Evolvability Metrics in Adaptive Operator Selection

Jorge A. Soria-Alcaraz
Instituto Tecnologico de Leon
Leon, Mexico
soajorgea@gmail.com

Gabriela Ochoa
University of Stirling
Scotland, UK
goc@cs.stir.ac.uk

Martin Carpio
Instituto Tecnologico de Leon
Leon, Mexico
jmcarpio61@hotmail.com

Hector Puga
Instituto Tecnologico de Leon
Leon, Mexico
pugahector@yahoo.com

ABSTRACT

Evolvability metrics gauge the potential for fitness of an individual rather than fitness itself. They measure the local characteristics of the fitness landscape surrounding a solution. In adaptive operator selection the goal is to dynamically select from a given pool the operator to apply next during the search process. An important component of these adaptive schemes is credit assignment, whereby operators are rewarded according to their observed performance. This article brings the notion of evolvability to adaptive operator selection, by proposing an autonomous search algorithm that rewards operators according to their potential for fitness rather than their immediate fitness improvement. The approach is tested within an evolutionary algorithm framework featuring several mutation operators on binary strings. Three benchmark problems of increasing difficulty, Onemax, Royal Staircase and Multiple Knapsack are considered. Experiments reveal that evolvability metrics significantly improve the performance of adaptive operator selection, when compared against standard fitness improvement metrics.The main contribution is to effectively use fitness landscape metrics to guide a self-configuring algorithm.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Design.

Keywords

Combinatorial optimization, evolutionary algorithms, evolvability, fitness landscapes, hyper-heuristics, adaptive operator selection, self-* search

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada

http://dx.doi.org/10.1145/2576768.2598220.

1. INTRODUCTION

Adaptive operator selection [7, 13, 18, 21], hyper-heuristics [4] and adaptive memetic algorithms [16] are amongst recent terms describing search methodologies that autonomously configure themselves on the fly when solving a given instance of an optimisation problem. This is related to the more general topic of adaptive evolutionary algorithms [14] but is restricted here to the process of dynamically selecting (perturbation) search operators. These methods acknowledge that the usefulness of specific search operators can vary dynamically across the search process, and therefore are complementary to automatic parameter tuning or algorithm configuration tools, which use offline learning [3, 8] or meta-evolutionary algorithms [10] on a set of training instances to establish the (static) configuration to solve new instances. These two complementary types of approaches share the goal of increasing the effectiveness and generality of search methodologies, and thus reduce the role of the human expert in their design and configuration.

This article deals with adaptive operator selection (AOS) in the context of evolutionary algorithms. Two cooperating mechanisms are required in this process: *operator selection*, which defines how the next operator to be applied should be chosen according to its estimated quality; and *credit assignment*, which defines how to estimate operators quality based on the impact brought by their most recent application. The most common way of assigning credit is to account for the fitness improvement brought by the operators. That is, the fitness difference of the generated offspring with respect to a reference value, which is generally taken as the parent. Other mechanisms in the literature have considered diversity metrics for credit assignment [13, 21].

Our proposal is to use metrics based on local characteristics of the fitness landscape surrounding a solution to measure the impact of operators, and incorporate these metrics into the credit assignment mechanism. In order to measure landscape local characteristics, we consider *evolvability* metrics. Evolvability is loosely defined as the capacity to evolve, i.e. the ability of an individual to generate fitter variants than any yet existing [1, 17]. Therefore, it is more closely linked with the *potential* for fitness rather than with fitness itself; two individuals with equal fitness can have different evolvabilities [17].

Little work so far has considered the use of fitness landscape metrics to dynamically guide the search. An example is the use of neutrality and evolvability metrics to escape

plateaus when solving the flow shop scheduling problem [12]. To the best of our knowledge evolvability metrics have not been implemented specifically for adaptive operator selection, although this idea may be related to the the concepts of *look ahead* [5] and *delayed reward* [22] in meta and hyper-heuristics. To prove the concept and allow comparisons, we select the algorithmic framework (high-level search strategy and relevant parameter settings) used in recent work on adaptive operator selection [7]. Three benchmark problems with binary representation are considered. Namely, the simple Onemax problem (also used in [7]), the more elaborate Royal Staircase family of functions; and a more realistic combinatorial optimisation problem, the Multiple Knapsack problem. The goal is both to determine whether operators' evolvability metrics produce a performance advantage over fitness improvement metrics, and to study the adaptive dynamics of operators selected according to each type of metric.

The next section describes evolvability and how to measure it. The proposed approach is described in section 3. Section 4 describes the empirical setup, while section 5 reports and analyses the experimental results. Finally, section 6 summarises our findings and suggests directions for future work.

2. EVOLVABILITY

Evolvability loosely refers to the capacity of an individual or population to evolve. The first formalisation of this notion is attributed to Altenberg [1] who defined evolvability as "the ability of the genetic operator/representation scheme to produce offspring that are fitter than their parents." This is clearly necessary for adaptation in natural and artificial systems and therefore relevant to adaptive operator selection. Within evolutionary algorithms, evolvability was initially considered as a measure of performance. More specifically, "as the most *local* or finest-grained scale for measuring performance in an evolutionary algorithm" [1]. Later on, Smith et al. [17] linked evolvability to the notion of fitness landscapes. They developed a notion of evolvability based on the fitness distribution of the offspring of sampled solutions. By averaging evolvability over a set of equal fitness solutions, fitness *evolvability portraits* of a landscape were constructed and shown to describe general features of ruggedness, modality, and neutrality. Moreover, the study showed that some features of the fitness evolvability portraits for real evolutionary electronics search spaces are linked to the ease of finding good solutions.

In order to use the notion of evolvability in an algorithmic setting, we need to measure it. More precisely, we are interested in measuring the evolvability of specific search operators. This is equivalent to measuring the potential of the incumbent solution from the point of view of a given operator by considering the current solution's fitness and that of its neighbourhood according to the operator. Evolvability then give us an estimate of how much we can achieve with this pair operator-solution.

As discussed in a recent survey [11], several evolvability metrics have been proposed in the literature. After a preliminary wider exploration, we selected the two best performing metrics for discrete search spaces proposed in [17]. Namely, E_a, the probability of the offspring fitness being higher or equal to the parent fitness (Equation 1); and E_b, the mean fitness of the offspring solutions (Equation 2). We follow here the notation and definitions by Smith et al. [17]. The fitness landscape can be considered as a directed graph (V, E) with vertices V (solutions) connected by edges E. There is an edge between two solutions if one is generated from the other through a single application of a given operator. Let the pair $< h, k >$ denote a solution of genotype h, and fitness value k. The set G of offspring from a parent solution $< h, k >$ is thus determined by the vertices connected to the parent vertex: $G(< h, k >) = \{g \in V : E(< h, k >) = g\}$

The fitness function F maps every solution to a single $\mathbb{R}$ value. Then, we can define a set of offspring with fitness $F(g)$ greater than a given value:

$$G_c^+(< h, k >) = \{g \in V : E(< h, k >) = g, F(g) \geq c\}$$

E_a, is the probability of the offspring fitness being higher or equal to their parent fitness. It is simply measured as the ratio between the number of offspring with fitness higher or equal to that of the parent, and the total number of offspring (when sampling this is the size of the offspring sampling size). More formally it is defined as:

$$E_a = \frac{|G_k^+(< h, k >)|}{|G(< h, k >)|} \quad (1)$$

E_b, is the mean fitness of the offspring solutions (when sampling this is the mean fitness of the sample offspring). More formally it is defined as:

$$E_b = \frac{\sum_{g \in G(< h,k >)} F(g)}{|G(< h, k >)|} \quad (2)$$

For practically calculating the evolvability metrics defined above, a sampling methodology is required. A complete enumeration of the whole neighbourhood of the incumbent solution for each available operator would require impractical computational resources. We therefore consider estimated rather that exact evolvability metrics. Sampling the search space following a uniform distribution will give equal importance to all points. As discussed in [20], many solutions in the space may not be relevant from the point of view of evolutionary search. A methodology that gives prevalence to "important" points (i.e. those with higher fitness) is desirable as they reflect the bias followed by evolutionary and other heuristic search algorithms. As suggested in [20] for calculating *fitness clouds* we use the *Metropolis-Hastings* algorithm, which extends the standard Metropolis sampling to non-symmetric stationary probability distributions.

For estimating the evolvability metrics, E_a and E_b, the Metropolis-Hastings sampling method was used to collect information about the offspring produced from the incumbent solution given a search operator. A sampling size of 7 (offsprings) was used in our experiments. This value was found empirically to produce a good balance between fast computation and good metric approximation. A larger size would produce a more accurate estimate but would take longer, impairing the use of evolvability as a competitive metric for credit assignment.

3. TECHNICAL APPROACH

We investigate the use of the evolvability metrics described above to guide the dynamic selection of mutation operators in an evolutionary algorithm. An adaptive operator selec-

tion scheme consists of two components: (i) a *credit assignment* mechanism, which associates a reward with each operator, modeling its predicted utility; and (ii) a *selection rule*, which determines the operator to be used at each time step, as a function of reward. It is within the credit assignment that evolvability metrics are introduced as described in subsection 3.1. Subsection 3.2 describes the selection rules implemented, while subsection 3.3 the high-level search strategy used.

3.1 Evolvability based credit assignment

Most credit assignment methods compute operators' credit using the fitness of the new solution compared with that of its parent [7, 18]. We use solutions' evolvability instead of their fitness values. The motivation is to have an enriched view of the operator's quality according to the local characteristics of the fitness landscape surrounding the offspring and parent solutions. We consider *extreme* values for assigning credits [7], which is based on the principle that large (but possibly infrequent) credit improvements are more effective than small frequent improvements. It rewards operators which have had a recent large positive impact, while consistent operators yielding only small improvements receive less reward. Rewards are updated as follows, when an operator *op* is selected, it is applied to the current solution. The evolvability value of this new solution is computed and the change in evolvability is added to a list of size *W*. Thereafter, the operator reward is updated to the maximal value in the list. Table 1 reports the value used for this parameter in our experiments, while Figure 2 gives some evidence of the suitability of the selected value. The credit assignment mechanism needs to be combined with a selection rule as described below.

3.2 Selection rule

Operator selection probabilities are calculated from their quality estimates following a selection rule. These rules maintain a probability vector $(p_i, t)_{i=1,\ldots,K}$ (where K denotes the number of operators), and use the operator's raw credit estimate to calculate probabilities. Two recent and well performing selection rules, namely, *Adaptive Pursuit* and *Dynamic Multi-Armed Bandit*, were implemented and tested in our study. An overview of these two rules is presented below. These methods introduce control parameter. The parameter settings used in our experiments are reported in Table 1.

Adaptive pursuit was originally proposed for learning automata and was adapted to operator selection in [18]. It follows a winner-takes-all strategy, selecting at each step the operator with maximal reward, increasing its selection probability, while all other operators get their probability reduced. This method has two parameters: p_{min} that indicates the minimal probability of selection for each operator, and β, the *learning rate* taken from $(0, 1]$.

The multi-armed bandit framework is commonly used in game theory for studying the *exploration vs. exploitation* dilemma. It involves N arms and a decision making-algorithm for selecting one arm at each time step with the goal of maximising the cumulative reward gathered along time. The exploration vs. exploitation balance is also relevant for heuristic search. Indeed, adaptive operator selection can be formulated using multi-armed bandits with arms corresponding to search operators [6, 7]. Specifically, the *upper confidence bound* multi-armed bandit [2] was used as it provides optimal maximisation of cumulative rewards. Two considerations were required to use this framework for adaptive operator selection. First, a scaling factor C is needed, in order to properly balance the tradeoff between exploration and exploitation. Second, the original setting is static, while adaptive operator selection is dynamic, i.e., the quality of the operators is likely to change along the different stages of the search. The multi-armed bandit framework is thus combined with the *Page-Hinkley* statistical test for detecting changes in the reward distribution, and, upon such a detection, restarting the process [7]. This combined approach is termed *dynamic multi-armed bandit*, and introduces two additional parameters associated to the *Page-Hinkley* test: γ, which controls the trade-off between false alarms and unnoticed changes; and δ, which enforces the robustness of the test when dealing with slowly varying environments.

3.3 High-level search strategy

For comparison purposes, we implemented the algorithmic framework used in recent adaptive operator selection research [7]. Namely, a standard (1+λ)-EA without recombination, where λ offspring are created from a single parent, and the best individual among the current offspring and parent becomes the parent in the next generation. The AOS module is incorporated for choosing the mutation operator to generate the next offspring as indicated in Algorithm 1.

Algorithm 1 $(1 + \lambda) - EA$ with AOS.

Require: $f : \delta \to \mathbb{R}$ fitness function, $maxGener$ max number of generations, μ size of parent pool, λ size of the offspring pool

1: $ParentPool_0 = generatePopulation(\mu)$
2: **for** $i = 0$ until $maxGener$ **do**
3: initialize $OffspringPool_i = pool(\lambda)$
4: **for** $j = 0$ until λ **do**
5: $OffspringPool_i.add(AOS(ParentPool_{i-1}))$
6: **end for**
7: $ParentPool_i = Best(OffspringPool_i)$
8: **end for**
9: **return** $Best(ParentPool_i)$

4. EXPERIMENTAL SETUP

This section describes the experimental setup including test problems, algorithm variants and their parameter settings, the performance metric and statistical testing employed.

4.1 Test problems

Three benchmark problems encoded as binary strings are considered. The first benchmark function is the Onemax or counting ones problem, traditionally used in theoretical and proof of concept studies in genetic algorithms. We selected it as it was one of the benchmarks of choice in recent adaptive operator selection studies based on fitness improvement [7]. Two additional more complex test problems are also considered, as described below.

Royal staircase functions. This class of functions is related to the more familiar *Royal Road* functions [15]. They were proposed in [19] for analyzing epochal evolutionary search, that is the existence of long periods of fitness stasis punctuated by occasional fitness leaps. Although simple, Royal Staircase functions capture some essential ele-

ments found on complex problems, namely, highly degenerate genotype-to-phenotype maps, and the existence neutrality. A formal definition of the Royal Staircase class of functions is given below.

1. Genotypes are binary strings $s = s_1 s_2 \dots s_L$, $s_i \in \{0, 1\}$, of length $L = NK$, where N is the number of blocks and K the number of bits per block.
2. Starting from the first position, the number $I(s)$ of consecutive 1s in a string is counted.
3. The fitness $f(s)$ of string s with $I(s)$ consecutive ones, followed by a zero, is $f(s) = 1 + \lfloor I(s)/K \rfloor$. The fitness is thus an integer between 1 and $N+1$, corresponding to 1 plus the number of consecutive fully-set blocks starting from the left.
4. The single global optimum is $s = 1^L$; namely, the string of all 1s.

Fixing N and K determines a particular problem or fitness landscape.

Multiple knapsack problem. The combinatorial optimisation problem described here, called the 1/0 multiple knapsack problem, follows the formulation in [9]. This problem is a generalisation of the 0/1 simple Knapsack problem where a single knapsack of capacity C, and n objects are given. Each object has a weight w_i and a profit p_i. The objective is to fill the knapsack with objects producing the maximum profit P. In other words, to find a vector $x = (x_1, x_2, \dots, x_n)$ where $x_i \in \{0, 1\}$, such that $\sum_{i=1}^{n} w_i x_i \leq C$ and for which $P(x) = \sum_{i=1}^{n} p_i x_i$ is maximised.

The multiple version consists of m knapsacks of capacities $c_1, c_2, \dots, c_m$ and n objects with profits $p_1, p_2, \dots, p_n$. Each object has m possible weights: object i weighs w_{ij} when considered for inclusion in knapsack j ($1 \leq j \leq m$). Again, the objective is to find a vector $x = (x_1, x_2, \dots, x_n)$ that guarantees that no knapsack is overfilled: $\sum_{i=1}^{n} w_{ij} x_i \leq c_j$ for $j = 1, 2, \dots, m$; and that yields maximum profit $P(x) = \sum_{i=1}^{n} p_i x_i$.

4.2 Algorithm variants and parameter settings

The high-level search strategy is the (1+ λ)-EA (Algorithm 1) with offspring population size, $\lambda = 50$ in order to have the same high level algorithm and population size used in [7]. A common pool of mutation operators for binary representation was selected. The standard *n-flip* operator is considered, which chooses uniformly at random n bits in the current solution and flips their values (0 is changed to 1 an vice-versa). Our implementation used n values of 1, 3 and 5, as this was the choice of operators in [7].

For the Onemax problem six Adaptive Operator Selector (AOS) variants were considered by combining the three alternative credit assignment mechanisms: fitness improvement (*Fit*) and two evolvability metrics (E_a and E_b); with the two selection rules: adaptive pursuit (*ap*), and dynamic multi-armed bandits (*dmab*). These variants are identified with the pair of mechanisms separated by a hyphen. For example, E_a-*dmab* indicates a variant using credit assignment based on the evolvability metric E_a coupled with the *dmab* selection rule (see Figure 1). For the two remaining test problems, and due to space constraints, only the best performing evolvabibilty metric (E_a) and selection rule (*dmab*) are considered. This reduces the number of AOS variants to two, namely: E_a-*dmab* and *Fit-dmab*. Notice that variant *Fit-dmab*, corresponds to the algorithm proposed [7]. As a control, a variant that selects operators uniformly at random at each iteration (identified as *Random*) is also considered for all test problems.

Table 1 reports the parameter settings used in the experiments. The first 4 rows show the parameter values that are common to all the experiments. Namely, the offspring population size λ, the credit assignment window size W, the Metropolis-Hastings sampling size for calculating evolvability metrics, and the control parameter δ associated to the *dmab* rule (this value is suggested in [7]). The last 6 rows in Table 1 show the parameter values specific to each test problem. Namely, the maximum number of generations for the evolution strategy *MaxGener*, the chromosome length L, and the selection rule control parameters (C, γ for *dmab*; and β, p_{min} for *ap*). For the Onemax problem, these values follow the suggestions in [7]. For the remaining two test problems, the values were manually tuned.

Table 1: Parameter settings.

Parameter	*Onemax*	*Royal Staircase*	*M. Knapsack*
λ	50	50	50
W	30	30	30
S	7	7	7
dmab δ	0.15	0.15	0.15
MaxGener	1,000	10,000	5,000
L	10,000	100 to 700	50 to 105
dmab γ	100	75	80
dmab C	10	7	7.5
ap β	0.75	-	-
ap p_{min}	0.10	-	-

4.3 Performance metric and statistical testing

The performance metric is simply the value of the fitness function after the preselected fixed number of generations of the evolutionary strategy. The three test problems considered are maximisation problems. Therefore, the higher the value of the objective function at the end of the run, the better the performance of the algorithm variant. For each AOS variant and test problem instance, 33 independent runs were conducted and results are reported using descriptive statistics and box-plots summarising the performance distribution. Statistical inference (i.e. multiple comparison hypothesis testing) was used to asses the comparative performance of the AOS variants. Upon checking, our results were found to approximately follow a Normal distribution and have stable variances. Therefore, the use of parametric hypothesis testing is adequate. We used the one-way ANOVA F test (also called Omnibus test) for the Onemax problem, as there is a single factor (the AOS variant) that can explain performance differences in the unique Onemax instance considered. For the other two problems, namely, Royal Staricase and Multiple Knapksack, a two-way ANOVA F test is required as not only the AOS variant but also the specific problem instance can explain observed differences. In order to asses the statistical significance of the results, a pairwise t test is applied post-hoc to identify specific differences (if they exist) in a given pair of AOS variants. Furthermore, the *p-value* for each test is computed including a Bonferroni correction, which provides control over the so-called *family-wise* error rate. In our study the null hypothesis H_0 rep-

resents no significant performance differences between the algorithm variants. Finally, the Tukey HSD test is used to enforce t test results. All these tests were executed with the standard significance level of 0.05.

5. RESULTS AND ANALYSIS

We start by showing in Table 2 some basic descriptive statistics summarising the performance of the proposed algorithm *Ea-dmab* as compared with the state-of-the-art AOS variant *Fit-dmab* [7] and the control *Random* strategy across all the test problems. Results clearly indicate the superior performance *Ea-dmab*. The stopping condition in our experiments considers the number of generations of the of the evolutionary strategy. The calculation of evolvability through sampling, will incur in additional computational costs. However, as Table 3 indicates, running times were not greatly increased in our experiments.

Table 2: Descriptive statistics of main AOS variants across all test problems.

Onemax	Mean	Median	Std.dev	Best
Ea-dmab	**8074.12**	**8050**	**27.92**	**8200**
Fit-dmab	7832.52	7750	27.42	7957
Random	6834.53	6812	30.01	6892
R.StairCase	Mean	Median	Std.dev	Best
Ea-dmab	**16.45**	**16**	**16.03**	**19**
Fit-dmab	10.97	11	12.01	16
Random	10.78	10	13.63	15
M.Knapsack	Mean	Median	Std.dev	Best
Ea-dmab	**706600**	**718500**	**73105.9**	**863198**
Fit-dmab	495423.24	512432	77231.6	697673
Random	276634.77	269983	78332.4	35728

Table 3: Empirical running times in seconds on selected test instances.

Test Instance	*Fit*-dmab	E_a-dmab
$Onemax_{10,000}$	7.56	8.32
$Royal\ Staircase_{N40K5}$	7.48	12.75
$Royal\ Staircase_{N100K7}$	16.27	37.26
$Multiple\ Knapsack_{Weing7}$	0.84	1.12
$Multiple\ Knapsack_{Sento1}$	3.45	5.14

A more detailed statistical analysis of the results and the dynamic behaviour of the AOS variants is presented below for the three test problems.

5.1 Onemax

Figure 1 illustrates the magnitude and distribution of best fitness values at the end of the run for the 33 independent replicas on the Onemax instance. All the adaptive algorithms outperform the uniform random selection of operators. Clearly, the AOS variants using *dmab* outperform those using the *ap* selection rule. Within a fixed selection rule, the credit assignment mechanisms based on evolvability have the best performance, with the E_a metric consistently producing the best result. The one-way ANOVA F test, reported in Table 4, supports the existence of significant differences in the performance means given by the algorithms. A pairwise t test was also conducted with Bonferroni *p-value* corrections as shown at the bottom of Table 4. This shows evidence about the significant difference between algorithms. Specifically, we are interested in the performance of *Ea-dmab* as compared to the other AOS variants.

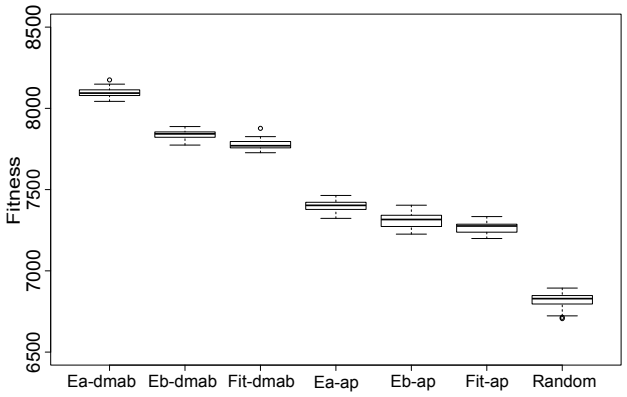

Figure 1: Onemax. Distribution of fitness values for all the AOS variants.

Table 4: Onemax. One-way ANOVA F test, and pairwise t test.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Algorithms	6	32928932	5488155	3748.5	1.68e-18
Residuals	217	317710	1464		
t **tests**	E_a-*ap*	E_a-*dmab*	E_b-*ap*	E_b-*dmab*	*Fit-ap*
E_a-*dmab*	**2.4e-16**	-	-	-	-
E_b-*ap*	1.2-16	**1.5e-17**	-	-	-
E_b-*dmab*	2e-9	2.**4e-12**	3.2e-9	-	-
Fit-ap	2.4e-8	**4.3e-18**	1.5e-09	2e-16	-
Fit-dmab	2.1e-16	**3.3e-16**	2.6e-16	1	4.15e-6

Figure 2 illustrates the magnitude and distribution of best fitness at the end of the run of the best performing AOS variants E_a-*dmab* and *Fit-dmab*, with different window sizes $W = \{10, 30, 50\}$. The *Random* AOS variant is also included for comparison purposes. The window size is a parameter of the extreme-value credit assignment (described in Section 3.1) indicating the memory size of previous operator's credit. Results indicate that a window size of 30 consistently produced good results, not only in the Onemax problem as illustrated in Figure 2 but in the other test problems also (not shown here due to space constraints). It seems that a small window does not give enough scope for estimating the operator's reward, while a large one maintains historic information that is no longer relevant.

Figure 3 shows the dynamic of operators' probabilities autonomously adapted through an execution for *Fit-dmab* (top) and E_a-*dmab* (bottom). The curves show a similar behaviour for both variants. A clear transition or abrupt change (most probably detected by the *dmab* mechanism) is observed around iteration 750 in both cases. The transition occurs slightly early for the fitness-based mechanism suggesting that evolvability is a more reliable indicator of change. Moreover, after the transition, the E_a based rule increases the selection probability of operator *1-flip* and decreased that of operator *3-flip* at a higher rate than the fitness based rule.

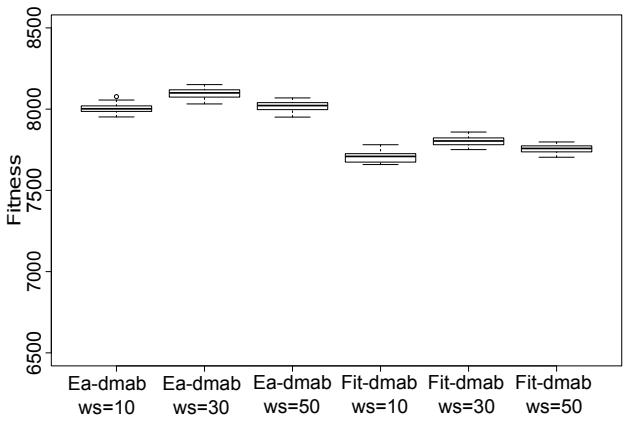

Figure 2: Onemax. Fitness distributions illustrating the effect the window size (W) parameter.

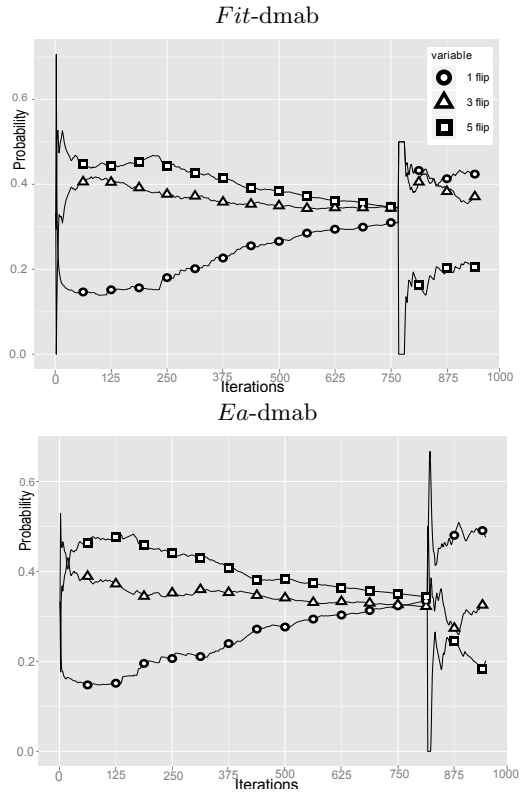

Figure 3: Onemax. Dynamic of adapted operators' probabilities over a run.

5.2 Royal Staircase

Royal Staircase functions are always unimodal, but we can increase the landscape ruggedness by decreasing the number of blocks N. Modifying the number of blocks also alters the overall shape. Four combinations of N and K were selected for the experiments in this subsection (i.e. the following <N,K> pairs: <20,5>, <40,5>, <80,5>, and <10,7>), with string lengths ($N * K$) varying from 100 to 700.

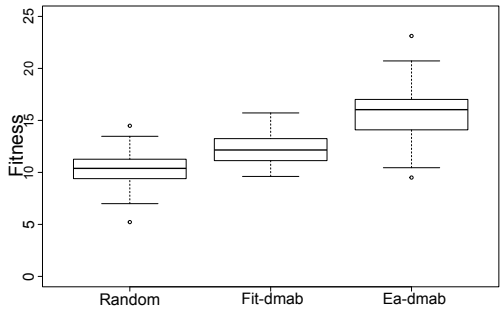

Figure 4: Royal Staircase. Post-hoc boxplot with performance distribution across all instances.

The two-way ANOVA F test combining the 4 test instances and 3 algorithm variants, reported in Table 5, supports the the existence of significant differences in the performance means of the studied AOS variants. Further analysis is required to identify which pairs of algorithms are significantly different. This is achieved with the pairwise t test with Bonferroni adjustments reported at the the middle portion of Table 5. The test supports that the E_a-*dmab* variant has significant performance differences as compared to the other variants. Finally, the Tukey HSD test with *family-wise* confidence level of 95 % (reported at the bottom of Table 5) supports this evidence. Figure 4 shows box-plots with post-hoc performance of the AOS variants. Clearly, our approach *Ea-dmab* outperforms the other variants.

Table 5: Royal Staircase. Two-way ANOVA F test, pairwise t test and Tukey HSD test.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Algorithm	2	950.1	475.05	42.222	**4.16e-15**
Instance	3	4990.3	1663.42	147.841	1.23e-16
Residuals	390	4388.1	11.25		

t **Tests**	E_a-*dmab*	*Fit-dmab*
Fit-dmab	**2.1e-6**	-
Random	**3.9e-8**	0.016

TukeyHSD	diff	lwr	upr	**p adj**
Fit vs E_a	-3.03	-4.0	-2.05	**0.00**
Rand vs E_a	-3.49	-4.46	-2.52	**0.00**
Rand vs Fit	-0.46	-1.43	0.50	0.023

Figure 5 shows the dynamic of operators' learned probabilities across a run for *Fit-dmab* (top) and E_a-*dmab* (bottom) on the $N = 80, K = 5$ Royal staircase case instance. The curves show a different behaviour for both variants. The evolvability, E_a based variant (bottom plot), shows a varying selection dynamic of the tree operators, while the fitness based variant seems to converge to fixed probabilities since very early in the run.

5.3 Multiple Knapsack

Four multiple-knapsack instances, taken from the literature, were selected as test problems. These are multimodal constrained problems with sizes ranging from 50 to 105 objects and from 2 to 30 knapsacks. These (and several

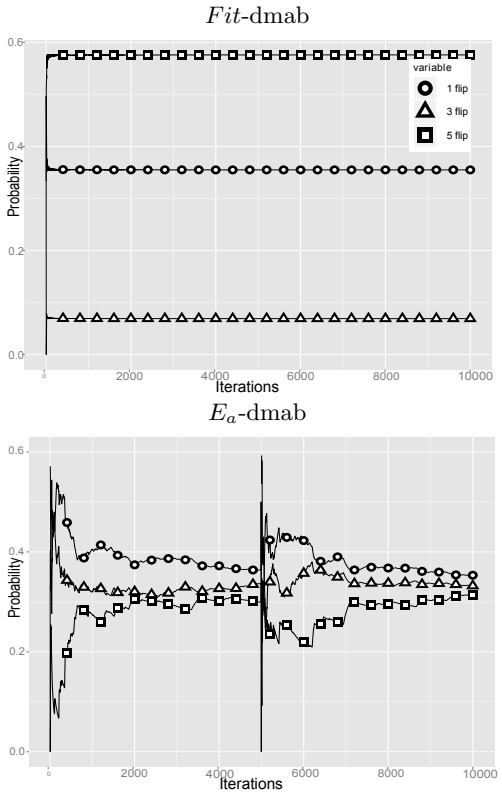

Figure 5: Royal Staircase ($N = 80, K = 5$). Dynamic of adapted operators' probabilities over a run.

other) problems are available online from the OR-library by Beasley[1].

Table 6 reports the two-way ANOVA F test (top part of the Table), and the pairwise t test with Bonferroni adjustments (middle portion of the Table). The tests support that the E_a-*dmab* variant has significant performance differences as compared to the other variants. The Tukey HSD test with *family-wise* confidence level of 95 %, also conducted and reported at the bottom of Table 6, supports this evidence. Figure 6 shows box-plots with post-hoc performance of the AOS variants. Our approach *Ea-dmab* also outperforms the other variants in the Multiple Knapsack problem.

Table 6: Multiple Knapsack. Two-way ANOVA F test, pairwise t test and Tukey HSD test.

ANOVA	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Algorithm	2	6.46	3.23	9.33	**1.0-4**
Instance	3	8.42	2.80	8.11	2.2e-16
Residuals	390	1.35e10	3.46e7		
t Tests	E_a-*dmab*		*Fit-dmab*		
Fit-dmab	**3.12e-3**		-		
Random	**8.52e-5**		1.11e-2		
TukeyHSD		diff	lwr	upr	**p adj**
Fit vs Ea		-986	-2690	717	**3.27e-3**
Ran vs Ea		-3066	-4770	-1362	**8.66e-5**
Rand vs Fit		-2079	-3783	-375	0.0012

[1]The OR Library is available at `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

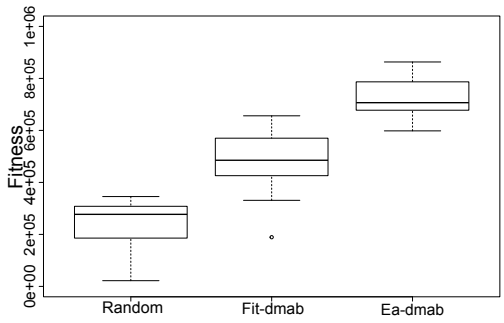

Figure 6: Multiple Knapsack. Post-hoc boxplot with performance distribution across all instances.

Figure 7 shows the dynamic of operators' learned probabilities across a run for *Fit-dmab* (top) and E_a-*dmab* (bottom) on the *Weish30* multiple knapsack instance. The curves for both plots sown a simliar overall behaviour; with the E_a based variant (bottom plot) more sharply detecting probability distribution changes. This, together with the observed improved performance, seems to indicate that the evolvabilty metric is producing a richer feedback to guide adaptive operator selection.

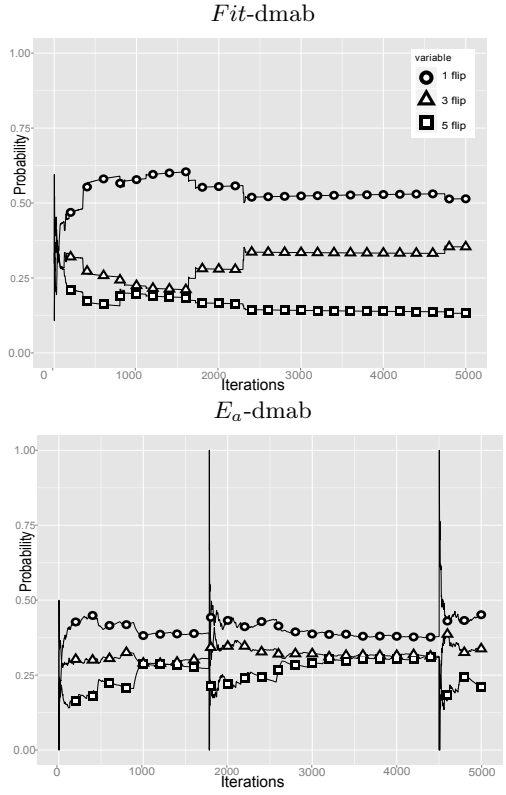

Figure 7: Multiple Knapsack ($Weish30$). Dynamic of adapted operators' probabilities over a run.

6. CONCLUSIONS

We investigated the benefits of using evolvability metrics to inform the adaptive selection of operators. Evolvability metrics are estimated using a Metropolis-Hasting algorithm, which reduces the computational complexity of their calculation. Results on binary-based combinatorial problems reveal that assigning credits to operators based on their evolvability rather than on their direct fitness improvement produces statistically significant improved results. This suggest that considering the fitness landscape local structure of combinatorial problems provides a richer picture to guide the self-configuration of heuristic search algorithms. From the two selection rules tested, the dynamic multi-armed bandit produced better performance, suggesting that it is critical to be able to detect and act upon abrupt changes in the local structure of the fitness landscape. Our research contributes to the goal of using fitness landscape local features to inform dynamic self-configuring algorithms.

Future work will test the proposed approach on real-world combinatorial optimisation problems such as educational timetabling and vehicle routing. Our study considered a standard evolution strategy with a pool of mutation operators as the high-level search strategy. It is worth stressing that adaptive operator selection methods are relevant to any heuristic search strategy or algorithmic component where a pool of search operators is available, be they mutation, recombination, memes (local searchers) or construction-destruction heuristics. This research is therefore relevant to selective hyper-heuristics, adaptive large neighbourhood search and memetic algorithms. Further work will then explore how to incorporate the proposed adaptive mechanism within other algorithmic frameworks.

Acknowledgements

This work was supported by Consejo Nacional de Ciencia y Tecnología (CONACYT) México under PhD Scholarship 311849 and the Engineering and Physical Sciences Research Council (EPSRC - grant number EP/J017515), UK.

7. REFERENCES

[1] L. Altenberg. The evolution of evolvability in genetic programming. In *Advances in Genetic Programming*, pages 47–74. MIT Press, Cambridge, MA, USA, 1994.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, pages 11–18. Morgan Kaufmann, 2002.

[4] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society (JORS)*, 64(12):1695–1724, 2013.

[5] M. Caserta, S. Schwarze, and S. Voß. A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg, 2009.

[6] L. Da Costa, A. Fialho, M. Schoenauer, and M. Sebag. Adaptative operator selection with dynamic multi-armed bandits. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 913–920. ACM, 2008.

[7] A. Fialho, L. Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *Learning and Intelligent Optimization*, volume 5851 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2009.

[8] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR)*, 36:267–306, 2009.

[9] S. Khuri, Thomas Bäck, and Jörg Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In *Proceedings of the 1994 ACM Symposium of Applied Computation*, pages 188–193. ACM Press, 1994.

[10] S. Luke and AKM K. A. Talukder. Is the meta-ea a viable optimization method? In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 1533–1540, New York, NY, USA, 2013. ACM.

[11] K. M. Malan and A. P. Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241(0):148 – 163, 2013.

[12] M-E. Marmion, C. Dhaenens, L. Jourdan, A. Liefooghe, and S. Verel. On the neutrality of flowshop scheduling fitness landscapes. In *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 238–252. Springer Berlin Heidelberg, 2011.

[13] J. Maturana and F. Saubion. A compass to guide genetic algorithms. In *Parallel Problem Solving from Nature, PPSN X*, pages 256–265. Springer, 2008.

[14] S. Meyer-Nieberg and H-G. Beyer. In *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 47–75. Springer Berlin Heidelberg, 2007.

[15] M. Mitchell, S. Forrest, and J. H. Holland. The Royal Road for genetic algorithms: Fitness landscapes and GA performance. In *Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems*, pages 245–254. MIT Press, Cambridge, MA, 1992.

[16] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(1):141–152, 2006.

[17] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34, 2002.

[18] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1539–1546, New York, NY, USA, 2005. ACM.

[19] E. van Nimwegen and J. P. Crutchfield. Optimizing epochal evolutionary search: Population-size dependent theory. Technical Report Preprint 98-06-046, Santa Fe Institute, 1998.

[20] L Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Verel. Fitness clouds and problem hardness in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation (GECCO 2004)*, volume 3103 of *Lecture Notes in Computer Science*, pages 690–701. Springer Berlin Heidelberg, 2004.

[21] N. Veerapen, J. Maturana, and F. Saubion. An exploration-exploitation compromise-based adaptive operator selection for local search. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 1277–1284. ACM, 2012.

[22] T. Wauters, K. Verbeeck, P. Causmaecker, and G. Berghe. Boosting metaheuristic search using reinforcement learning. In *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 433–452. Springer Berlin Heidelberg, 2013.