Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies

Christopher Jankee, Sébastien Verel, Bilel Derbel, Cyril Fonlupt

Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies

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, CRIStAL – CNRS – INRIA Lille

Abstract. In Distributed Adaptive Metaheuristics Selection (DAMS) methods, each computation node can select, at run-time during the optimization process, one metaheuristic to be executed from a portfolio of available metaheuristics. Within the DAMS framework, we investigate different metaheuristic selection strategies which enable to choose locally at each time step a metaheuristic to execute. We conduct a throughout experimental analysis in order to better understand the accuracy and the behavior of the proposed strategies, as well as their relative performance. In particular, we analyze the impact of sharing metaheuristic performance information between compute nodes and the relative effect on each of the considered distributed selection strategies depending on communication topology. Our experimental analysis is performed on the simple one Max problem, for which the best metaheuristics that should be executed at run-time are known, as well as on the more sophisticated NK-landscapes for which non-linearity can be tuned.

1 Introduction

1.1 Motives

A challenging question accruing in practice when solving an optimization problem using evolution algorithms or metaheuristics is the choice of the relevant algorithm, or at least the choice of the parameters of a given algorithm. This choice should typically be guided by the specific features of the tackled problem, even if in a black-box context, those features could be hard to extract.

In this context, a technique for *algorithm selection* consists in selecting the 'best' algorithm to solve a given problem. The original framework of algorithm selection has been proposed by Rice [12]: First some problem features are extracted. According to those features, one algorithm is selected from a set of available algorithms. Then the performance of the selected algorithm is measured on the problem. With the increasing number of available algorithms, and the number of components that can take part in good algorithms, this framework has become more and more popular. Instead of developing a new optimization algorithm, the "design" of relevant algorithm turns out to the identification of the most suitable one or the most suitable components (See [10] for a recent review on algorithm selection).

Similarly, the performance of metaheuristics heavily depends on the correct choice of their parameters. Indeed, algorithm selection is related to parameter setting, in the sense that parameters setting can be associated to a specific algorithm, and *vice-versa*. Eiben *et al.* [4] propose to classify parameter setting methods into two classes. In off-line tuning methods, an algorithm is selected before applying it effectively. Some tuning methods use performance prediction methods based on problem features such as in SATzilla [17], and some others are based on searching in the set of possible algorithm or configurations such as in racing technics [11]. In on-line control methods, the algorithm is selected during the optimization process. At each round, an algorithm is selected from a portfolio of algorithms according to the performance observed in previous rounds. On-line algorithm selection can be modeled as a (dynamic) multi-armed bandit problem: each arm is an optimization algorithm, the reward reflects the quality of solutions produced by the algorithm, and the objective is to select the arms during the optimization process in order to maximize the quality of the final solution. In this context, the so-called Adaptive Operator Selection methods aims at selecting sequentially an operator at each time step. To cite a few, Thierens [14] uses probability matching and adaptive pursuit technics to perform the selection, and Fialho *et al.* [6] propose different selection strategies based on the Upper Confidence Bounds strategy with dynamics restart techniques. For continuous optimization, on-line portfolio techniques have also been recently investigated in [1] using specific reward functions specific to the continuous case.

In this paper, we extend the so-called Distributed Adaptive Metaheuristic Selection (DAMS) framework [3] by investigating on-line portfolio methods in a distributed environment. The DAMS framework is basically motivated by the increasing number of parallel computing facilities (multi-cores, clusters, etc) and the compute power that can offer when tackling hard optimization problems. DAMS is also tightly related to Evolutionary Algorithms (EAs) based on the Island model [15]. parallelize EAs. In such a model, the population is divided into several subpopulations. Each compute node (an Island) applies an EA on those subpopulations, and the subpopulations can interact within a migration phase where solutions can be exchanged. In the context of on-line portfolio methods, we are interested in a *heterogeneous* island model where each island applies its own and possibly different EA. More precisely, the DAMS framework focuses on setting up adaptive strategies to select at each round a relevant EA which is applied to the local sub-population in order to maximize the performance of the whole distributed system. The goal of this paper is to integrate new distributed adaptive strategies and to study their impact within the DAMS framework. In the rest of this paper, we first review some works related to DAMS. Then, we propose a classification of distributed selection strategies into independent and collective ones according to the information exchange. An experimental analysis is then provided and the impact of the considered strategies is reported.

1.2 Related work

Two classes of parameters can be controlled in an island model: the parameters related to the migration policy, and the parameters that define the algorithm at each node.

Control of the migration policy: Candan *et al.* [2] propose to control the migration policy on-line in an heterogeneous island model where each island can apply its own EA. A parameter p_{ij} is used to define the migration rate between islands i and j. According to the island performance in producing promising solutions, the rates are updated using a reinforcement learning principle. Ferdandez *et al.* [5] propose a control method of the EA migration policy when the population is 2d-spatially structured following a 2d-grid. The migration, and thus the EA matting, is controlled by moving the solutions on the grid either randomly, or towards a cell surrounded by similar solutions.

Control of the EA parameters: Instead of using the same parameters setting in every island, in a heterogeneous island model, each island applies its own algorithm. In order to demonstrate the usefulness of such heterogeneous model, Tanabe *et al.* [8, 13] show that a collection of random parameters provides better performance than a uniform static setting. The study focuses on continuous optimization and differential evolution algorithms, and also on two classes of combinatorial problems (QAP, TSP) using a simple genetic algorithm. Following similar ideas, Garcia-Valdez *et al.* [7] showed that for distributed pool-based EA which is another model of heterogeneous islands, a random set of parameters used by a simple GA on the P-Peaks problems outperforms a static setting.

However, in a heterogeneous island model, random parameter setting is not the only possibility. In fact, each EA associated to each island can be controlled during the optimization process according to state of the search in past iterations. For instance, Tongchim *et al.* [16] proposed to select the parameters (cross-over and mutation) of a simple EA adaptively. Two set of parameters are compared on the same compute node, and the best setting with the best solution is sent to other islands. The authors showed that this kind of on-line mechanism improves over static or random settings.

The DAMS framework [3] proposes to locally select at each round and for each node a metaheuristic from a portfolio of metaheuristics in order to maximize the performance of the whole distributed system. For each compute node using a selection strategy, a metaheuristic is selected not only according to the previous performance of the node, but also according to the performance observed and communicated by neighboring nodes. In their paper [3], Derbel *et al.* propose a simple but yet effective strategy called Select-Best-and-Mutate. In this paper, we propose to analyze other alternative selection strategies taking inspiration from existing multi-arm bandit strategies, but in a distributed (island) model.

2 Adaptive selection strategies for DAMS

We first recall the DAMS framework and the original Select-Best-and-Mutate selection strategy. Alternative independent and collective selection strategies based on classical multi-arms bandit strategies are then proposed.

2.1 DAMS and Select-and-Best-Mutate strategy

The Distributed Adaptive Metaheuristic Selection (DAMS) framework has been introduced in [3]. Algo. 1 gives the original algorithm using a generic metaheuristic selection strategy. DAMS is a heterogeneous island-like model algorithm. In each compute node, a metaheuristic from a portfolio is applied on the local subpopulation, and the metaheuristic could be different from one node to another. The authors distinguish three basic levels that can be controlled during one round of a DAMS algorithm: the distributed, the metaheuristic selection and the atomic levels. At the distributed level, information between neighboring nodes are shared, migration of solutions is achieved, and the reward of the metaheuristic that has been executed on the node is communicated to neighbors, and *vice-versa*. At the metaheuristics selection level, one metaheuristic is selected from the portfolio according to previously collected rewards. At the last level of the algorithm, called 'the atomic low level' in the original paper, the selected metaheuristic is applied and the corresponding reward is computed.

The authors also proposed the so-called Select-Best-and-Mutate (SBM) to be used at the selection level. SBM strategy is simply based on a metaheuristic mutation rate p_{mut}. With probability $1 - p_{mut}$, SBM selects the metaheuristic having the best reward in the last round among all neighbors (including the current node), and with rate p_{mut}, SBM selects one random metaheuristic from the portfolio $\mathcal{M}$ different from the best one. In others words, SBM has an intensification component that selects the best rewarded metaheuristic at the previous round from the neighboring metaheuristics, and a diversification component that allows to explore new randomly selected metaheuristic. This strategy is related to the well-known ϵ-greedy strategy in multi-armed bandit problem, which selects the arm with the highest estimated expectation with rate $1 - \epsilon$, and uniformly random arm with rate ϵ. In SBM, the reward of metaheuristic is the maximum reward observed in the last round in the node and the neighboring nodes. There is no long-term memory mechanism which computes an estimated average reward from the previous rounds, and the maximum reward is estimated using the neighboring nodes.

2.2 Independent *vs.* collective selection strategies

Similar to the distributed multi-arm bandit problem, in the distributed adaptive portfolio methods, the collaboration of the k compute nodes can contribute to improve the estimation of the quality of metaheuristics, but with an additional communication cost due to information sharing between nodes. Hence, a distributed metaheuristic selection strategy has to take care of this classical

Algorithm 1: DAMS algorithm for each computation node

Inputs: A portfolio of metaheuristics $\mathcal{M}$
$r \leftarrow$ Init_Reward()
$M \leftarrow$ Init_Meta($\mathcal{M}$)
$P \leftarrow$ Init_Pop()
repeat
  /* Distributed Level:
     migration and information sharing */
  Send Msg(r, M, P) to each neighbor
  $\mathcal{P} \leftarrow \{\}$; $\mathcal{S} \leftarrow \{\}$
  for *each neighbor* w **do**
    Receive Msg(r', M', P') from w
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{P'\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(r', M')\}$
  $P \leftarrow$ Update_Population($P, \mathcal{P}$)
  /* Metaheuristic Selection Strategy Level */
  $M \leftarrow$ Select_Meta($\mathcal{M}, (r, M), \mathcal{S}$)
  /* Atomic Low Level:
     apply metaheuristic and compute reward */
  $P_{new} \leftarrow$ Apply(M, P)
  $r \leftarrow$ Reward(P, P_{new})
  $P \leftarrow P_{new}$
until *Stopping condition is satisfied*;

trade-offs in distributed systems. Moreover, multi-arm bandit strategies are often a combination of two parts, one exploitation part which promotes the best estimated arm, and one exploration part which looks at new random arms. The exploration part is particularly important when facing a non-stationary problem. The strategy should be able to explore arms for which the reward could have changed. Therefore, when several computation nodes collaborate to improve the metaheuristic quality estimation, the exploitation part could be reinforced too much forcing the strategy to converge too quickly in a non-stationary scenario.

We distinguish two extreme types of selection strategies according to the information sharing between nodes. In *independent* selection strategies, the metaheuristic selection depends solely on the reward information produced locally by the node. In *collective* selection strategies, the selection takes into account the reward information communicated by the neighboring nodes. For example, the SBM strategy is a collective strategy, and a baseline strategy which selects a metaheuristic uniformly at random is an independent strategy.

2.3 Independent selection strategies

First, we can derive a simple independent selection strategy from the original SBM strategy. In fact, instead of selecting the best rewarded metaheuristic from neighboring nodes, we can select the best rewarded metaheuristic in the last W rounds and executed locally by a node – no reward information from neighbors is

used. Accordingly, the original collective SBM strategy will be denoted as SBMc, and the newly designed independent SBM by SBMi. Notice that SBMi comes with two parameters, the original mutation rate p_{mut}, and the windows size W.

The so-called Adaptive Pursuit (AP) belongs to the class of probability matching algorithms. AP is a classical adaptive selection strategy used in optimization [14], and can be used as an independent selection strategy. In adaptive pursuit algorithm, a metaheuristic i is applied at time step t in proportion to a probability $p_{i,t}$, and those probabilities are updated according to the rewards of metaheuristics. This technique is then divided into three parts: the update of the reward estimation $\hat{q}_{i,t}$ of the metaheuristics, the update of the probabilities $p_{i,t}$, and the selection of the metaheuristic. Eq. 1 defines the update of estimated reward of the metaheuristic i. Variable $r_{i,t}$ is the reward at round t of the metaheuristic i, and parameter $\alpha \in (0, 1]$ is the adaptation rate.

$$\hat{q}_{i,t+1} = \hat{q}_{i,t} + \alpha \ . \ (r_{i,t} - \hat{q}_{i,t}) \tag{1}$$

The update of the probabilities $p_{i,t}$ is given by Eq. 2 where i_t^* denotes the metaheuristic with the best $\hat{q}_{i,t}$:

$$p_{i,t+1} = \begin{cases} p_{i,t} + \beta \ . \ (p_{max} - p_{i,t}), \text{ if } i = i_t^* \\ p_{i,t} + \beta \ . \ (p_{min} - p_{i,t}), \text{ otherwise.} \end{cases} \tag{2}$$

For the best estimated metaheuristic, the probability converges to p_{max} with the learning rate β, for the other metaheuristics, the probability converges to p_{min}. At round t, the AP selects the metaheuristic at random in proportion of probability $p_{i,t}$. This independent strategy is denoted by APi.

Several Upper Confidence Bound (UCB) algorithms are used in the context of adaptive metaheuristic selection (see [6] for a review). Let $n_{i,t}$ denotes the number of times the i^{th} metaheuristic is applied up to round t, and let $\hat{q}_{i,t}$ denotes the average empirical reward of metaheuristic i. At each round t, UCB selects the metaheuristic that maximizes the following quantity:

$$\hat{q}_{i,t} + C \ . \ \sqrt{\frac{2 \log(\sum_j n_{j,t})}{n_{i,t}}}$$

Parameters C enable to control the exploitation / exploration trade-off. This independent selection strategy is denoted by UCBi.

The UCB strategy is an optimal strategy for stationary problems with independent arms which is actually not the case metaheuristics control. The average empirical reward could be far from the current new reward. To overcome this drawback, the average empirical reward can be computed over a slicing windows by considering the last W rounds. This variant is denoted by UCB-Wi.

Finally, a dynamic version of UCB is introduced in [6] and uses the Page-Hinkley test to detect whether the empirical rewards collected for the best metaheuristic have changed significantly. For more details, the reader is referred to page 6 in [6]. This selection strategy will be denoted by UCBP-PHi, and it requires two parameters: a restart threshold γ and a robustness threshold δ.

2.4 Collective selection strategies

Each of the above-mentioned independent selection strategies can be used to define a collective selection strategy that takes into account the reward information exchanged with neighboring nodes. In collective SBM which is the original one, the best rewarded metaheuristic is selected from the set of neighboring nodes. In collective AP, the rewards of all neighbors are iteratively used to update the estimation of reward $\hat{q}_i$. Notice that the order of the update could have an impact on the estimation. So, at the initialization phase, a pre-established order between neighboring nodes is randomly chosen. Then, after the updates of reward $\hat{q}_i$, the probability p_i is updated once for all neighbors. In the collective versions of UCB strategies, the empirical average $\hat{r}_i$ is also updated using the rewards of neighboring nodes. The numbers of times $n_{i,t}$ that each metaheuristic is applied is also update according to the information given by each nodes. Notice that in that case, the order of the update does not matter. The selected metaheuristic is the metaheuristic selected after taking into account all neighbors information. Those collective strategies versions are denoted respectively SBMc, APc, UCBc, UCB-Wc, UCB-PHc.

3 Experimental Analysis

3.1 Experimental Setup

Following previous works [16, 3, 6, 2] on adaptive portfolio selection, we also use the well known one-Max problem, which counts the number of 1 in a bit string. In a similar scenario, we use a portfolio of four $(1 + \lambda)$-ES: from one parent solution, the algorithm produces λ solutions according to a stochastic operator and selects the best one for the next iteration. Four operators are used: three operators respectively flip exactly 1, 3 and 5 bits, and the last one uniformly flips each bit with rate $1/N$ where N is the bit strings size set to $N = 1000$.

We use an elitism migration mechanism. Each node (island) sends their current solution to their neighboring nodes. Then, each node receives all solutions from the neighboring nodes. The best solution from the set containing the received solutions and the current solution of the node replace the current solution of the each node. The DAMS algorithm stops when the global maximum is found by one node of the distributed system, when the number of rounds exceeds $T_{limit} = 5.10^4$. 200 runs are computed for each possible strategy and topology. The performance of algorithms is measured either with the number of rounds to reach the global maximum, either using the expected running time (ERT). ERT is expected running time to reach a level fitness of the algorithm with simulated restart. It is equal to $E_s[T] + (1 - \hat{p}_s)/\hat{p}_s.T_{limit}$ where $\hat{p}_s$ the estimated success rate, and $E_s[T]$ is the average number of rounds when the fitness level is reached.

We study four topologies of network: the complete topology where each node is connected to all others nodes, a random topology where there is an edge between two nodes with probability $p = 0.1$, the grid topology which is a two-dimensional regular square grid where each node is connected to the four nearest

neighbors, and the circle topology where the nodes are connected to two others nodes to form a circle. The size of the networks is $n \in \{4, 16, 32, 64\}$. In order to have the same number of fitness evaluations in one round whatever the network size n, the λ parameter is set to $64/n$.

A couple of parameters are used in the different selection strategies. For the SBM strategies, the value of metaheuristic mutation rates are $p_{mut} \in \{0.001, 0.002, 0.01, 0.1\}$. The window size of the SBMi is set to 5. For AP, the extreme values are set to $p_{min} = 0.1$ and $p_{max} = 1$. The adaptive and the learning rates are $\alpha \in \{0.1, 0.25, 0.5, 0.75, 1\}$ and $\beta \in \{0.1, 0.25, 0.5, 0.75, 1\}$. For all the UCB strategies, the parameter C values are $\{0.1, 0.5, 5, 25, 100\}$. For the variant UCB-W, the set of windows sizes is $\{10, 100, 1000\}$. Following [6], the parameters of Page-Hinkley test are to $\delta = 0.15$, the restart thresholds γ are from $\{0.5, 0.75, 1, 2, 5, 10\}$. Moreover, 2 baseline strategies are used: the random one (rnd.) select at random at each round a metaheuristic, and the constant one (cst.) always select the same metaheuristic which is randomly chosen at the beginning.

3.2 Computational Results

One-Max Overall Performance. From a purely distributed perspective, the first interesting measure is the number of rounds it takes for an algorithm to find the global maximum. The number of rounds provides an idea about the degree of parallelism in an ideal scenario where the communication cost is assumed to be negligible compared to the cost of function evaluation. The relative performance of the different strategies is summarized in Table 1. The best performing parameters are set for each strategies. Several observations can be extracted from Table 1. First, the performance of the different strategies are consistent with the considered configurations in the sense that they can overall be ranked similarly independently of the topology type or graph size. More importantly, we remark that the impact of exchanging rewards information between node has a strong impact on performance. Interestingly, this impact is positive in the case of SBM and AP, whereas it is not when considering UCB. In fact, SBMc appears to overall outperform all the other strategies and APc appears to performing best when both considering the circle, grid and random topologies with large number of nodes. In contrast, the performance of the four implemented versions of UCB is deteriorating systematically as the information from neighbors is incorporated. We attribute this to the fact that this information is actually pushing the UCB strategy to diversify more the search as soon as some operators (even with a good rewards) has been used by other neighboring nodes. UCB is less effective than random selection. The C-value which tunes the exploration-exploitation tradeoff has no impact on this result. Indeed, we have performed an extended sensitive analysis of parameter C (not presented here to save space) which does not changed this result. This also suggests that the UCB strategy has to be completely rethought in order to infer accurate exploration-exploitation tradeoff in the dynamic distributed setting. Notice however, that independent UCB-HPi is still able to provide very competitive results compared to SBMc and APc.

Table 1. For each topology and graph size, number of selection strategies which statistically outperforms (according to the Wilcoxon test at confidence level $p = 0.05$) a given strategy method for the one-Max problem with $N = 1000$. The 0 value is the best one: no other strategy significantly outperforms the considered one.

Topo.	Size	cst.	rand.	SBMi	SBMc	APi	APc	UCB					
								UCBi	UCBc	HPi	HPc	Wi	Wc
circle	4	8	4	1	0	7	7	10	11	2	3	3	3
circle	16	4	6	3	0	4	0	10	11	1	6	6	6
circle	32	4	6	3	1	4	0	10	11	2	6	6	9
circle	64	4	6	3	2	4	0	10	11	1	6	6	9
grid	4	8	4	1	0	4	9	10	11	2	4	3	3
grid	16	4	5	2	0	4	0	10	11	1	4	6	4
grid	32	4	5	3	1	4	0	10	11	1	4	4	6
grid	64	4	6	3	1	4	0	10	11	1	6	6	9
rnd.	4	7	3	0	0	5	7	10	11	0	3	3	3
rnd.	16	4	4	1	0	4	3	10	11	1	4	4	5
rnd.	32	4	4	3	1	4	0	10	11	2	4	4	9
rnd.	64	4	4	3	1	4	0	10	11	1	4	4	9
compl.	4	7	3	1	0	7	7	10	10	2	3	3	3
compl.	16	6	3	1	0	5	6	11	10	1	4	3	9
compl.	32	3	3	2	0	3	8	11	10	1	3	3	9
compl.	64	3	3	2	0	3	3	11	10	1	3	7	9
Average		4.875	4.312	2	0.4375	4.375	3.125	10.187	10.75	1.25	4.187	4.437	6.562

Sensitivity to parameters. In the previous discussion, we were focused on the overall behavior of the different strategies for a fixed parameter setup. In fact, one may wonder what is the impact is of the parameters used for every strategy. This is illustrated in Fig. 1 where we give representative examples on the sensitivity of SBM, AP and UCB-HP to different parameter settings both in the case of an independent and a collective strategy. We can appreciate that SBM is rather stable under different configurations although for the collective variant, the impact of the mutation rate is slightly more pronounced (a small values is advised). The same thing holds for the AP strategy where the algorithm is robust to a wide range of values of α and β, with he exception of the adaption rate $\alpha = 1$ which is to be avoided since it promotes strong convergence in the reward estimation. For the UCB-HP strategy, the value of C, which appears in the confidence bound, plays an important role but only in the independent strategy. For the collective strategy, where the information from neighbors is actually deteriorating performance, the C-value does not seem to have any impact and cannot help obtaining improved results.

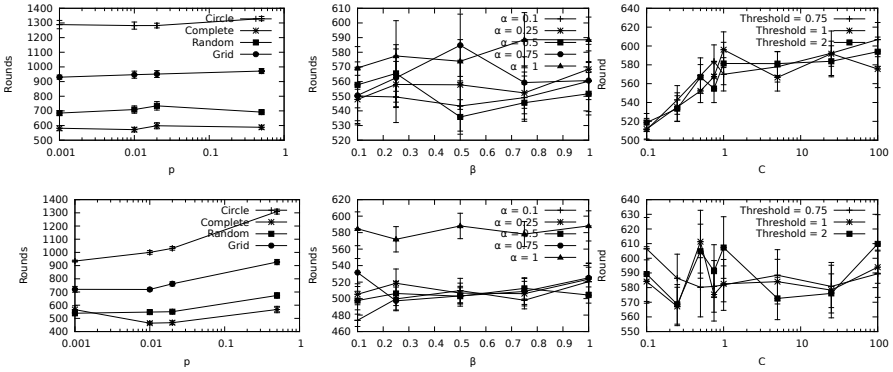

Fig. 1. Average number of rounds to find the maximum of the one-Max problem as function of the parameter values of different selection strategies. From left to right: SBM, AP, UCP-HP strategies ; top: independent selection, bottom collective selection.

Parallelism. In the previous discussions, we were only interested in analyzing the relative behavior of the strategies for a fixed topology. In particular, the results of Table 1 do not allow us to appreciate the relative impact of different topologies on the performance of each strategy. For this purpose, we show in Fig. 2 the relative performance of SBM and AP in different configurations. It is important to recall that the number of function evaluations at every single round and for all the considered configurations is the same which means that the number of function evaluations needed overall in any of the considered configuration is by the *same* multiplicative factor similar to the number of rounds depicted in Fig. 2. This observation has an important impact, since then, we are able to obtain different trade-offs when considering the number of exchanged messages as an important indicator of *parallel speed-ups* that one could obtain when effectively deploying our strategies in a real distributed setting. In fact, the number of messages needed to exchange information is exactly the number of rounds times the number of edges used in the considered topology. In the case of the complete (resp, circle, grid, random) topology, the number of edges is $n(n-1)/2$ (resp. $n-1$, $O(n)$, $O(p.n^2)$) where n is the number of nodes. From Fig. 2, we can notice that the number of rounds stays stable for the complete and random topology (except for 4 nodes) with the complete topology being slightly better. However the number of rounds increases sharply for the circle and the grid which we attribute to the increase of the topology diameter. Roughly speaking, although the increase in the number of rounds for the circle and the grid is at most by a factor of 2, the number of needed messages stays linear in the number of nodes. This is to contrast with the complete topology where the increase in the number of messages is polynomial. Hence, in a practical setting where the cost of message-passing is non-negligible, we claim that the best choice would

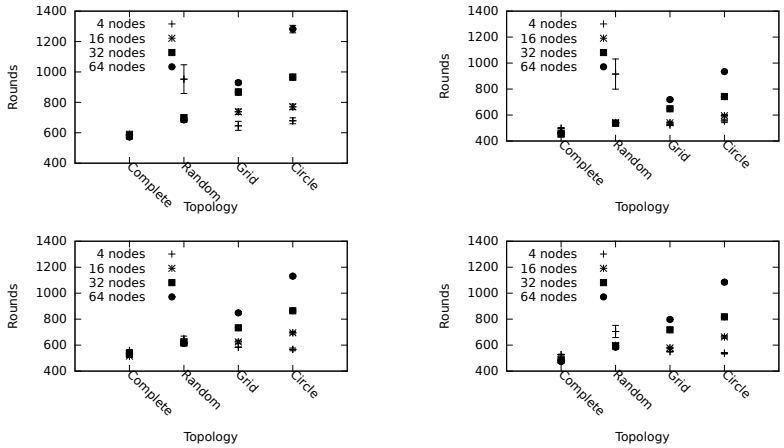

Fig. 2. Average number of rounds to find the maximum (one-Max problem) according to the topology and the number of nodes. From left to right and top to bottom: SBMi, SBMc, APi, and APc strategies.

be the random topology which exhibits the most appealing tradeoffs in terms of the number of rounds *v.s.* the number of messages exchanged overall.

NK-landscapes In this paper, we also consider a more sophisticated class of problems captured by the so-called NK-landscapes. The family of NK-landscapes constitutes a model of multimodal problems [9]. The search space is binary strings of size N: $\{0,1\}^N$. N refers to the problem size, and K to the number of bits that influence a particular position from the bit-string, *i.e.* the epistatic interactions. The objective function $f : \{0,1\}^N \rightarrow [0,1)$ to be maximized is defined as follows.

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i, x_{i_1}, \ldots, x_{i_K})$$

where $f_i : \{0,1\}^{K+1} \rightarrow [0,1)$ defines the component function associated with each bit x_i. By increasing the number of epistatic interactions K from 0 to $(N-1)$, NK-landscapes can be gradually tuned from smooth to rugged. In this work, we set the position of these interactions at random. Component values are uniformly distributed in the range $[0,1)$.

Our interest in the NK-landscapes stems from the fact that usually different bit-flip mutation rates are believed to provide different performances. To illustrate this claim, we show in Fig. 3, the empirical probability that a solution with the fitness given by the x-axis is be improved if a uniform bit-flip operator with rate c/N is applied, where c varies in the range $\{1,2,4,8,16\}$. We can clearly see that the operator which is likely to provide an improvement depends strongly of the attained fitness level. Hence, this kind of landscapes appears to be particularly interested to be studied within the DAMS framework. Accordingly, we

Table 2. Rank of the different strategies according to the topology and the number of computation nodes for NK-landscapes with $N = 1000$ and $K = 1, 4, 8$.

Topo.	Size	unif.	cst.	rand.	SBMi	SBMc	APi	APc	UCB					
									UCBi	UCBc	HPi	HPc	Wi	Wc
		$K = 1$												
compl.	16	0	9	6	3	7	12	2	1	10	4	5	8	11
compl.	64	0	10	7	6	1	4	9	12	3	2	5	11	8
circle	16	0	2	11	3	5	1	10	8	7	6	4	12	9
circle	64	0	7	8	2	1	6	4	12	9	3	10	11	5
average		0	7	8	3.5	3.5	5.75	6.25	8.25	7.25	3.75	6	10.5	8.5
		$K = 4$												
compl.	16	0	6	12	9	1	11	3	2	4	8	10	5	7
compl.	64	0	6	3	8	5	11	12	1	4	10	2	7	9
circle	16	1	11	12	8	4	5	6	3	7	0	2	10	9
circle	64	0	10	9	11	6	7	12	5	4	3	2	1	8
average		0.25	8.25	9	9	4	8.5	8.25	2.75	4.75	5.25	4	5.75	8.25
		$K = 8$												
compl.	16	1	3	9	0	11	7	6	2	10	4	8	5	12
compl.	64	0	12	4	10	3	6	9	11	2	5	8	1	7
circle	16	7	0	4	5	6	12	3	9	10	2	1	8	11
circle	64	0	2	12	3	11	8	9	5	10	7	1	4	6
average		2	4.25	7.25	4.5	7.75	8.25	6.75	6.75	8	4.5	4.5	4.5	9

perform the same experiments while considering different NK-landscapes with $N = 1000$ and $K \in \{1, 4, 8\}$. The portfolio of metaheuristics is composed by five $(1 + \lambda)$-ES based on the uniform bit-flip rate c/N with rates $c = 1, 2, 4, 8$, and 16. We tune the parameters according to the results the one-Max problem: $p_{nut} = 0.01$ for SMB, $\alpha = 0.5$ and $\beta = 0.5$ for AP, and $C = 25$ for UCB strategies. Interestingly, we find that no significant differences can be reported between any of the considered selection strategies when looking at the final fitness value (this is nor reported due to space limitations). However, we are able to report different behavior when examining the empirical expected running time (ERT) to attain the median fitness value (computed over all configurations).

The ERT results are summarized in Table 2. In addition to adaptive selection strategy, we also tested a uniform and static strategy, denoted unif in the table, where every nodes share the same metaheuristic all along the execution. In the table, we choose to present the performance of the best uniform-static strategy which is not the same according to the topology and the number of nodes. Perhaps, the most interesting observation is than the uniform-static strategy is the best performing and none of the considered DAMS variants is able to outperform it. This might be surprising at first sight, but not if we account for the time required to learn the best metaheuristic to apply. In fact, when examining carefully Fig. 3 in light of the information given by the empirical improvement probability, we can see that the fitness level is increasing very abruptly for NK-

landscapes in the early stages of the search. Hence, the different fitness windows where one has to choose the best operator are very tight which is to contrast with the time it may need for a strategy to detect which operator is actually the best to apply. As a consequence, even though the fixed operator used by a uniform static strategy is not optimal in all the stages of the execution, it still does not loose time in learning by testing less efficient operators. It worth-noticing that the previous experiments raise the question of whether we really need to adapt the search heuristics at runtime and does it really serve in practice? We argue that the answer to this question is definitively yes. In fact, the general lessons that we can learn from our experiments with the NK-landscapes can be formulated as following. First, in a black-box scenario, the time during which a metaheuristic is the best one depends strongly on the landscape. Hence, learning this landscape at runtime is for sure a plausible alternative. Second, we need to study more carefully the cost of the learning stage of selection strategy in function of the considered landscape, and to design novel alternative adaptive strategy that would be able to minimize the learning cost at the aim of improving efficiency.

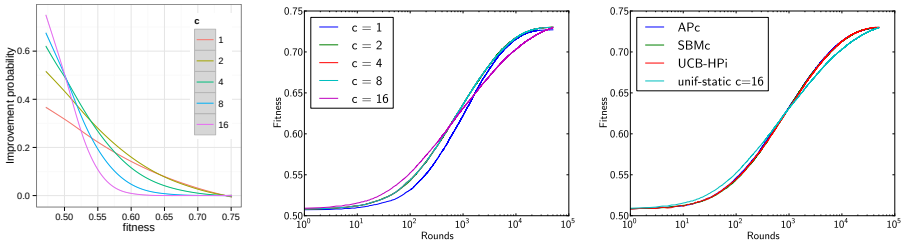

Fig. 3. Empirical improvement probabilities *vs.* fitness level (left). Fitness *vs.* rounds in log-scale. Center: uniform-static, right: different strategies. NK-landscapes with $K = 4$.

4 Conclusion

In this paper, we investigate new adaptive strategies for distributed metaheuristic selection. Accordingly, we explored the applicability of adaptive pursuit and upper bound confidence based algorithms in the distributed setting where several heterogeneous islands have to cooperate in order to select the most accurate metaheuristic dynamically at runtime. In particular, we consider the possibility of incorporating the distributed information coming from the neighboring islands and study its impact on the search behavior by considering independent and collective schemes. We conduct a throughout experimental study in order to better understand the major ingredients toward making such schemes successful. We find that special care must be taken when attempting to use the rewards observed distributively at different islands in order to obtain accurate

exploration-exploitation trade-offs. Besides, our study keeps open several questions that deserve further investigation in the future. For instance, we could analyze the selection strategies on others benchmarks such as knapsack or graph coloring problems. It would also be interesting to study the gain one can achieve by the proposed strategies when effectively deployed in a real distributed testbed. In such a setting, the communication cost is very likely to introduce new challenges; but the increasing power offered by modern computation systems is worth to be investigated in order to derive highly efficient adaptive strategies.

References

1. P. Baudiš and P. Pošík. Online black-box algorithm portfolios for continuous optimization. In *PPSN XIII*, pages 40–49. Springer, 2014.
2. C. Candan, A. Goeffon, F. Lardeux, and F. Saubion. A dynamic island model for adaptive operator selection. In *GECCO '12*, pages 1253–1260, 2012.
3. B. Derbel and S. Verel. DAMS: distributed adaptive metaheuristic selection. In *GECCO '11*, pages 1955–1962, 2011.
4. 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.
5. C. M. Fernandes, J. L. Laredo, J. J. Merelo, C. Cotta, R. Nogueras, and A. C. Rosa. Shuffle and mate: A dynamic model for spatially structured evolutionary algorithms. In *PPSN XIII*, pages 50–59. Spriger, 2014.
6. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *AMAI*, 60:25–64, 2010.
7. 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.
8. Y. Gong and A. Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *CEC 2011*, pages 820–827, 2011.
9. S. A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.
10. L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, pages 48–60, 2012.
11. M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The R package irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, 2011.
12. J. R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.
13. R. Tanabe and A. Fukunaga. Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In *CEC'13*, pages 1263–1270, 2013.
14. D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *GECCO'05*, pages 1539–1546, 2005.
15. M. Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag, 2005.
16. S. Tongchim and P. Chongstitvatana. Parallel genetic algorithm with parameter adaptation. *Information Processing Letters*, 82(1):47 – 54, 2002.
17. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.