A Fitness Cloud Model for Adaptive Metaheuristic Selection Methods

Christopher Jankee, Sébastien Verel, Bilel Derbel, Cyril Fonlupt

A Fitness Cloud Model for Adaptive Metaheuristic Selection Methods

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

[1] Univ. Littoral Côte d'Opale, EA 4491 - LISIC, France,
[2] Université Lille, CRIStAL – UMR 9189 – INRIA Lille, France

Abstract. Designing portfolio adaptive selection strategies is a promising approach to gain in generality when tackling a given optimization problem. However, we still lack much understanding of what makes a strategy effective, even if different benchmarks have been already designed for these issues. In this paper, we propose a new model based on fitness cloud allowing us to provide theoretical and empirical insights on when an on-line adaptive strategy can be beneficial to the search. In particular, we investigate the relative performance and behavior of two representative and commonly used selection strategies with respect to static (off-line) and purely random approaches, in a simple, yet sound realistic, setting of the proposed model.

1 Introduction

Context and motivation. In the last decades, the optimization community has gained much expertise in the design of general purpose randomized heuristics to tackle hard optimization problems. Nonetheless, there cannot exist a universal solving method; which partially explains the plethora of available algorithms. We argue that the automatic choice of an effective algorithm, the smart combination of low level components and the proper tuning of their underlying parameters is one of the most challenging questions that the optimization community has to face in the next coming years. This issue is of interest both for its practical importance and also for the new research opportunities it opens for the design of novel high level techniques.

Two main approaches can be reported [3]: (i) off-line tuning (static choice of parameters before optimization) and (ii) on-line tuning (dynamic tuning of parameters). It is still an open issue to understand what makes these approaches act differently both at the practical level, and also at a more fundamental level with respect to the performance of solvers as a function of problem features. Generally speaking, this research aims at enhancing our understanding of such an issue by abstracting from a specific problem and instead proposing a high level model allowing us to provide both theoretical and empirical evidence on the expected behavior of algorithm configuration methods.

Background on models for adaptive selection strategies. In this article, we focus on on-line adaptive algorithm selection. From a portfolio of algorithms at each iteration of the search, a *selection strategy* aims at choosing the hopefully "best" algorithm to execute on the current set of solutions according to the

previously observed performance of available algorithms in the portfolio. In [1, 2], the authors use some specific benchmarks to study novel selection strategies and improving the underlying reward metrics. For instance, in [1], continuous benchmarks are experimented using a portfolio of variants of the well-established differential evolution operator. Alternatively, other works considered to directly define the rewards associated with the algorithms using particular stochastic distributions [11, 4]. The purpose is to be able to study some specific properties of a given adaptive selection strategy such as its ability to detect and to learn the best algorithm from the portfolio. In [11], the set of possible rewards is defined by different uniform random distributions that are reassigned randomly to the portfolio at different time intervals. For instance, in [4], the so-called "Two-Values benchmarks" is used where two possible reward values and a probability of wining the highest is defined depending on pre-computed time intervals. Recently, a benchmark was proposed in [6] where the rewards depend on the number of times that an operator is applied during a time window in order to study a scenario where a number of operators providing different exploration/exploitation tradeoffs are available. Several properties should be fulfilled by a relevant benchmark depending on the target issue to be studied. First, one has to take into account the stochasticity of most heuristic algorithms. Hence, the reward of each algorithm in the portfolio should typically be defined by choosing a relevant probability distribution. In order to appreciate the relative quality of the target selection strategies, the so-called "oracle", that is the optimal selection strategy, should be known. At last, since the performance of an algorithm in the portfolio could evolve during the optimization process, the reward distribution has to be tightly coupled with the state of the search. This aims at increasing generality and abstracting away specific algorithmic design issues. For instance, in [11] and related benchmarks, the reward depends on time, and not directly on the state of search; in [6] and related benchmarks, the state of the search is defined by the number of times an operator is used independently of the quality of current solutions. We argue that despite their skillful design, the existing benchmarks are not sufficient by their own to allow for a global fundamental understanding of the design of adaptive methods and the setting of relevant theory for them.

Contribution. In this work, we propose a new model called Fitness Cloud (FC) model inspired by *fitness cloud* [12]. The proposed model is to be viewed in a complementary manner to existing benchmarks. In the FC model, the state of the search is naturally defined by the fitness of the current solution, and the performance of a given metaheuristic is function of the current fitness value. The reward distribution is hence *not* controlled explicitly; but instead, kept as an implicit feature implied by the considered adaptive mechanisms or approaches to be designed independently and studied subsequently.

As a preliminary step we consider in this work a simple usage of the FC model with a portfolio composed by two metaheuristics having fixed performance qualities across two configurable fitness ranges. This setting allows however to study two main issues. First, it allows us to provide theoretical evidence on when a static (off-line) selection strategy is more beneficial upon an adaptive (on-line)

Algorithm 1 A single-solution single-operator basic metaheuristic.

```
1: x_0 ← initialization()
2: repeat
3:    for i = 1 ... λ_t do
4:       y_i ← operator(x_t)
5:    end for
6:    x_{t+1} ← selection(x_t, y_1, ..., y_{λ_t})
7: until stopping criterion is true
```

strategy. Second, through an empirical analysis, and by considering two widely used on-line adaptive strategies based on multi-armed bandits, we gain a more deep understanding on when and why such approaches could be effective with respect to baseline static or purely random strategies [9, 5].

The rest of the paper is organized as follows. In Sec. 2, the fitness cloud model is defined with a simple theoretical analysis. In Sec. 3, different instantiations of the proposed scenario are considered and the relative performance and behavior of different selection strategies are elicited by a throughout empirical study. In Sec. 4, we conclude the paper and discuss future research directions.

2 Fitness cloud model and theoretical analysis

Before going into more details, and although the proposed model is independent of a particular metaheuristic, let us consider for the sake of clarity the template of Algo. 1 rendering the design of a basic single-solution single-operator metaheuristic. The considered iterative algorithm has two parts. First, a stochastic local operator is applied to the current solution x_t to produce a set of λ_t candidate solutions y_i. Such an operator could be the random bit-flip mutation when the search space is the set of binary strings. Second, a new current solution x_{t+1} is selected. This is typically performed according to the fitness values, given by the fitness function f, of the newly generated solutions y_i, and the current solution x_t. A classical example of selection is the $(1 + \lambda)$-EA which selects the best solutions so far. Notice that despite its simplicity such a template encompasses a wide range of algorithms.

2.1 Model Definition

The Fitness Cloud (FC) model informs about the fitness value of solutions after one iteration according to the fitness of the current solution. To make it simple, the FC model supposes that the state of the search is only given by the fitness $f_t = f(x_t)$ of the current solution x_t (see Algo. 1). Assuming that the selection rule only takes into account the fitness values (which is a common practice for a wide range of metaheuristics), no particular model is required for the selection step. But a specific model is needed to capture the stochastic behavior of most evolutionary operators when generating new candidate solutions. The basic idea behind the FC model is to assume that the fitness after applying a stochastic operator is given by a conditional probability distribution:

$$\Pr(f(y) = z' \mid f_t = z) \tag{1}$$

Being said, different choices of this probability distribution can be made such as discrete distributions (binomial, Poisson, etc.), or continuous distributions (normal, Weibull, etc.). Given its properties of convergence, we choose the use a normal distribution in this paper:

$$\Pr(f(y) = z' \mid f_t = z) \sim \mathcal{N}(\mu(z), \sigma^2(z)) \tag{2}$$

where $\mu(z)$ and $\sigma^2(z)$ are respectively the mean and the variance of the normal distribution which can depend on the fitness z of the solution and which are to be set to map a target setting. As a consequence, the evolution of the fitness during one iteration follows a conditional probability distribution which embeds the previous distribution. One important feature of the probability distribution is the *expected improvement* of one metaheuristic iteration, denoted by $E^+(z)$, which is the expected progress of the fitness given the current fitness value is z:

$$E^+(z) = \int_z^\infty \Pr(f_{t+1} = z' \mid f_t = z)\ z'\ dz' \tag{3}$$

2.2 Definition of a simple scenario with two fitness ranges

The previous considerations are broad enough to allow us to define a more concrete and relevant simple scenario, where we are given a portfolio of two elitist metaheuristics (see Fig. 1). More precisely, we first assume that the possible fitness values are normalized in the range $[0, 1]$. The search is assumed to start with fitness value 0 and stops when the fitness value 1 is reached. The whole range $[0, 1]$ is then divided into two fitness ranges: the first one from fitness 0 to $r \leqslant 1$, and the second range from r to 1. We then consider a portfolio of two heuristic algorithms having different relative performance in these two ranges. For this purpose, the relative behavior of each algorithm in the portfolio is modeled accordingly in each fitness range using the fitness cloud model. More precisely, at each fitness range, we shall fix the mean and variance of the conditional normal distribution in Eq. 2 as follows: $\mu_i(z) = z + K_{\mu_i}$ and $\sigma_i^2(z) = K_{\sigma_i}$ where for each metaheuristic M_i, $i \in \{1, 2\}$, parameters K_{μ_i} and K_{σ_i} are different constant numbers at each fitness range. Therefore, we end up with 9 parameters to be fixed in this scenario: r, and the 8 parameters for the normal distributions for each metaheuristic and at each fitness range. However, as it will be shown in section 2.3, the expected running time to reach the optimal value 1 depends on the expected improvement of each metaheuristic. Hence, only 5 parameters are free as illustrated by Fig. 1; where $E^+_{i,j}$ denotes the expected fitness improvement of metaheuristic M_i, $i \in \{1, 2\}$, for the fitness range $j \in \{1, 2\}$. Additionally, we assume that the best metaheuristic for the first fitness range is M_1, whereas it switches to M_2 in the second fitness range, i.e., $E^+_{2,1} < E^+_{1,1}$, and $E^+_{1,2} < E^+_{2,2}$. Finally, like in many optimization problems, we assume that the expected improvement decreases when the fitness value increases: $E^+_{1,2} < E^+_{1,1}$, and $E^+_{2,2} < E^+_{2,1}$. It is important to notice that the relative performance of algorithms in the portfolio does not depend explicitly neither on time (number

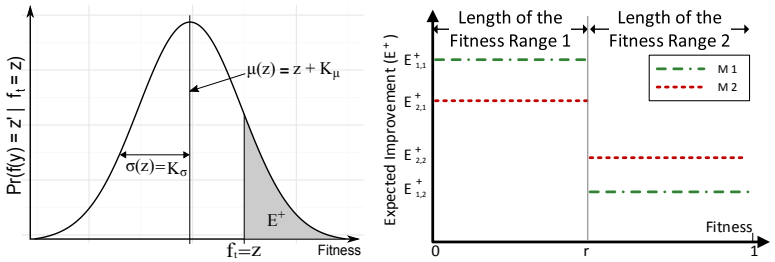

Fig. 1. Fitness cloud model: scenario with two metaheuristics and two fitness ranges.

of iterations), nor on the number of times a metaheuristic is applied; but solely on the state of the search which is assumed to be implied by the current fitness value.

2.3 Theoretical insights

In this section, we assume an off-line static strategy that selects arbitrary one metaheuristic, denoted M, in the portfolio and executes it on the previously described scenario until reaching the target fitness value 1. We shall assume that the considered metaheuristic follows the template of Algo. 1 initialized with a solution having fitness value 0 and implementing an elitist selection when deciding on the next solution, *i.e.* the best solution is retained for the next iteration. Notice that the expected improvement of the considered metaheuristic M at each iteration is by definition constant within each of the two fitness ranges defined in the considered scenario. We shall denote by E_1^+ (resp. E_2^+), the expected improvement within the first (resp. second) fitness range. Let us consider the running time of metaheuristic M, that is, the first hitting time (number of iterations) to reach the final fitness value: $T = \min\{t \mid F_t \geqslant 1\}$ where F_t is the random variable which gives the fitness value of the best solution found at iteration t. Notice that the total number of evaluations depends on λ_t and the number of evaluation in each operators. We then can prove the following:

Theorem 1. *The expected running time verifies:* $T_{up} - \frac{\delta}{E_2^+} \leqslant E[T] \leqslant T_{up}$ *with*

$$T_{up} = \frac{r}{E_1^+} + \frac{1-r}{E_2^+} \qquad \text{and} \qquad \delta = \begin{cases} 1-r \text{ if } 1-r \leqslant E_1^+, \\ E_1^+ \quad \text{if } E_1^+ < 1-r \end{cases}$$

Proof. Let $T_1 = \min\{t \mid F_t \geqslant r\}$ and $T_2 = \min\{t - T_1 \mid F_t \geqslant 1\}$. By definition and by the linearity of expectation, we have that $E[T] = E[T_1] + E[T_2]$. Now we prove the following lemma:

Lemma 1. *Let* $(k, \ell) \in [0,1]^2$ *such that either* $r \leqslant k \leqslant \ell$ *or* $k \leqslant \ell \leqslant r$. *Let* $T' = \min\{t \mid F_t \geqslant \ell\}$ *and* $F_0 = k$. *Let* E^+ *is the expected improvement in the fitness range* $[k, \ell]$. *Then,* $E[T'] = (\ell - k)/E^+$.

The proof of the Lemma is an application of theorem 1 in [8] which can be stated in short by: if $E[X_t - X_{t+1}|X_t] = \delta$, then $E[T_0|X_0] = X_0/\delta$. In fact, by considering the random variable $X_t = \ell - F_t$, we have by definition of the fitness cloud model $E[X_t - X_{t+1} \mid X_t] = E^+$ which gives the necessary additive drift condition and the proof of the lemma follows immediately.

Since the expected improvement in the first fitness range is E_1^+, applying the previous lemma with $F_0 = 0$ provides: $E[T_1] = r/E_1^+$. Similarly, let $k \geqslant r$ the fitness of the (best) current solution just after a solution x_t with fitness greater than r is found for the first time. Since the expected improvement in the second fitness range is E_2^+, applying the previous lemma with $F_0 = k \geqslant r$ provides: $E[T_2] = (1-k)/E_2^+ \leqslant (1-r)/E_2^+$. The stated upper bound is hence proved. Let's now define $Y_{t'} = 1 - F_{t'}$ where $t^{'} = t - T_1$. Hence, $E[T_2|Y_0] = Y_0/E_2^+$. By applying the law of total expectation, we get $E[T_2] = E[Y_0]/E_2^+$. Let $F_{T_1} = F_{T_1-1} + \Delta_{T_1-1}$ where Δ_{T_1-1} is the random variable of the fitness difference between the iterations $T_1 - 1$ and T_1. By definition of T_1, $F_{T_1-1} < r$, and from the fitness cloud model we have that $E[\Delta_{t'-1}] = E_1^+ > 0$. It follows that: $E[Y_0] \geqslant 1 - r - E_1^+$. When $1 - r - E_1^+ \leqslant 0$, the algorithm is able to reach final fitness without any iteration in the second fitness range. Otherwise, for $1 - r - E_1^+ > 0$, the algorithm spends at least $(1 - r - E_1^+)/E_2^+$ iterations in the second fitness range 2. □

For example, with the $(1+1)$-EA which generates a single candidate solution and keeps it if it is better than the current one, the expected improvement $E^+(z)$ at fitness value z is given by: $E^+(z) = \int_z^\infty \frac{t}{\sigma\sqrt{2\pi}} \exp(\frac{-(t-\mu)^2}{2\sigma^2})dt = (\mu - z).(1 - \Phi(\frac{-(\mu-z)}{\sigma})) + \frac{\sigma}{\sqrt{2\pi}} \exp(\frac{-(\mu-z)^2}{2\sigma^2})$ where Φ is the cumulative distribution function of the standard normal distribution. Accordingly, the evolution of the expected running time for the two possible metaheuristics M_1 and M_2 as a function of the length r of the first fitness range, is illustrated in the right side of Fig. 2.

3 Experimental analysis of adaptive selection strategies

3.1 Experimental design

From the scenario defined previously (see Fig. 1) where a portfolio of two metaheuristics are given; with the metaheuristic M_1 (resp. M_2) being better on the fitness range 1 (resp. 2), we were able to experiment several possible settings of the underlying FC model. Overall, and considering that M_1 and M_2 are implemented as an elitist $(1+1)$-EA, we only retain 3 cases corresponding to typical different instantiations that were found to be the most representative of the different challenges that this scenario allows to consider. In Fig. 2, we summarize these 3 experimental cases while providing the parameters used in the FC model. Actually, as shown in the theoretical analysis, the expected improvement (EI for short) is crucially important in each fitness range. Hence, the mean and the standard deviation of the normal distributions are chosen to obtain the desired EI. By choosing K_μ negative, we emulate the behavior of a typical stochastic operator that decreases on average the fitness of current solution as it is the case very often in practice. In all the 3 cases, the EIs of M_1 and M_2 are the same in the first fitness range, while being different by a factor of 2. This actually does not penalize much metaheuristics when moving from one range to the other, which makes Case 2 a reference case with respect to the other cases. In fact, in Case 1, the EI values are much more closer in the second range, *i.e.* an oracle would

Values are given with a factor of 10^{-3}.

Cases	Meta.	Fitness range 1			Fitness range 2		
		$E^+_{i,1}$	K_{μ_i}	K_{σ_i}	$E^+_{i,2}$	K_{μ_i}	K_{σ_i}
Case 1	M_1	6	-1	16.27	1.8	-2	6.72
	M_2	3	-1	8.72	2	-2	7.24
Case 2	M_1	6	-1	16.27	1	-2	4.59
	M_2	3	-1	8.72	2	-2	7.25
Case 3	M_1	6	-1	16.27	0.2	-2	2.14
	M_2	3	-1	8.72	0.4	-2	2.84

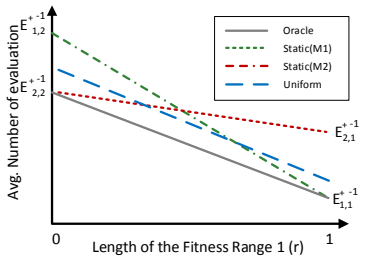

Fig. 2. Parameters values of the 3 cases with its corresponding sketch (upper bounds of expected running time): exp. impr. ($E^+_{i,j}$), mean difference (K_{μ_i}), and std. dev. (K_{σ_i}).

act as in Case 2, but the performance of a static selection strategy that would always choose M_1 becomes much closer to the oracle than in Case 2. As for Case 3, the same factor of 2 is kept between the EIs in the second range; but the EIs has been reduced by a huge factor of 15, hence making the progress in the second range relatively much more difficult than in the first range compared to Case 2.

For our experimental investigations, we consider two selection strategies used in multi-armed bandits framework. Due to the lack of space, we only detail the experimented parameters without going into a technical discussion; the reader is referred to [6] for a review and a detailed description of the following adaptive selection strategies, namely Upper Confidence Bound (UCB) and Adaptive Pursuit (AP). The UCB strategy [4] estimates the upper confidence bound of the expected reward of each arm and selects the one with the higher bound. A parameter C tunes the exploitation/exploration trade-off. The AP selection strategy [10] uses exponential recency weighted average to estimate the average, tuned by an adaption rate α, and selects a metaheuristic according to probabilities updated by a learning rate β. For UCB, the set of studied parameters C is $\{0.0008, 0.01, 0.1, 0.75, 2, 4, 10, 20, 25, 50\}$, and for AP, both adaptation and learning rate parameters are in the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Additionally, we include three other strategies in our analysis. The *oracle strategy* selects the best metaheuristic in each fitness range, *i.e.* M_1 in the first range and M_2 in the second range. The *uniform strategy* selects at each iteration one metaheuristic uniformly at random among M_1 and M_2. Notice that in this case the expected improvement is the mean of expected improvements of M_1 and M_2. The *static strategy* selects always the same metaheuristic, that is either M_1 or M_2 before the execution is started. All results are averaged over 100 independent runs.

3.2 Empirical analysis

In this section, we analyze the relative performance and the behavior of the considered strategies. The performance measure is the average number of evaluations to reach the target fitness value 1. For fairness, we consider the best parameter setting for each strategy. For each case, we compute the average rank of a setting over all values of r, and the best ranked setting is selected. The performance comparison is based on the Mann-Whitney test with a confidence

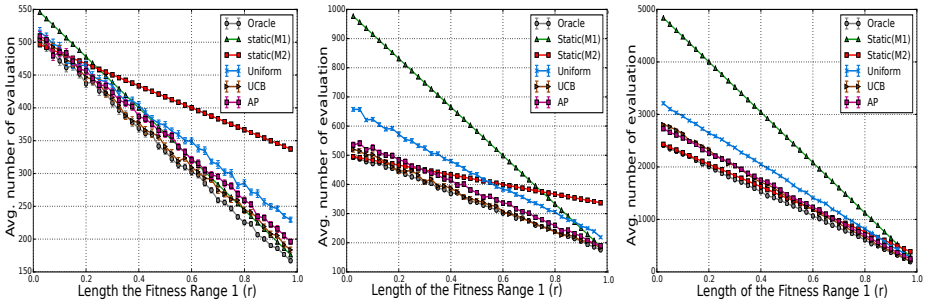

Fig. 3. Comparison of selection methods with the best parameters settings for UCB $C = 4$, and for AP $\alpha = 0.1$, $\beta = 0.1$. From left to right: test cases 1, 2, and 3.

level of 0.05. In Fig. 3, we show the performance obtained in the three test cases as a function of the length r of the first fitness range. Notice that for the considered Cases 1, 2, and 3, the average difference over the r-values of the performance between the oracle and the best static strategy (either with M_1 or M_2) is respectively 20, 57, and 100 evaluations. This is the maximum performance gap between an optimal adaptive method and an optimal off-line static strategy tuned for each value of r.

Adaptive strategies *vs.* uniform. As suggested by Th. 1, the expected performance of static, oracle, and uniform selection strategies decreases linearly with the length of the first fitness range. The performance of UCB and AP strategies are also linear from our empirical data (Pearson correlation coefficients are very close to -1). For all test cases, and for any length r of the first range, UCB and AP strategies perform significantly better than the uniform random selection strategy (except for few values of r where no significant difference with AP is found). Interestingly, the average gap between UCB and uniform strategy in test cases 1, 2, and 3 is respectively around 26, 87, and 263 evaluations, which is much higher than the difference between the oracle and the best static strategy.

Adaptive strategies *vs.* static. The performance of adaptive strategies can be worse than a static strategy. For example, in the test case 2, UCB is better than any static strategy for $r \in [0.11, 0.99]$. Otherwise, when the length of the fitness range where the expected improvement of M_1 is the best, is short ($r < 0.11$), the static strategy choosing M_2 is better than UCB. At the opposite, for $r > 0.99$, the static strategy choosing M_1 is better than UCB. The performance of the adaptive selection strategies also depends on the expected improvements in the second range. Respectively for the cases 1 and 3, the r-values intervals where UCB strategy outperforms static strategies are $[0.09, 0.81]$ and $[0.61, 1.00]$. On average over the r-values, in test case 1 and 2, UCB outperforms the static strategy by 10 and 49 evaluations respectively. However, in the test case 3, when the expected improvements of algorithms in the portfolio in the second range are very small compared to the first one, a static strategy is preferred, indeed UCB requires 71 additional evaluations on average than static.

UCB *vs.* AP strategies. Overall, the AP strategy never outperforms UCB strategy significantly except in 3 minor exceptions for the lowest values of r

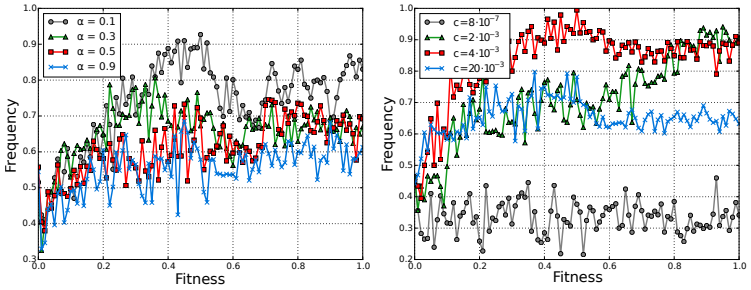

Fig. 4. Frequency of the best metaheuristic selection according to the fitness value for different parameter settings. Case 2 with $r = 0.5$ AP (left) with $\beta = 0.1$ UCB (right).

in the case 3. In the case 1, UCB is better than AP for $r > 0.35$, and the difference between UCB and AP is 10 evaluations in average, which is half of the difference between best static strategy and oracle. In test case 2, UCB strategy is better than AP except for the 3 largest values of r, while being very close to the oracle, *i.e.* the average difference for UCB is only 8 evaluations compared to the 32 evaluations for AP, and 57 evaluations for a static strategy. In case 3, the performance difference between of UCB and AP is much closer (only 15 evaluations on average). Although UCB outperforms AP for r-values larger than 0.37, both strategies are on average worst than the optimal static strategy by a factor of 1.8. The Fig. 4 shows the selection frequency of the best metaheuristic according to the current fitness value for UCB and AP in Case 2 and when the expected improvement of metaheuristics is changed at fitness value $r = 0.5$. The UCB strategy converges at fitness 0.3 for $C = 0.004$, and AP around fitness 0.5 for $\alpha = 0.1$. In addition, when the best metaheuristic changes at fitness value 0.5, the UCB strategy recovers more quickly the best metaheuristic than AP.

Uniform *vs.* static. It is shown several times that random selection of parameters could outperformed a tuning method with static value of parameters [9, 5]. The model with two fitness ranges scenario helps us to understand why and when random selection can be advantageous. The uniform strategy is better than the best static strategy when the length r belongs to the intervals $[0.14, 0.4]$, $[0.56, 0.88]$, and $[0.6, 1]$ respectively for cases 1, 2, and 3. Roughly speaking, a random uniform selection, and moreover an adaptive strategy, is more efficient when the performances of each metaheuristic in the portfolio are close.

Discussion. This two-fitness-range scenario allows us to fine-tune the performance of each metaheuristic at the two stages of the search. The comparison of case 2 and 3 shows that when the average expected improvements at the second stage is much lower than in the first stage, an adaptive method becomes less efficient except when the length of the first stage is very large. Indeed, the time that can be gained in the first stage becomes negligible and the main difficulty then turns out to be the final convergence to the optimum value. When the scale of the average expected improvements between the two stages is moderate like in test case 2, an adaptive method like UCB strategy is very effective. However, when the performance difference between metaheuristics at one stage of the

search becomes small, like in test case 1, the problem is equally difficult for all metaheuristics in the portfolio, and the adaptive selection becomes rather useless besides the fact that it becomes more difficult to detect the best performing metaheuristic at a given iteration.

4 Conclusion

It is our hope that the fitness cloud model opens new research paths allowing to understand, to test and to design new adaptive portfolio methods in different settings. In this work, using a simple scenario, we provide properties of when and why an (on-line) adaptive selection strategy, or random selection could outperform an (off-line) static strategy. Indeed, the fitness cloud model goes beyond the intuition and allows to give a formal framework in order to analyze selection strategies in portfolio and to understand their behavior in different settings.

Following the natural question of Baudiš *et al.* in his conclusion [1] on *"the influence of portfolio size and composition on performance of various strategies"*, it would be possible to design relevant scenarios to deeply study those questions. It would also be possible to conduct a fine grained analysis of other selection strategies in a sequential as well as in parallel context [7]. It will also be interesting to extend the fitness cloud model to multi-objective optimization where the design of adaptive portfolio method is relatively in its infancy beginning.

References

1. P. Baudiš and P. Pošík. Online black-box algorithm portfolios for continuous optimization. In *PPSN XIII*, pages 40–49. Springer, 2014.
2. L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *GECCO'08*, page 913, 2008.
3. A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Param. Setting in EA*, pages 19–46. Springer, 2007.
4. A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *AMAI*, 60:25–64, 2010.
5. 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, 2014.
6. A. Goëffon, F. Lardeux, and F. Saubion. Simulating non stationary operators in search algorithms. *Applied Soft Computing*, 38:257 – 268, 2016.
7. C. Jankee, S. Verel, B. Derbel, and C. Fonlupt. Distributed Adaptive Metaheuristic Selection: Comparisons of Selection Strategies. In *EA 2015*, pages 83–96, 2015.
8. P. K. Lehre and C. Witt. General drift analysis with tail bounds. Technical Report 1307.2559, arXiv, 2013.
9. R. Tanabe and A. Fukunaga. Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In *CEC'13*, pages 1263–1270, 2013.
10. D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *GECCO'05*, pages 1539–1546, 2005.
11. D. Thierens. Adaptive strategies for operator allocation. In *Param. Setting in EA*, volume 54, pages 77–90. Springer, 2007.
12. S. Verel, P. Collard, and M. Clergue. Where are Bottlenecks in NK Fitness Landscapes? In *CEC'03*, pages 273–280, 2003.