Adaptive Operator Selection at the Hyper-level

Eduardo Krempser[1], Álvaro Fialho[2], and Helio J.C. Barbosa[1]

[1] Laboratório Nacional de Computação Científica, Petrópolis, Brazil
{krempser,hcbm}@lncc.br
[2] Nokia Institute of Technology (INdT), Manaus, Brazil
alvaro.fialho@indt.org.br

Abstract. Whenever a new problem needs to be tackled, one needs to decide which of the many existing metaheuristics would be the most adequate one; but it is very difficult to know their performance a priori. And then, when a metaheuristic is chosen, there are still its parameters that need to be set by the user. This parameter setting is usually very problem-dependent, significantly affecting their performance. In this work we propose the use of an Adaptive Operator Selection (AOS) mechanism to automatically control, while solving the problem, (i) which metaheuristic to use for the generation of a new solution, (exemplified here by a Genetic Algorithm (GA) and a Differential Evolution (DE) scheme); and (ii) which corresponding operator should be used, (selecting among five operators available for the GA and four operators for DE). Two AOS schemes are considered: the Adaptive Pursuit and the Fitness Area Under Curve Multi-Armed Bandit. The resulting algorithm, named as Adaptive Hyper-Heuristic (HH), is evaluated on the BBOB noiseless testbed, showing superior performance when compared to (a) the same HH without adaptation, and also (b) the adaptive DE and GA.

Keywords: Hyper-heuristics, adaptive operator selection, parameter control, multi-armed bandits, area under the curve.

1 Introduction

Metaheuristics have been used to solve a wide range of complex optimization problems. Many different algorithmic schemes can be found in the literature, each of them presenting its own specifications, resulting into different behaviors with respect to the exploration of the search space. The resulting characteristics might be more adequate or not to a given problem or class of problems. It is very difficult, however, to know *a priori* which would be the most adequate metaheuristic whenever a new problem needs to be tackled. Additionally, metaheuristics usually have many user-defined parameters that might significantly affect their behavior and performance. In the case of evolutionary algorithms, for example, there is the population size, the selection and replacement mechanisms and their inner parameters, the choice of variation operators and their corresponding application rates, etc. As a result, once a given metaheuristic is chosen, there is still the need of correctly setting its parameters, what can be

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 378–387, 2012.
© Springer-Verlag Berlin Heidelberg 2012

seen as a complex optimization problem *per se*. There are thus two levels of decision: (i) which algorithm should be used, and (ii) which values should be used for setting its parameters. As of today, these decisions are usually done by following the user's intuition, or by using an off-line tuning procedure aimed at identifying the best strategy for the problem at hand. Besides being computationally expensive, off-line tuning however generally delivers sub-optimal performances, as the appropriate strategy depends on the stage of the optimization process: intuitively, exploration-like strategies should be more frequently used in the early stages while priority should be given to exploitation when approaching the optimum. Regarding the first decision, a solution might come from the so-called hyper-heuristics. In [10], a hyper-heuristic is described as a heuristic scheduler that does the scheduling over a set of heuristics in a deterministic or non-deterministic way. A more comprehensive survey can be found in [2]. Concerning the second decision, more specifically in the case of setting the application rates of variation operators, a recent trend is to use methods that control, while solving the problem, which variation operator should be applied according to the recent performance of all available operators. These methods are commonly referred to as Adaptive Operator Selection (AOS) [6].

In fact, the problem of selecting which variation operator to apply can be seen as exactly the same problem of selecting which metaheuristic to use, but at a different abstraction level. Thus, in this paper we propose the use of AOS schemes at the two levels of abstraction (the hyper- and the lower level), in an independent way, while solving the problem.

We empirically analyze the use of two prominent schemes found in the literature, the probability-based Adaptive Pursuit (AP) method [13], and the most recent bandit-based method, referred to as the Fitness-based Area-Under-Curve-Bandit (AUC) [5]. Both are compared with each other at both levels, and also with what would be the choice of a Naive user, namely, the uniform selection of the operators. On the hyper-level, the AOS schemes are expected to autonomously select, while solving the problem, which metaheuristics (in our numerical experiments, Differential Evolution (DE) or Genetic Algorithm (GA)) should be applied. At the lower level, there are five operators for the GA case, and four in the case DE is chosen. DE and GA were chosen because they have been widely used in many fields and their efficiency has already been verified several times. Other heuristics as well as more than two could have been used.

A brief overview of the GA and the DE adopted in this work, as well as an introduction to AOS and to the existing schemes employed, is presented in section 2. In section 3 the proposed schemes are depicted. The computer experiments are presented in section 4, and the paper ends with some conclusions in Section 5.

2 Background

In this work, AOS schemes are used at the hyper-level in order to select between the DE and GA metaheuristics. Both of them will now be briefly described.

2.1 Genetic Algorithms

Here a real-coded GA was applied with a rank-based selection scheme. Moreover, a large number of genetic operators have already been developed and those we adopted are listed below (considering x_i, $i = 1, \ldots, N$ the variables in a chromosome, x_i^L and x_i^U respectively the lower and upper bounds for x_i):

- The one-point (1X) crossover operator which is the analogue of the standard one-point crossover for binary-coded GAs.
- The Uniform crossover (UX) [12], where each gene in the offspring is created by copying the corresponding gene from either parent according to a randomly generated crossover mask.
- The blend crossover operator (BLX-α) [3].
- A simple mutation operator (delta mutation – DM) that increments each variable with a given rate of application according to:

$$x_i = p_i + \delta \Delta_{max}$$

  where p is the parent, x is the offspring, δ is a random number, and Δ_{max} is a fixed quantity, which represents the maximum permissible change in the parent.
- The non-uniform mutation (NUM) operator [9]. When applied to an individual x at generation *gen* and when the total number of generations allowed is *maxgen*, mutates a randomly chosen variable x_i according to

$$x_i \leftarrow \begin{cases} x_i + \Delta(gen, x_i^U - x_i) & \text{if} \quad \tau = 0 \\ x_i - \Delta(gen, x_i - x_i^L) & \text{if} \quad \tau = 1 \end{cases}$$

  where τ is randomly chosen as 0 or 1 and the function $\Delta(gen, y)$ is defined as

$$\Delta(gen, y) = y(1 - \mu^{(1 - \frac{gen}{maxgen})^{\eta}})$$

  with μ randomly chosen in [0, 1] and the parameter η set to 2.

2.2 Differential Evolution

The original proposal of DE by Storm and Price [11] presents a simple and efficient algorithm for global optimization over continuous spaces. The main variants (or strategies) of the DE modify the way the individuals are selected to participate in the mutation, which in the original proposal was done randomly (called DE/rand/1/bin). The Algorithm 1 shows the pseudo-code for this variant.

The additional variants considered here basically change line 11 of Algorithm 1:

- **DE/rand/2/bin:**
  $u_{j,i} = x_{j,r_1} + F(x_{j,r_2} - x_{j,r_3}) + F(x_{j,r_4} - x_{j,r_5})$
- **DE/rand-to-best/2/bin:**
  $u_{j,i} = x_{j,r_1} + F(x_{j,best} - x_{j,r_1}) + F(x_{j,r_2} - x_{j,r_3}) + F(x_{j,r_4} - x_{j,r_5})$
- **DE/current-to-rand/1/bin:**
  $u_{j,i} = x_{j,i} + F(x_{j,r_1} - x_{j,r_i}) + F(x_{j,r_2} - x_{j,r_3})$

where r_1, r_2, r_3, r_4 and r_5 are randomly selected individuals and $x_{j,best}$ is the best individual in the population.

Algorithm 1. Algorithm DE/rand/1/bin.

input : NP (population size), GEN (# of generations), F (mutation scaling), CR (crossover rate)

```
1  G ← 0;
2  CreateRandomInitialPopulation(NP);
3  for i ← 1 to NP do
4      Evaluate f(x_{i,G}) ;   /* x_{i,G} is an individual in the population */
5  for G ← 1 to GEN do
6      for i ← 1 to NP do
7          SelectRandomly(r1, r2, r3) ;                    /* r1 ≠ r2 ≠ r3 ≠ i */
8          jRand ←RandInt(1, N) ;     /* N is the number of variables */
9          for j ← 1 to N do
10             if Rand(0, 1) < CR or j = jRand then
11                 u_{i,j,G+1} = x_{r3,j,G} + F.(x_{r1,j,G} − x_{r2,j,G});
12             else
13                 u_{i,j,G+1} = x_{i,j,G};
14         if f(u_{i,G+1}) ≤ f(x_{i,G}) then
15             x_{i,G+1} = u_{i,G+1};
16         else
17             x_{i,G+1} = x_{i,G};
```

2.3 Adaptive Operator Selection

The Adaptive Operator Selection aims to adjust the application of operators while the search process is performed, according to the operators performance. Thus, we need to define two aspects: how to measure the performance of the operators, usually referred to as Credit Assignment, and how to select among them after these performance measurements are made, simply called here Operator Selection.

More specifically, the Credit Assignment firstly measures the impact caused by the operator application in the optimization process, and then transforms this impact into a meaningful numerical credit that will be used for updating the empirical quality estimates of each operator. The most common impact measure is simply the fitness improvement achieved by the generated offspring w.r.t. its parent(s). Then, the credit assigned to the operator can be: (i) the Instantaneous reward, *i.e.*, received after the last application; (ii) the Average of the rewards received over a few recent applications; (iii) or the Extreme reward recently received by the operator [4]. The number of recent applications considered for the latter two is usually a user-defined parameter, referred to as W (size of the sliding window).

For the Operator Selection, we consider here two existing schemes from the literature, which were chosen for having shown superior performance in recent works. The first one is called Adaptive Pursuit (AP) [13]. It calculates an application probability for each operator, and use a roulette wheel to select the next operator to be applied. A lower bound on the probabilities is employed

to preserve some level of exploration, and a winner-takes-all scheme is used to push forward the current best operator. In this work, the AP Operator Selection scheme is used in combination with the Extreme Credit Assignment.

Although alleviating the user from the need of selecting which operators should be applied to the problem at hand, and doing so in an on-line manner, the most common operator selection schemes, including the AP method, involve some hyper-parameters that need to be tuned as well. The use of credit assignment schemes based on the raw values of the fitness improvements make these hyper-parameters highly problem-dependent.

Motivated by this issue, the Fitness-based Area-Under-Curve - Bandit (AUC), a fully comparison-based adaptive operator selection was recently proposed[5]. Its robustness comes from its Credit Assignment scheme, which is based on the ranks of the fitness improvements, and not on their raw values. Briefly, it works as follows. The latest W rewards achieved by all operators are ranked, and an exponentially decaying factor is applied over the rank values, so that the top ranked rewards have a significant weight, while the very low rewards have a weight close to zero. These decayed rank values are then used to construct a curve analog to the Area Under the ROC Curve, a criterion originally used in Signal Processing and later adopted in Machine Learning to compare binary classifiers. The ROC Curve associated to a given operator s is drawn by scanning the ordered list, starting from the origin: a vertical segment is drawn when the current offspring has been generated by s, a horizontal segment is drawn otherwise, and a diagonal one is drawn in case of ties. The length of each segment is proportional to its decayed rank value. Finally, the credit associated to operator s is the area under this curve. This Credit Assignment scheme is coupled with a bandit-based Operator Selection which deterministically chooses the strategy to be applied based on (a variant of) the Upper Confidence Bound algorithm [1].

3 Adaptive Hyper-Heuristic

The adaptive algorithm proposed here combines the GA and DE techniques by choosing during the evolutionary process which metaheuristic and which operators should be used. The metaheuristics are used in an interleaved way by choosing one of them to generate each new individual in the population, *i.e.*, each new individual is generated following the choice algorithm with its operators and parent selection mechanism. The generated individual is thus compared with the target (current) individual of the population and the fittest one is maintained in the population of the next generation, following the DE replacement mechanism. The algorithms and operators are chosen by AOS methods where the impact measure is defined by the improvement in fitness between the offspring (generated individual) and its parent (target individual for DE and the best parent for GA).

A similar idea was applied to select the operators of each algorithm, *i.e.*, the four variants described in section 2.2 in the DE case, and the five operators in section 2.1 in the GA case are selected according with the response of the AOS method. The pseudo-code of the proposed algorithm is presented in Algorithm 2.

Algorithm 2. HH-AOS

```
input : NP (population size), GEN (# of generations)
1  G ← 0;
2  CreateRandomInitialPopulation(NP);
3  for i ← 1 to NP do
4      Evaluate f(x_{i,G}) ;   /* x_{i,G} is an individual in the population */
5  for G ← 1 to GEN do
6      for i ← 1 to NP do
7          op ← AOS-selectOperator();
8          if op == DE then
9              u = DE-generate-one-individual(NP, x_G);
10         else
11             u = GA-generate-one-individual(NP, x_G, p);
12         if Evaluate f(u) < Evaluate f(x_{i,G}) then
13             x_{i,G+1} = u;
14             if op == DE then
15                 AOS-ApplyReward(x_{i,G} - x_{i,G+1});
16             else
17                 AOS-ApplyReward(p - x_{i,G+1}) ;   /* p is the best parent
                   selected to generate the new individual */
```

4 Comparative Results

In order to evaluate the performance of our proposal, experiments were conducted using the BBOB noiseless testbed [7],which includes 24 single-objective functions from 5 different classes with very different characteristics and levels of complexity. The default guidelines were followed: 15 trials per function [8], with the maximum number of function evaluations being fixed at $10^5 \times d$. The BBOB experimental set-up uses as performance measurement the Expected Running Time (ERT), defined as follows: given a target function value, ERT is the empirical expected number of function evaluations for attaining a fitness value below the target. In other words, it is the ratio of the number of function evaluations for reaching the target value over successful trials, plus the maximum number of evaluations for unsuccessful trials, divided by the number of successful trials. Due to space constraints, the presented results are restricted to dimension $d = 20$, although similar conclusions can be taken for the other considered dimensions.

Our proposal, herein called Adaptive Operator Selection at the Hyper-Heuristic (HH) level, was compared with each algorithm (DE and GA) individually with three different selection techniques: (i) uniform selection (Naive), (ii) the adaptive pursuit selection (AP) and (iii) the fitness area under curve bandit selection (AUC).

The first analysis, depicted in Figs. 1 and 2, presents a comparison among the three operator selection techniques within HH. The uniform choice reached the target value in an least one instance, for the highest level of precision (1e-8), in only 12 of the 24 function, while the HH with any of the two AOS techniques solved 17 functions. Besides, the AOS methods require fewer function evaluations to reach the target value (Fig. 2): for 50% of the cases, HH using an AOS method is at least two times faster than HH with the naive uniform operator selection. Although no significant difference could be found between the AP and the AUC AOS schemes, considering the speed-up ratio presented in the figure 2, and also the analysis described in [6] only the latter will be used in the following. A further

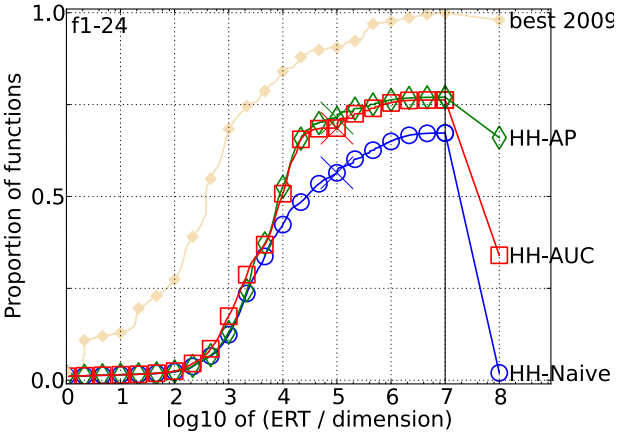
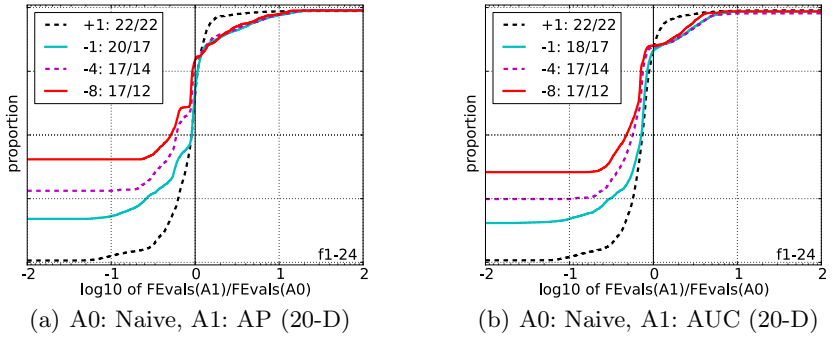


Fig. 1. Empirical cumulative distribution of the bootstrapped distribution of ERT over dimension for 50 targets in $10^{[-8..2]}$ for all functions to HH



(a) A0: Naive, A1: AP (20-D) (b) A0: Naive, A1: AUC (20-D)

Fig. 2. Empirical cumulative distributions (ECDF) speed-up ratios in 20-D to HH. ECDF of FEval ratios of Adaptive Pursuit (AP) and AUC-Bandit (AUC) divided by Naive, all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the number of functions that were solved in at least one trial (AP/AUC first)

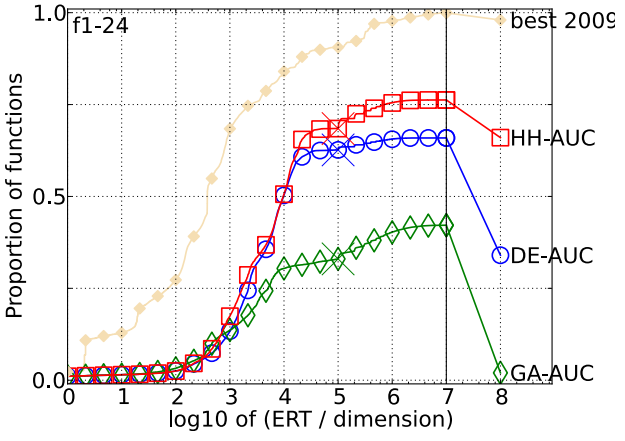
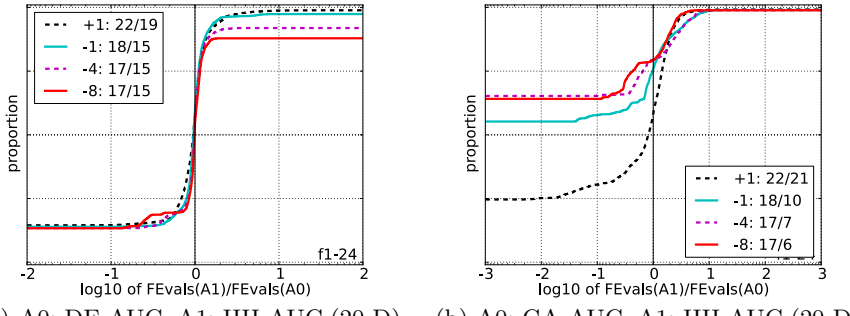


Fig. 3. Empirical cumulative distribution of the bootstrapped distribution of ERT over dimension for 50 targets in $10^{[-8..2]}$ for all functions to DE, GA and HH, all using the AUC adaptive operator selection



(a) A0: DE-AUC, A1: HH-AUC (20-D) (b) A0: GA-AUC, A1: HH-AUC (20-D)

Fig. 4. Empirical cumulative distributions (ECDF) speed-up ratios in 20-D to DE-AUC, GA-AUC and HH-AUC. ECDF of FEval ratios of HH-AUC respectively divided by DE-AUC and GA-AUC, all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the number of functions that were solved in at least one trial (HH-AUC first).

analysis compares the performance of the HH-AUC with both DE and GA also using the AUC AOS mechanism. The difference is that the HH variant uses independent AOS schemes in the two levels of abstraction, while DE and GA have only one AOS instance selecting between their operators in the usual way. The results are presented in Figs. 3 and 4. As it can be seen, the autonomous selection between DE and GA done by the HH algorithm by means of the AOS methods is able to solve more instances than both DE and GA individually. This empirically confirms that the efficient mixture of DE and GA is better than each of the original methods alone.

5 Conclusions

In this paper, we propose the use of existing Adaptive Operator Selection (AOS) schemes at the Hyper-level, in order to automatically select between different metaheuristics for the generation of each new solution. The metaheuristics exemplified here were Differential Evolution (DE) and Genetic Algorithm (GA). Additionally, an independent AOS instance was also employed to automatically select between the corresponding variation operators in the usual way, selecting between four different operators whenever DE was chosen by the Hyper-AOS, and five operators otherwise. The resulting algorithm, that can be seen as an adaptive Hyper-Heuristic (HH), employs thus three independent instances of the recent Fitness Area Under Curve Multi-Armed Bandit AOS algorithm [5]: one instance controlling the choices at the Hyper-level, and the other selecting between the operators for the DE and GA algorithms. For both levels of abstraction, the impact of each AOS decision is computed by means of the fitness improvement achieved when comparing the newly generated offspring with its parent.

The proposed algorithm, tested under the light of the very comprehensive Black Box Optimization Benchmarking (BBOB) noiseless testbed [7], showed superior performance when compared to: (i)the same Hyper-Heuristic without adaptive behavior (uniformly selecting between the metaheuristics and operators), and (ii) the single-heuristic counterparts, *i.e.*, the DE and GA alone, using the same AOS mechanism to select between their corresponding variation operators. These results empirically confirm that the AOS at the Hyper-level is efficient, and that the intelligent switching between different metaheuristics is a path worth to be further investigated.

There are mainly two different paths that might be taken in the follow up of this work. One concerns its extension from the algorithmic point of view, by trying to improve and/or propose new AOS mechanisms for better efficiency at the Hyper-level. The other regards its extension from the application point of view, by analyzing the same adaptive scheme selecting among different metaheuristics and/or considering different problem domains.

Acknowledgments. The authors acknowledge the support from CNPq (grants 140785/2009-4 and 308317/2009-2).

References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. Machine Learning 47(2-3), 235–256 (2002)
2. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Tech. rep., U. of Nottingham (2010)
3. Eshelman, L.J., Schaffer, J.D.: Real-coded genetic algorithms and interval-schemata. In: Foundation of Genetic Algorithms 2 (1993)
4. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme Value Based Adaptive Operator Selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)

5. Fialho, Á., Schoenauer, M., Sebag, M.: Toward comparison-based adaptive operator selection. In: Proc. Genetic Evol. Comput. Conf. (2010)
6. Fialho, Á.: Adaptive Operator Selection for Optimization. Ph.D. thesis, Université Paris-Sud XI, Orsay, France (December 2010)
7. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking: Noiseless functions definitions. RR-6829. Tech. rep., INRIA (2009)
8. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: Experimental setup. RR-7215. Tech. rep., INRIA (2010)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolutionary Programs. Springer (1992)
10. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyperheuristics. Intell. Data Anal. 12(1), 3–20 (2008)
11. Storn, R., Price, K.V.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Opt. 11, 341–359 (1997)
12. Sywerda, G.: Uniform crossover in genetic algorithms. In: Proc. of the Third Int. Conf. on Genetic Algorithms, San Francisco, CA, USA, pp. 2–9 (1989)
13. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proc. GECCO, pp. 1539–1546 (2005)