(x)$ of the current solution x, the probability distribution of the fitness $f(y)$ of one solution obtained by a specific operator is defined by: $\Pr(f(y) = z' \mid f(x) = z) \sim \mathcal{N}(\mu(z), \sigma^2(z))$ where $\mu(z)$ and $\sigma^2(z)$ are respectively the mean and the variance of the normal distribution. In [16], a simple scenario with two operators is studied. The mean and variance of the conditional normal distribution are defined as follows: $\mu_i(z) = z + K_{\mu_i}$ and $\sigma_i^2(z) = K_{\sigma_i}$ for each operator $i \in \{1, 2\}$. Parameters K_{μ_i} and K_{σ_i} are different constant numbers. An adaptive algorithm is assumed to start with a search state where the fitness value is 0, and stops when a fitness value of 1 is reached. Notice that in the FCM, on the contrary of benchmark of the first class (oneMax, etc.), one can control the average quality and the variance of each operator as well the relative difference between the considered operators which are two of the main features to analyze from the perspective of adaptive selection of operators. Please refer to [16] for more details on the design and motivation of the FMC benchmark.

3 M/W Framework Description

First, a portfolio of k (local search) operators is assumed to be given, and no a priori knowledge is assumed on the behavior of the operators with respect to the black-box problem under consideration. Naturally, k is an integer value greater or equal than 2. The global architecture of the proposed adaptive M/W framework is summarized in Algorithm 1 depicting the high level code executed by the master and in Algorithm 2 depicting the high level code executed in parallel by each worker. The overall algorithm operates in different parallel rounds. At each round, the master sends the best solution $x^\star$ and the operator identifier θ^i assigned to each worker node i. Based on $x^\star$ and θ^i, the role of each worker is to compute a new candidate solution to be send back to master. Although one could consider and study different alternatives, in this work, a standard $(1+1)$-EA is simply executed by each worker. In addition, the worker computes a local reward in order to render the quality of its assigned operator θ^i. Different kinds of local rewards can be considered at this stage [10]. In our work, and since an elitist selection is applied locally by each worker, the local reward of an operator is the positive improvement observed when applying the $(1+1)$-EA. The master waits for all local solutions computed in parallel by the workers, and updates the global best solution $x^\star$ to be considered in the next round, and so on. More importantly, the local rewards collected by the master are used in order to select a new set of operators to be assigned to the workers in the subsequent rounds, which actually constitutes the adaptive and core part of our framework. Two tightly coupled issues are to be handled by the master in order to set up an effective adaptive mechanism: (i) how to aggregate the local rewards sent by the workers and (ii) how to select the new set of operators accordingly. This is described next.

Algorithm 1 Adaptive M/W algorithm for the master node

$(\theta^1, \theta^2, ..., \theta^n) \leftarrow$Selection_Strategy_Initialization()
$x^\star \leftarrow$ Solution_Initialization() ; $f^\star \leftarrow f(x^\star)$
repeat
  for each worker i **do**
    Send $\mathsf{Msg}(\theta^i, x^\star, f^\star)$ to worker i
  end for
  Wait until all messages are received from all workers
  for each worker i **do**
    $(r^i, x^i, f^i) \leftarrow$ Receive $\mathsf{Msg}()$ from worker i
  end for
  $x^\star \leftarrow x^i$; $f^\star \leftarrow f^i$ s.t. $f^i = \max\{f^\star, f^1, f^2, \ldots, f^n\}$
  $(R_1, R_2, ..., R_k) \leftarrow$ Reward_Aggregation$((\theta^1, r^1), ..., (\theta^n, r^n))$
  $(\theta^1, \theta^2, ..., \theta^n) \leftarrow$ Decision_Strategy$(R_1, R_2, ..., R_k)$
until stopping criterion is true

Algorithm 2 Adaptive M/W algorithm for each worker node

$(\theta, x^\star, f^\star) \leftarrow$ Receive Msg() from master
$x' \leftarrow$ Apply operator θ on $x^\star$; $f' \leftarrow$ Evaluate fitness of x'
$\delta_b \leftarrow \max(0, f' - f^\star)$
if $f(x^\star) < f(x')$ **then**
  $x^\star \leftarrow x'$; $f^\star \leftarrow f^{'}$
end if
Send Msg$(\delta_b, x^\star, f^\star)$ to master

3.1 Aggregation of local reward values

On one hand, all adaptive operator selection strategies such as ϵ-greedy, Adaptive Pursuit, Upper Confidence Bound, etc. (see Sec. 2.1) need to get one single reward value as a feedback when one operator is executed. On the other hand, in our framework, a set of local rewards are computed by the workers and provides us with a feedback on the quality of an operator when executed in parallel by several workers. Unlike sequential algorithms, the set of local rewards observed in parallel cannot be viewed simply as a sequence of independent rewards that would be given iteratively to a sequential strategy. Hence, one specific design component of an adaptive M/W algorithm is the way to aggregate the local reward values into one global reward value. Consequently, we distinguish two main aggregation strategies: (i) the mean or the (ii) maximum of the local rewards. In other words, at each round, the (global) reward computed by the master, with respect to one operator executed by at least one worker, is either the average or the maximum of the local values sent by the corresponding workers.

Despite their simplicity, the two previous local reward aggregation strategies are fundamentally different. In fact, assuming that the fitness improvement after applying a stochastic operator is given by a probability distribution, the mean of the reward values computed by the n workers allows to estimate the expectation of this distribution with a high accuracy, whereas the maximum gives information

on its extremes [10]. Additionally, we consider a sliding window of size W to estimate the expected reward $\hat{\mu}_i$ in ϵ-greedy, and UCB as considered in previous works.

3.2 Homogeneous *vs.* Heterogeneous Adaptive Selection

As mentioned previously, the master needs to select one operator for each worker. We consider both (i) a *Homogeneous* (Ho) adaptive strategy, in which the same operator is selected by the master and assigned to all worker, and (ii) a *Heterogeneous* (He) adaptive strategy, in which the master selects, possibly different, operators to be assigned to the workers. The rationale behind a homogeneous strategy is that in each round there exists one relevant operator providing an optimal performance, and hence should be executed simultaneously in parallel by all workers. This a rather exploitation-guided strategy which aims at avoiding to loose function evaluations, and to post-pone the exploration component to act in-between two consecutive rounds. In contrast, the rationale behind a heterogeneous strategy is that a set containing a mixture of different operators is expected to perform better than a set containing the same operator, in the sense that: (i) the probability of obtaining a better solution when executing different operators in each round is larger, and/or (ii) a relatively small number of evaluations spent exploring non-necessarily optimal operator(s) at each round allows to better predict the best operator(s) to select next.

In the homogeneous setting, we consider the three standard selection strategies (cf. Sect. 2.1), namely, ϵ-greedy, AP, and UCB. The same operator computed by any of these strategies is assigned by the master to the workers. Notice that the difference with a sequential selection is the way the reward is computed by the workers and maintained by the master, which is crucially important for those methods to operate accurately. In the heterogeneous setting, we consider to execute either the ϵ-greedy strategy or the AP strategy iteratively for each worker. Notice in fact that these two strategies are randomized, i.e., for ϵ-greedy, the best operator is selected with rate $1 - \epsilon$ and the other ones with rate ϵ, and for AP, each operator is selected proportionally to a rate p_i. Hence, by running iteratively those strategies, the selected operators is likely to be different in each execution and the master is then able to assign different operators to the workers. In contrast, running iteratively an UCB selection does not give an heterogenous strategy due to its deterministic nature (same operator is given at each selection step). Designing an heterogeneous UCB-based strategy is actually a challenging open question which is left for future investigations.

4 Experimental analysis

We consider the Fitness Cloud Model as an abstract benchmark. We have three competing adaptative selection mechanisms (ϵ-G, UCB, and AP) which combined accordingly with the two considered reward aggregation strategies (mean

Table 1. Parameters setting of the selection strategies

Selection strategy	Parameters value
ϵ-G He. Max.	$\epsilon = 0.5$ $W = 400$
ϵ-G He. Mean	$\epsilon = 0.5$ $W = 400$
ϵ-G Ho. Max	$\epsilon = 0.05$ $W = 4500$
ϵ-G Ho. Mean	$\epsilon = 0.05$ $W = 4500$
UCB Max.	$C = 0.005$ $W = 700$
UCB Mean	$C = 0.05$ $W = 5000$
AP	$\alpha = 0.2$ $\beta = 0.2$

and max), and the two homogeneity scenarios (Ho and He), provide us 10 variants. Moreover, we consider two baseline random strategies, which consist in selecting the next operator randomly, both in a homogeneous or in a heterogeneous setting. In the following, we first start discussing the overall relative performance, then provide a more focused analysis to better understand the behavior and the dynamics of the different variants.

4.1 Overall Relative Performance

We adopt a simulation-based approach where we count the number of rounds performed by the master until reaching the optimal fitness value. This allows us to abstract away the communication issues and to evaluate the accuracy of the considered algorithms in adapting the search process to operate optimally. We consider a portfolio with $k = 2$ operators. Following the Fitness Cloud Model, each operator impacts differently the fitness of solution: the first one follows the normal distribution $\mathcal{N}(-10^{-4}, 10^{-4})$, and the second one $\mathcal{N}(-10^{-3}, 5 \times 10^{-4})$. These distributions are fixed and do not change in the course of the optimization which is a simple, yet challenging, scenario in order to elicit the behavior of adaptive algorithms in a black-box scenario. For the sake of presentation, the choice of the benchmark parameters will be discussed later. The parameter set of the different selection strategies is given in Table 1. This setting can be shown to be robust and is in accordance with previous studies [16, 15]. Each variant is executed 100 times and an overview of the performance in terms of number of distributed rounds is given in Fig. 1. Three main observations can be made.

Firstly, using the mean reward aggregation function is clearly outperformed by the maximum reward function. Secondly, the difference between a homogeneous and heterogeneous setting is mitigated and depends on the selection it-self. Thirdly, according to a Mann-Whitney statistical test at confidence level of 5%, and when comparing the best setting of given selection variant, the UCB strategy appears to be the best one, followed by ϵ-greedy strategy and are better than the AP strategy. These first results can be explained by the ability of the UCB machine-learning inspired strategy to efficiently learn the best operator to apply in a given round when using the maximum reward. The other strategies, although being competitive, spend some rounds to explore non-relevant operators. More importantly, all adaptive strategies are found to share a relatively good performance when the other design components, that is the choice of the reward, and the heterogeneity, are well tuned. To better understand such a behavior, we provide next a more throughout analysis.

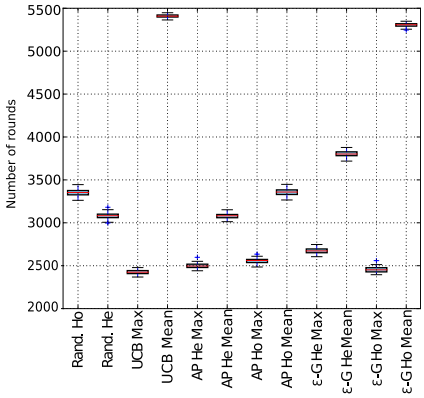

Fig. 1. Number of rounds to the optimal fitness using operator 1 $\sim \mathcal{N}(-10^{-4}, 10^{-4})$, operator 2 $\sim \mathcal{N}(-10^{-3}, 5 \times 10^{-4})$ and $n = 256$ workers.

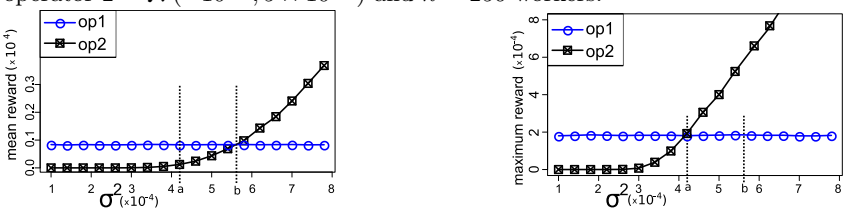

Fig. 2. Mean (top) and maximum (bottom) reward values with operator 1 $\sim \mathcal{N}(-10^{-4}, 10^{-4})$, operator 2 $\sim \mathcal{N}(-10^{-3}, \sigma^2)$ and $n = 256$ workers as a function of the variance σ^2.

4.2 Analysis of the Reward Aggregation Functions

To understand the fundamental difference between using the maximum or the mean as a reward function, as well as its crucial importance when designing an adaptive strategy, we consider to study the property of the considered fitness cloud benchmark in an extended setting. More precisely, let us fix the parameters of the normal distribution corresponding to the first operator in the portfolio to $\mu_1 = -10^{-4}$ and $\sigma_1^2 = 10^{-4}$. Let us also fix the mean of the normal distribution of the second operator to $\mu_1 = -10^{-3}$. Since both means are negative, the fitness value is decreased *in expectation* by both operators. For the fixed number of workers, and since the parameters of the normal law does not change in the course of optimization, operator 1 would always provide the same expected improvement, i.e., 8.33×10^{-2} [15], which corresponds to the local reward. Let us now study how the relative reward value would be for operator 2 if its variance was set to take different values than in our initial setting. This is summarized in Fig. 2 showing the expected rewards of both operators when using a mean aggregation function (top) and a maximum aggregation function (bottom), for $n = 256$ workers as a function of a range of variance values σ^2 for operator 2.

For both reward aggregation functions, the reward value of operator 2 increases with the variance σ^2. Below the value of $a = 4.17 \times 10^{-4}$, the reward of

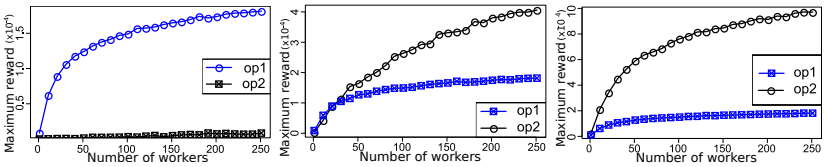

Fig. 3. Maximum reward value as function of the number of workers n for operator $1 \sim \mathcal{N}(-10^{-4}, 10^{-4})$ and different variance values of operator $2 \sim \mathcal{N}(-10^{-3}, \sigma^2)$: $\sigma^2 = 10^{-4}$ (left), $\sigma^2 = 5 \times 10^{-4}$ (middle), and $\sigma^2 = 7 \times 10^{-4}$ (right).

operator 1 is higher than the reward of operator 2 for both mean and maximum functions. Hence, an (elitist) operator selection strategy which selects the operator according to the highest reward value would select the same operator 1, no matter which aggregation function is used, i.e., there is no difference between the two aggregation function in the case the difference between the fitness variances of both operators is relatively large, given that their mean fitness is same. Similarly, when the variance σ^2 of operator is much larger than the variance of operator 1 ($\sigma^2 > b = 5.6 \times 10^{-4}$), the reward value for operator 2 is larger compared to operator 1 no matter the reward aggregation used. Hence, a selection strategy based on one or the other reward function would likely take the same decision, i.e., select operator 2. However, the challenging situation is when the variance σ^2 is in the interval $[a, b]$; since, according to the mean reward function, operator 1 (resp. 2) is better (resp. worst), but according to the maximum reward function, operator 1 (resp. 2) is worst (resp. better). In this case, it is not clear that two selection strategies following the mean and the maximum reward function would select the same operator.

Additionally, the mean reward value does not depend on the number of workers. In fact, increasing the number of local rewards computed by the workers simply reduces the confidence interval around the global mean reward value. In contrast, the maximum reward value increases logarithmically with the number of workers. This is shown in Fig. 3 where three representative examples are considered ($\sigma^2 = 3 \times 10^{-4} < a$, $\sigma^2 = 5 \times 10^{-4} \in [a, b]$, and $\sigma^2 = 7 \times 10^{-4} > b$). When the variance is small or large, the number of workers does not change the rank of each operator with respect to the maximum reward value. However, when the variance σ^2 is between a and b, the best operator according to the maximum reward changes: for low number of workers, operator 1 has a highest maximum reward, whereas operator 2 is preferred when $n \geq 30$.

These empirical observations explain why the maximum reward was found to clearly outperform the mean reward (Sec. 4.1, Fig. 1), since the variance of the second operator was set to a the value $5 \times 10^{-4} \in [a, b]$ which corresponds to a challenging scenario for adaptive selection. In fact, the mean reward can only measure the expected quality of an operator when executed locally and independently by each individual worker, whereas the maximum reward measures the expected quality of the next solution that would be obtained more globally by the cooperative master-worker system in one round. In this sense, an accurate *distributed* selection strategy has to acquire information about the quality of an operator when executed cooperatively by all the entities of the system, and not

only on the quality of one operator taken independently of the distributed and cooperative environment where it can be executed. Hence, the maximum reward aggregation has to be preferred when the goal of the adaptive master-worker algorithm is to increase as much as possible the fitness value in each round of computation which is typically the case of a $(1+\lambda)$-EA. More generally, it should be possible to extend this kind of results for others adaptive M/W algorithms which are less explorative, i.e., the global reward should then take explicitly into account an additional diversity measure.

4.3 Analysis of the Heterogeneity Scenarios

The impact of selecting and assigning workers different operators can also be studied as a function of the relative variance of the portfolio operators. In the following, we only consider the maximum reward strategy since it was proved to perform better. For the sake of analysis, let us consider the fitness cloud benchmark where operator 1 follows $\mathcal{N}(-10^{-4}, 10^{-4})$, and operator 2 follows $\mathcal{N}(-10^{-3}, \sigma^2)$. By varying σ, we compute the maximum global reward, *i.e.* the expected improvement of one round of the M/W algorithm when using $n = 256$ workers, in a heterogeneous setting that would split the workers into those that execute operator 1 and those that execute operator 2. By varying the proportion of heterogeneous workers we are then able to compute the optimal number n_1 of workers which should executes operator 1 (the $n - n_1$ executing operator 2) as a function of operator 2 variance σ^2. More precisely, for each value $n_1 \in [0, n]$, the average of the maximum reward on 1000 independent rounds is computed, and the value n_1 with the highest maximum reward is selected and reported in Fig. 4. Clearly, for a wide range of σ values, the optimal value n_1 is either 256, or $n_1 = 0$ for large variance. This indicates that a homogenous setting (with only operator 1 or 2) is optimal except for a small range of variance (between 3.4×10^{-4} and 4.4×10^{-4}). Moreover when an heterogenous strategy is optimal, the gain of maximum reward with an homogeneous strategy is very small (cf. Fig. 4 right). Given these observations we can know understand better the relative performance observed for the different strategies in Fig. 1 for which $\sigma = 5 \times 10^{-4}$.

For the baseline random strategy, the heterogeneous setting is clearly better; since because of the elitism of a $(1 + \lambda)$-EA, it is better to select the wrong operator for half of the workers than one over two rounds. Notice that the baseline heterogeneous random strategy is never better than any others adaptive strategies when using the maximum reward. The homogeneous version of the ϵ-greedy strategy based on the maximum reward significantly outperforms the heterogenous version according to the Mann-Whitney test at level 5%. In contrast, the heterogenous AP outperforms the homogeneous one. Nevertheless, the best strategy is UCB which is homogeneous. According to the exploration power of the strategy, the heterogeneity could help to select the relevant operator; but, when the selection strategy is able to detect the best operator, and when the relative expected gain in fitness improvement is small, a homogeneous setting is to be preferred.

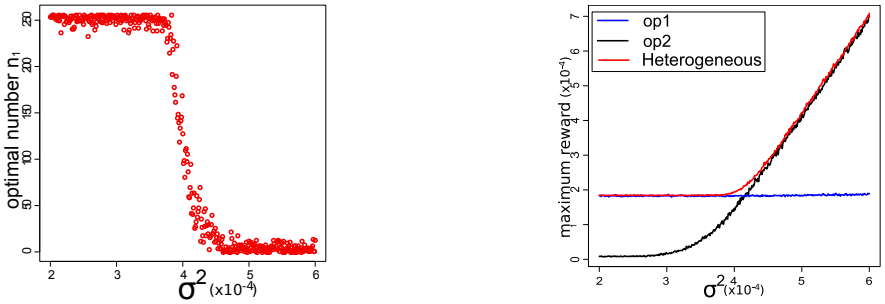

Fig. 4. Optimal number of workers n_1 with operator 1 which maximizes the maximum reward value for an heterogeneous strategy as a function of variance parameter σ^2. The operator 1 and 2 follow respectively the normal law $\mathcal{N}(-10^{-4}, 10^{-4})$, and $\mathcal{N}(-10^{-3}, \sigma^2)$ for $n = 256$ workers. Left: optimal number n_1. Right: Maximum reward values for homogeneous strategies with operator 1 and operator 2, and for the optimal heterogeneous strategy.

5 Conclusions

We conducted an in-depth analysis of the design components of a synchronous M/W adaptive algorithm selection framework. Our main findings can be summarized as follows. The reward associated to each algorithm, which gives the feedback measure for the adaptive selection method, must take into account the performance of the global system, and not only the local performance of each worker. Except when all algorithms have very close performance, an optimal set of algorithms is homogeneous. However, with respect to a particular adaptive strategy, a heterogeneous set could be helpful to continuously enhance its corresponding exploration level. At last, adaptive algorithm selection strategies can be highly effective when their design components in a master-worker architecture are well tuned.

Besides, this first work shall allow us to extend our results for expensive real-world problems, where the evaluation of the fitness function is typically based on computing intensive simulations, e.g., [1]. Another interesting question is the design of reward functions for the asynchronous M/W communication mode. Since a global snapshot of the distributed system is difficult to acquire by the master in such a setting, the reward function is expected to be critically important depending on the different communication to computation trade-offs faced by the master. It is our hope that the new insights provided by our fundamental analysis in the synchronous setting will help addressing such challenging issues.

References

1. R. Armas, H. Aguirre, S. Zapotecas-Martínez, and K. Tanaka. Traffic signal optimization: Minimizing travel time and fuel consumption. In *EA 2015*, pages 29–43, 2016.
2. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

3. P. Baudiš and P. Pošík. Online black-box algorithm portfolios for continuous optimization. In *PPSN XIII*, pages 40–49. Springer, 2014.
4. C. Candan, A. Goëffon, F. Lardeux, and F. Saubion. Non stationary operator selection with island models. In *GECCO*, pages 1509–1516, 2013.
5. L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *GECCO*, page 913. ACM Press, 2008.
6. D. Dasgupta and Z. Michalewicz. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.
7. B. Derbel and S. Verel. DAMS: distributed adaptive metaheuristic selection. In *GECCO*, pages 1955–1962. ACM Press, 2011.
8. M. Dubreuil, C. Gagne, and M. Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE T. on Systems, Man, and Cybernetics: Part B*, 36:229–235, 2006.
9. A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, 2007.
10. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *LION'09*, volume 5851, pages 176–190. Springer, 2009.
11. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *AMAI*, 60:25–64, 2010.
12. M. García-Valdez, L. Trujillo, J. J. Merelo-Guérvos, and F. Fernández-de Vega. Randomized parameter settings for heterogeneous workers in a pool-based evolutionary algorithm. In *PPSN XIII*, pages 702–710. Springer, 2014.
13. A. Goëffon, F. Lardeux, and F. Saubion. Simulating non-stationary operators in search algorithms. *Appl. Soft Comput.*, 38:257–268, 2016.
14. J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128, 1986.
15. C. Jankee, S. Verel, B. Derbel, and C. Fonlupt. Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies. In *EA 2015*, pages 83–96, 2015.
16. C. Jankee, S. Verel, B. Derbel, and C. Fonlupt. A fitness cloud model for adaptive metaheuristic selection methods. In *PPSN 2016*, pages 80–90. Springer, 2016.
17. L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, pages 48–60, 2012-10-30.
18. J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *CEC 2009*, pages 365–372. IEEE, 2009.
19. M. Muniglia, J.-M. Do, L. P. Jean-Charles, H. Grard, S. Verel, and S. David. A Multi-Physics PWR Model for the Load Following. In *ICAPP*, Apr. 2016.
20. D. Sudholt. Parallel evolutionary algorithms. In *Springer Handbook of Computational Intelligence*, pages 929–959, Berlin, Heidelberg, 2015. Springer.
21. R. Tanabe and A. Fukunaga. Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In *CEC*, pages 1263–1270, 2013.
22. D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *GECCO'05*, pages 1539–1546, 2005.
23. S. Wessing, G. Rudolph, and D. A. Menges. Comparing asynchronous and synchronous parallelization of the sms-emoa. In *PPSN XIV*, pages 558–567, 2016.
24. M. Yagoubi and M. Schoenauer. Asynchronous master/slave moeas and heterogeneous evaluation costs. In *GECCO*, pages 1007–1014, 2012.