A Dynamic Island Model for Adaptive Operator Selection

Caner Candan
LERIA - University of Angers
Angers, France
caner.candan@univ-angers.fr

Adrien Goëffon
LERIA - University of Angers
Angers, France
adrien.goeffon@univ-angers.fr

Frédéric Lardeux
LERIA - University of Angers
Angers, France
frederic.lardeux@univ-angers.fr

Frédéric Saubion
LERIA - University of Angers
Angers, France
frederic.saubion@univ-angers.fr

ABSTRACT

In this paper we propose a generic framework for Dynamic Island Models, which can be used as an original approach for the adaptive selection of operators in evolutionary algorithms. Assigning a variation operator to each island, we show that the dynamic regulation of migrations, which takes into account the pertinence of recent migrations, distributes the individuals on the most promising islands, i.e., the most efficient operators, at each stage of the search. The efficiency of this approach is assessed on the One-Max problem by comparing theoretical expected results to those obtained by our dynamic island model. Experiments show that the model provides the expected behavior.

Keywords

Island Models, Adaptive Operator Selection, Evolutionary Computation

Categories and Subject Descriptors

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

General Terms

Algorithms.

1. INTRODUCTION

Evolutionary algorithms (EA) have been widely used for tackling NP-hard problems [17, 7, 4]. Basically, an EA manages a population of individuals encoding possible configuration of the problem, in order to optimize a given fitness function. These individuals evolve by means of variations operators and selection processes. Although the efficiency of EAs is well-established on numerous optimization problems, their performance and robustness may depend on the correct setting of its components. Moreover, these technical choices often lead to the design of *ad hoc* EAs, dedicated to specific problem instances, and become out of the scope of non-specialist users. Therefore, parameter setting in EA has deserved much attention during recent years [15] in order to provide more generic and adaptive algorithms. Parameter setting may indeed be considered from two complementary points of view: the design of a suitable EA for a given problem and the improvement of the behavior of an EA to reach an acceptable performance. While the design will focus on the structural parameters of the EA such as its variation operators, its behavioral parameters, such as the application probabilities of these operators, will be adjusted to improve its solving efficiency. This efficiency greatly depends on the management of the balance between exploration and exploitation of the search space, which relies on the correct combination of suitable components with suitable behavioral parameters values. As suggested in [8], we distinguish parameter tuning (off-line setting before solving) from parameter control (on-line setting during solving). For instance, one may use automated tuning algorithms [18, 13] in order to adjust parameters by running the EA on test instances, before using it for solving new problems. In this paper we focus on the on-line control of the variation operators of an EA [8]. Adaptive operator selection (AOS) [10] consists in providing an adaptive mechanism for selecting the suitable variation operator to apply on the individual, at each iteration of the EA process. Recent approaches [11, 16] have proposed such adaptive mechanisms for general EA with possibly many variation operators, whose behavior may be unknown. However, as suggested by the *No Free Lunch* theorems [25], it is difficult to anticipate the efficiency of an algorithm on any instances of a wide class of problems without preliminary experiment or learning process. While most of the works on AOS (e.g., [22, 11, 16]) provide adaptive control mechanisms for an EA with multiple operators, another possible approach is to consider several single-operator EAs and to address the global management of this set of algorithms by sharing and exchanging information through individuals. The contribution of this paper is thus to use Islands Models in order to achieve such an adaptive operators management for EAs.

Island Models [24] consider simultaneously a set of populations, clustered on islands, which are evolving independently during some search steps and interacting periodically. This model, which constitutes an additional abstraction level in

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

comparison to classical EAs, provides an improved management of the diversity and simplify the parallel implementation of EAs.

Most of the time, island models are used in a static way, where individuals are migrating from populations to populations following a fixed predefined scheme [19], or are specifically chosen in order to reinforce the populations characteristics [20, 12, 1]. Nevertheless, it is possible to dynamically regulate migrations between islands by considering a transition matrix [14]. Such a model is used to increase or decrease the migration probabilities during the evolutionary process according to the impact of previous analogue migrations. The purpose is to control migration in order to dynamically regulate the diversity of the individuals in the populations, according to the search progress, and, consequently, to control the population sizes. In classical uniform island models, each island uses the same EA and differs only by its individuals. Considering now a different algorithm on each island, a dynamic model allows to regulate interactions between individuals or groups of individuals.

In this paper, we propose to generalize and extend classical island representations in order to take into account the dynamic nature of the model, as well as a possibly asynchronous solving processes. We show then that such a model can be used to achieve an adaptive selection of the most pertinent operators. To this end, we assign to each island a different variation operator, and we let the model distribute the individuals on the most promising islands, according to recent background information. Moreover, the proposed model does not require any global migration processing, and policies are updated with feedback information shared between islands.

An appropriate and adaptive regulation of migration flows should assign dynamically the resources to the most pertinent operators during the search process. In our experiments, we use EAs with basic mutation operators; the objective is thus not to regulate interactions between individuals, but to provide an adaptive control mechanism that assigns promising individuals to the most promising islands, *i.e.*, operators.

In the following, we first introduce an original generic representation of dynamic island models. Then we describe a specific instantiation of the model for the adaptive selection of operators in EAs and we assess the relevance of our approach on the One-Max problem by checking that our dynamic model allows an operator selection that fits the theoretically best sequence of operators for solving this problem.

2. GENERIC DYNAMIC ISLAND MODEL REPRESENTATION

Since twenty years and the first distributed evolutionary algorithms [21], island-based genetic algorithms (or island models [24]) are more and more studied in evolutionary computation. The main original problem consists in defining the model topology and the migration policies in order to reduce premature convergence of the population and to insure a global sharing of promising individuals. Numerous migration policies and model topologies have been proposed (see [2, 3, 23, 5, 9, 12, 1, 19]), and it is not obvious to figure out which topology and policies are the most suitable for a given purpose.

Let us so consider a previously proposed island model, which dynamically supervises the commonly-used EA parameters [1] such as population size, migration policy for the individuals, selection policy for immigrants or the topology of the communication between populations. A n-island model topology can be represented by a complete labeled digraph (see example in Figure 1). Migration policies are given by a transition (stochastic) matrix T of size n, where $T(i, j)$ represents the probability for an individual to migrate from island i to island j (or to stay on the same island if $i = j$). In order to make the model (possibly) independent to any global processing, T is actually given by n vectors that represent the migration policies for each island $V^i, i \in \{1, \cdots, n\}$, with $\Sigma_{j=1}^{n} V_j^i = 1$.

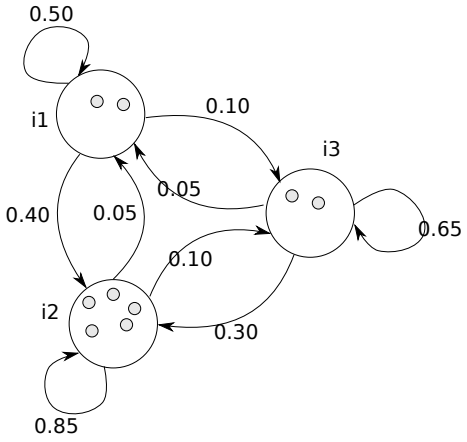


Figure 1: Island Model as a Complete Digraph

In this dynamic adaptation process, one has to determine pertinent migration probabilities at each step of the search process, considering a classical multi-population based EA. The dynamic regulation of the migration policies may result in different population sizes for the different islands, which prevent from assigning too much computational resources to poor-quality sub-populations or islands. However, if different islands use different variation operators, the control process should dynamically provide a well-balanced distribution of the individuals according to these operators and the current search state. As mentioned above, this can obviously be viewed as an adaptive operator selection process, which will be further detailed in Section 2.3.

2.1 Island processes

Figure 2 describes the mechanism that we propose for a generic Dynamic Island Models (DIM).

The vertical reading highlights the three divided layers defined as follows:

- upstream: the input part of the DIM with the received feedback data and individuals followed by a learning process;
- midstream: the evolution process (basic EA on the island) that is applied on the sub-population given as parameter;
- downstream: the output part of the DIM that uses intermediate functions (analysis of the sub-population and transition vector updating) before distributing the data to the other islands.

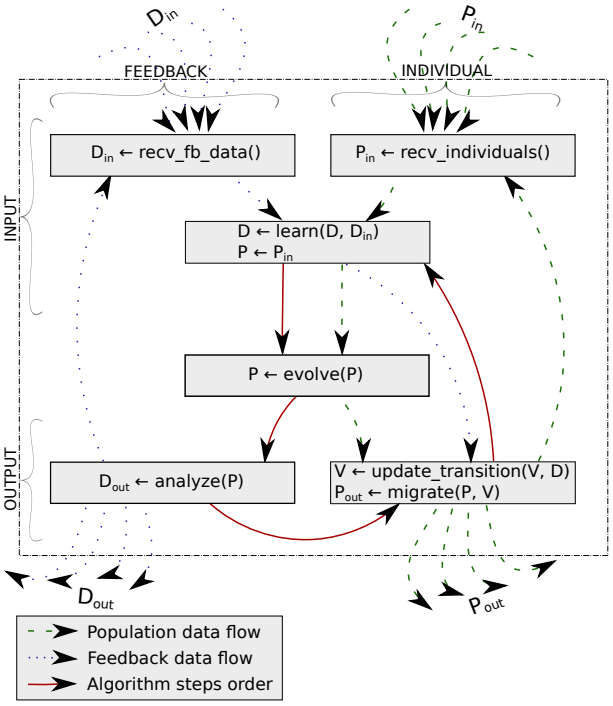


Figure 2: Dynamic island mechanism: population flow, feedback data flow and order of the basic steps.

The left-side of the figure shows the feedback data flow processes while the right-side shows the individuals flow processes.

Note that this DIM overall process is iterated on each island into a loop. The different blocks are ordered according to the arrows: learn, evolve, analyze and migrate. Each block is now described in Section 2.2.

2.2 Blocks description

$D_{in} \leftarrow$ **recv_fb_data():** this continuously running process collects the feedback data from other islands.

$P_{in} \leftarrow$ **recv_individuals():** this continuously running process collects individuals migrating to (or staying on) the current island.

Both `recv_fb_data()` and `recv_individuals()` functions do not cause any break during the execution of the main loop process since they should be processed in parallel. They fill out their own stacks each time they receive feedback data as well as new migrated individuals from other islands. An equivalent process can be simulated sequentially.

$D \leftarrow$ **learn**$(D,\ D_{in})$**:** the current knowledge vector D (n values representing the comparative pertinence of the last migrations, for all islands) is updated using data received since its last update that are recorded in D_{in}. Note that D_{in} may not have a fixed size.

$P \leftarrow$ **evolve**(P)**:** a basic EA process is applied to the subpopulation P.

$D_{out} \leftarrow$ **analyze**(P)**:** according to the origin of individuals, feedback information characterizing the pertinence of last migrations are sent to corresponding islands.

$V \leftarrow$ **update_transition**$(V,\ D)$**:** reward/penalty values given by D are used to update V with the following reinforcement learning process:

$$V = (1-\beta)(\alpha.V + (1-\alpha).D) + \beta.N$$

where N is a stochastic vector ($\Sigma_{j=1}^{n} N_j = 1$) with random values. α represents the importance of the knowledge accumulated during the last migrations (inertia or exploitation) and β is the amount of noise, which is necessary to explore alternative search space areas by means of individuals.

$P_{out} \leftarrow$ **migrate**$(P,\ V)$**:** each individual migrates to an island regarding to V (in this representation, the particular case of staying on the same island is also considered as a migration). P_{out} is a vector of n sets of individuals.

2.3 DIM for Adaptive Operators Selection

The generic DIM introduced in this section (2) can easily be instantiated in order to manage classical evolutionary algorithms like genetic, memetics or population-based local search algorithms. As mentioned in introduction, DIM can be used for the adaptive operator selection in an EA. This can be achieved with a specific instantiation of the model proposed in Figure 2. Each island represents a particular operator and has the following specifications:

- The *evolve* process uses the operator assigned to this island.
- The *analyze* process computes the feedback information and sends it to each islands (including its own). In our case, this information will be the average improvement of all individuals in function of their previous localization, during the last evolution step.
- The *learn* process receives feedback information from all other islands. D is a *reward vector* computed using an *intensification* strategy: only the best island is rewarded.

$$D[j] = \begin{cases} 1/|B| & \text{if } j \in B, \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{with } B = \underset{j'}{\operatorname{argmax}}\, D_{in}^{j'}$$

where $D_{in}^{i} \subseteq D_{in}$ are feedback values coming from island i

3. VALIDATION: ONE-MAX PROBLEM

The One-Max problem is a simple and well-known problem, already used to assess the performance of Adaptive Operator Selection mechanisms [11, 6]. The n-bits One-Max problem considers n-length bit strings; starting from 0^n individuals (*i.e.* strings made up of n *zeros*), the aim is to maximize the number of *ones*, that is to reach the 1^n bit string. The *score* of a bit string x, noted $|x|_1$, corresponds to its number of *ones*.

3.1 One-Max Mutation Operators

Recent works [11, 6] use four mutation (or local search) operators:

- *bit-flip*, which flips every bit with probability $1/n$,
- and *k-flip* (with $k = \{1, 3, 5\}$), which flips exactly k bits.

In the following and depending on the context, bit-flip and k-flip will denote the mutation operator as well as the corresponding neighborhood relation (according to well-know Hamming distance). k-flip can easily be defined by a neighborhood relation $\mathcal{N}_k : \{0,1\}^n \to 2^{\{0,1\}^n}$ such as $x' \in \mathcal{N}_k(x)$ if and only if $|h(x,x')| = k$ (where h is the Hamming distance). It is more difficult to describe the bit-flip operator using a neighborhood relation, since it corresponds to a complete neighborhood with a non-uniform move probability. In Proposition 2 we will show however how one bit-flip move can be reduced in one k-flip move with a determined probability of choosing k.

Intuitively, the 5-flip operator will be more efficient when applied on weak individuals (containing mostly *zeros*), while 1-flip will improve with a higher probability individuals with a high proportion of *ones*. This point is corroborated by the following theoretical comparisons of operators.

3.2 Probability of Improvement

PROPOSITION 1. *Let $x \in \{0,1\}^n$ and $x' \in \mathcal{N}_k(x)$ two bit strings of scores $s = |x|_1$ and $s' = |x'|_1$. We have:*

$$p(s' > s) = \sum_{\substack{i=\max(\lfloor k/2 \rfloor + 1,\\ k-s)}}^{\min(k,n-s)} \frac{C_{n-s}^i C_s^{k-i}}{C_n^k}$$

This is the probability to improve an individual x of size n and score s with a k-flip mutation.

PROOF. Given x a bit string of length n and by definition of the k-flip operator: $\{x', |h(x,x')| = k\} = \mathcal{N}_k(x)$. If $x' \in \mathcal{N}_k(x)$ and $|x'|_1 > |x|_1$, then more *zeros* than *ones* have been flipped in x. The number of *ones* in x is $|x|_1 = s$. The probability to flip exactly i *zeros* with a k-flip operator ($0 \leqslant i \leqslant k$) is equal to the probability of selecting i *zeros* and $k-i$ *ones*, that is : $\frac{C_{n-s}^i C_s^{k-i}}{C_n^k}$. The move is improving if $i > k-i$, *i.e.* $i > k/2$, so $i \geqslant \lfloor k/2 \rfloor + 1$. Consequently, the probability of selecting more *zeros* than *ones* is the sum of probabilities of selecting exactly i *zeros* for all $i \geqslant \lfloor k/2 \rfloor + 1$ (improving moves) such that $k - s \leqslant i \leqslant n - s$ (possible moves). □

PROPOSITION 2. *Considering the n-bit One-Max problem, the bit-flip operator consists in applying the k-flip operator with the probability $C_n^k \frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k}$.*

PROOF. The bit-flip operator flips each bit with the probability $1/n$. Consequently, the probability of flipping exactly k bits among n is $C_n^k \frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k}$ (binomial distribution). □

PROPOSITION 3. *Let $x, x' \in \{0,1\}^n$ two bit strings of scores $s = |x|_1$ and $s' = |x'|_1$, such that x' is generated from x with the bit-flip operator. We have*

$$p(s' > s) = \sum_{k=1}^{n} \left(\frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k} \sum_{\substack{i=\max(\lfloor k/2 \rfloor + 1,\\ k-s)}}^{\min(k,n-s)} C_{n-s}^i C_s^{k-i} \right)$$

This is the probability to improve an individual x of size n and score s with a bit-flip mutation.

PROOF. The probability to improve an individual in flipping exactly k bits is given by the proposition 1 and the k-flip operator. Since every k-flip moves ($0 \leqslant k \leqslant n$) are independent, the bit-flip improvement probability can be calculated in adding up the probability of improvement for each k, weighted by the probability to flip k bits with a bit-flip move (see Proposition 2). Consequently,

$$\begin{aligned} p(s' > s) &= \sum_{k=1}^{n} \left(C_n^k \frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k} \sum_{i=\lfloor k/2 \rfloor + 1}^{k} \frac{C_{n-s}^i C_s^{k-i}}{C_n^k} \right) \\ &= \sum_{k=1}^{n} \left(\frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k} \sum_{i=\lfloor k/2 \rfloor + 1}^{k} C_{n-s}^i C_s^{k-i} \right) \end{aligned}$$

with $k - s \leqslant i \leqslant n - s$. □

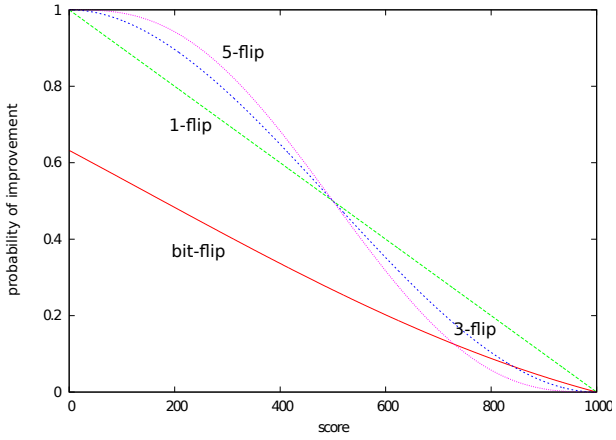


Figure 3: Probability to improve an individual for the 1000-bits One-Max problem using the flip operators: bit-flip, 1-flip, 3-flip and 5-flip.

3.3 Improvement Score

During the search, only improving mutations are effective while deteriorating ones provide reverse effects. So the improvement score of a mutation $x \to x'$ (x' results from a mutation applied on x) can be formalized by:

$$\Delta s(x, x') = \max(0, |x'|_1 - |x|_1)$$

PROPOSITION 4. *The expected value of the improvement score of a k-flip mutation on an individual $x \in \{0,1\}^n$ of score $s = |x|_1$ is equal to:*

$$\sum_{\substack{i=\max(\lfloor k/2 \rfloor + 1,\\ k-s)}}^{\min(k,n-s)} (2i - k) \frac{C_{n-s}^i C_s^{k-i}}{C_n^k}$$

PROOF. Let $x \in \{0,1\}^n$, $x' \in \mathcal{N}_k(x)$, $s = |x|_1$, $s' = |x'|_1$. Let i the number of *zeros* which are flipping during the k-flip mutation $x \to x'$. The number of *ones* in x' (s') is equal to the number of *ones* in x (s), minus the number of *ones* flipped ($k - i$), plus the number of *zeros* flipped (k). $s' = s - (k - i) + i = s - k + 2i$, *i.e.*, $s' - s = 2i - k$. If $i \leqslant \lfloor k/2 \rfloor$, then $s' \leqslant s$, and $\Delta s(x, x') = 0$. Otherwise, $\Delta s(x, x') = 2i - k$. The expected score can be deduced using Proposition 1 and its proof. □

PROPOSITION 5. *The expected value of the improvement score of a bit-flip mutation on an individual $x \in \{0,1\}^n$ of score $s = |x|_1$ is equal to:*

$$\sum_{k=1}^{n} \left(\frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k} \sum_{\substack{i=\max(\lfloor k/2 \rfloor + 1,\\ k-s)}}^{\min(k,n-s)} (2i-k) C_{n-s}^{i} C_{s}^{k-i} \right)$$

PROOF. This result follows from the three previous propositions. □

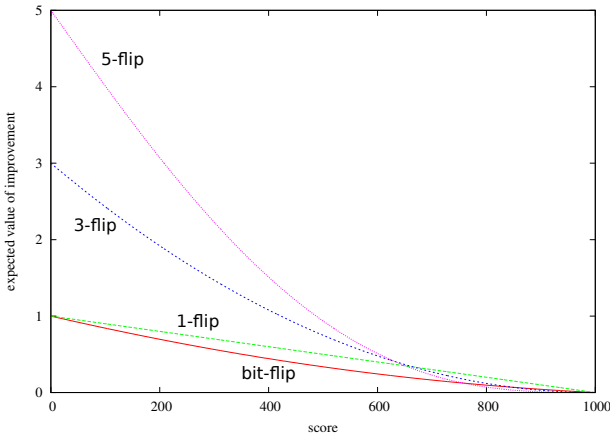


Figure 4: Expected value of the improvement scores for the 1000-bits One-Max problem.

3.4 Most Accurate Operators

DEFINITION 1. *Let $x \in \{0,1\}^n$ an individual of score s, $M = \{\mu_1, \ldots, \mu_l\}$ a mutation operators set and $x_1, \ldots, x_l$ l neighbors of x such that $x \xrightarrow{\mu_k} x_k$ (x_k results from the mutation on x by μ_k). $s_1, \ldots, s_l$ are the respective scores of individuals $x_1, \ldots, x_l$. We define the domination rate $\chi(\mu_k, M, x)$ of an operator μ_k on M in function of x, the probability $p(\Delta s(x, x_k) > \Delta s(x, x_{k'}), \forall k' \in \{1, \ldots, l\} \setminus \{k\})$, which is equal to:*

$$\sum_{i=1}^{n} \left(p(\Delta s(x, x_k) = i) \prod_{k' \neq k} \sum_{j=0}^{i-1} p(\Delta s(x, x_{k'}) = j) \right)$$

For k-flip operators (x', of score s', is the transformation of x, of score s, by k-flip), we have (see Proposition 1):

$$p_k(\Delta s(x, x') = i) = \begin{cases} \frac{C_{n-s}^{i} C_{s}^{k-i}}{C_n^k} & \text{if } i > 0, \\ 1 - p(s' > s) & \text{if } i = 0. \end{cases}$$

For bit-flip operator, $p_{\text{bit}}(\Delta s(x, x') = i) =$

$$\sum_{k=1}^{n} \left(C_n^k \frac{1}{n^k} \left(1 - \frac{1}{n}\right)^{n-k} p_k(\Delta s(x, x') = i) \right)$$

The domination rates evolution of the four considered operators in function of the score of an individual is shown in Figure 5 (with $M = \{$ 1-flip, 3-flip, 5-flip, bit-flip $\}$).

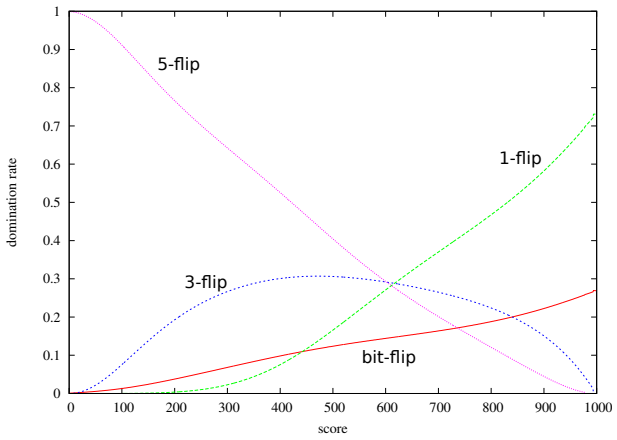


Figure 5: Domination rates evolution for the 1000-bits One-Max problem.

4. EXPERIMENTAL STUDY

In this section we show that the behavior of our DIM is very close of the expected theoretical results. Moreover, we remark that it is not very dependent of its own parameter tuning.

4.1 Theoretical vs Empirical Results

As described in Section 3, we start with a population of 0^n individuals. The expected behavior during the search is to use the 5-flip operator when the population quality is weak (at the beginning), then the 3-flip operator and finally the 1-flip operator when the population quality is sufficiently high. In our experiments, this can be observed though the sub-population sizes of each island with respect to the migrations. The more an island attracts individuals, the more its operator is used.

Parameters for this experiment are:

- number of islands: 4 (one for each operator)
- population size: 400
- initial probabilities of migrations: 1 to stay in the same island
- (α,β): (0.8,0.1)

In addition, let us precise that both fitness function and accuracy score used are the simple bit string score.

Figure 6 represents the evolution of the sub-population size of each island (bit-flip, 1-flip, 3-flip and 5-flip), the average fitness (avg) of the overall population and the best individual fitness (best) according to the number of iterations (migrations). Due to the operators definition, the average fitness of the population is always increasing. One can remark that the best individual fitness is close to the average one, indicating that all individuals receive equivalent possibilities of improvement during the search.

To compare the experimental results with the theoretical values, we represent in Figure 7 the sub-population sizes of each island with respect to the average fitness of the population. The fact that these changes of sub-population sizes, *i.e.* the computational effort of each operator, match the theoretical domination rates. This shows the accuracy of the proposed model and its ability to simulate an operator selection mechanism.

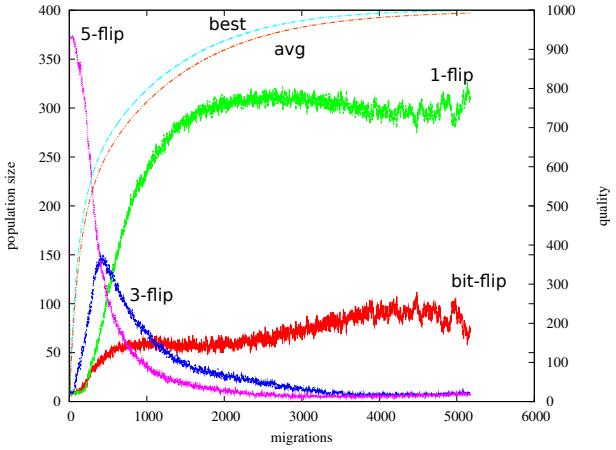


Figure 6: Attraction of each operator, average quality of the overall population and the quality of the best individual with respect to the migrations.

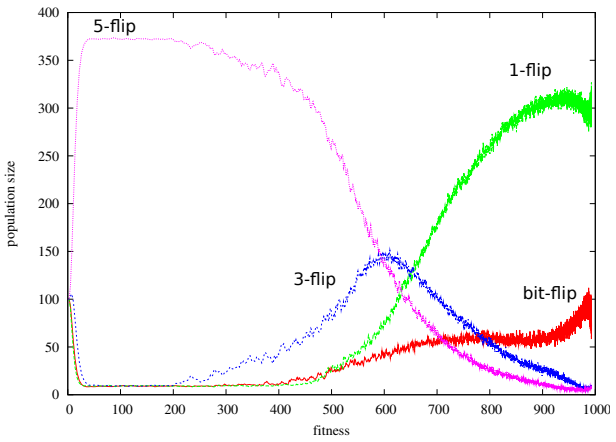


Figure 7: Evolution of the population size in each island with respect to the average fitness of the population.

4.2 DIM Parameters Setting

As presented in Section 2.2, this dynamic island model requires a few parameters: the number of individuals, the initial distribution of the individuals on the island and transition matrix values, as well as the update matrix parameters α and β.

4.2.1 Population Size

Our experiments show that the behavior of the DIM is almost identical with a small or a large population (more than 100 individuals by island). With more individuals, the mechanism is a bit more robust and less dependent to noise. With a small population, it is necessary to extend the number of steps, but the global number of operations is lower and updating the transition matrix is faster. A population size equal to $100 \times number\ of\ islands$ seems to be a good trade-off between running time and amount of migrations if we want to keep a nice matching to the theoretical operator efficiency. For practical uses, since this dynamic model is sufficiently inertial and reactive, smaller population provides equivalent results in terms of solution quality. This point will be assessed on more problems in future works.

4.2.2 Initial Transition Matrix

Except with random values, one may initialize the transition matrix using two natural strategies: firstly, with a *fully partitioned* island model (identity matrix); secondly, with a *fully connected* one (same value everywhere), which allows each individual to be processed by any combination of operators in the first steps. Both choices have been tested, with no impact on the results; after a few migration steps, both behaviors become identical thanks to the reactive characteristics of the model (on condition that $\beta \neq 0$). Figure 8 shows the evolution of the transition probabilities to stay on the same island with both initializations. These values coincide in less than 30 migrations in our experiments, which indicate that the initialization of the transition matrix does not change the global behavior of the DIM.

4.2.3 Matrix Update

Default used values for α and β are respectively 0.8 and 0.01. An increasing value of α slows down the search since the information obtained by recent migrations is less taken into account for the update. On the opposite side, decreasing value of α minimizes the impact of the knowledge (learning process) and overestimates the last migration effects; therefore, the search process may be badly driven by a migration that provides an exceptionally good result.

The influence of β is important, but its exact setting is not crucial for the smooth-running of the algorithm, even if too high values of β make the search slower. We must insure that $\beta \neq 0$, otherwise some islands may become unreachable (transition probability equal to 0). Effects of parameters α and β on the model are experimentally shown in figures 4.3 and 4.3.

4.3 Comparisons with Non-Dynamic Operators Selection

In Section 4.1 and Figure 7, we have shown the adaptive behavior of DIM for operator selection. However, we have to check that the learning process does not slow the overall search process compared to a classical EA. In order to evaluate the efficiency of DIM, we compare it with five EAs: four of them are based on the four operators presented in Section 3.1 (1-flip, 3-flip, 5-flip and bit-flip), and the last one uniformly selects at each iteration one of these four operators.

Figure 11 shows, for each operator, the (best current individual) fitness evolution depending on the number of total fitness computation (number of evaluations). This study enables a comparison of global efficiency between an adaptive search and a classical one.

Figure 11 shows that no EA with fixed operator is more efficient than others at each step of the search. As observed in Figure 7, the best operator selection at the begin of the search is 5-flip, then 3-flip and finally 1-flip. Uniform selection strategy obtains average performances and never outperforms all other EAs. DIM dominates every static local searches, and it is only met by 1-flip when the search is converging. These experiments show that DIM is able to detect very quickly the best operator(s) at a given state of the search, and that it is efficient even with no preliminary knowledge on these operators.

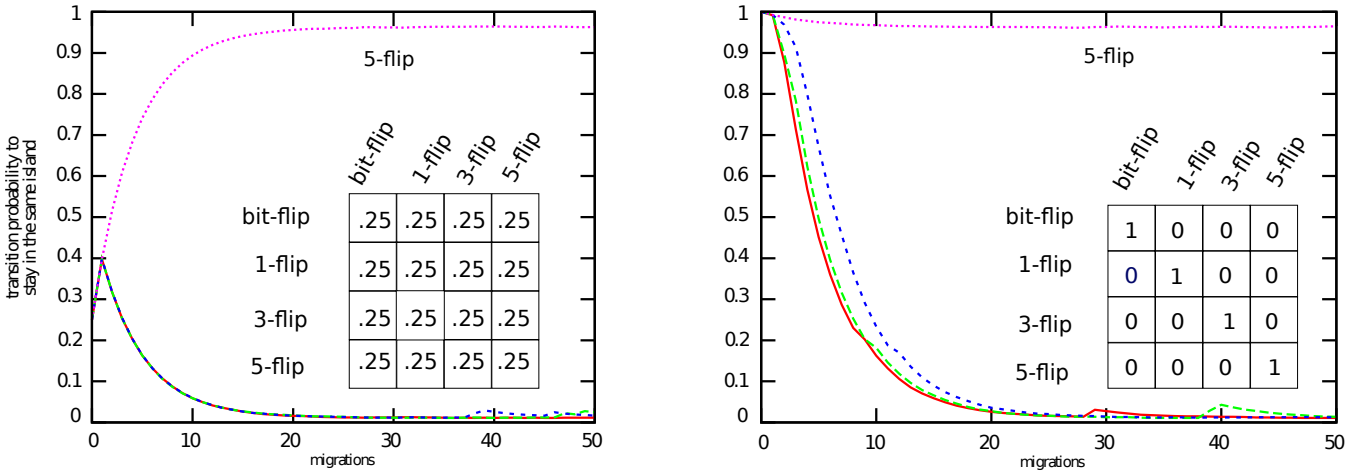
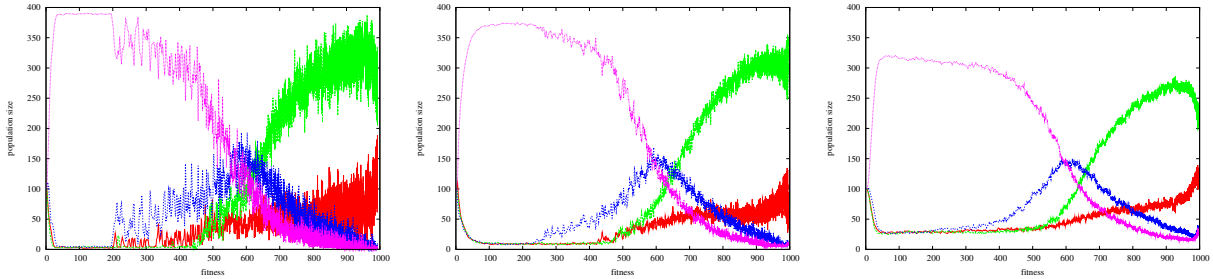
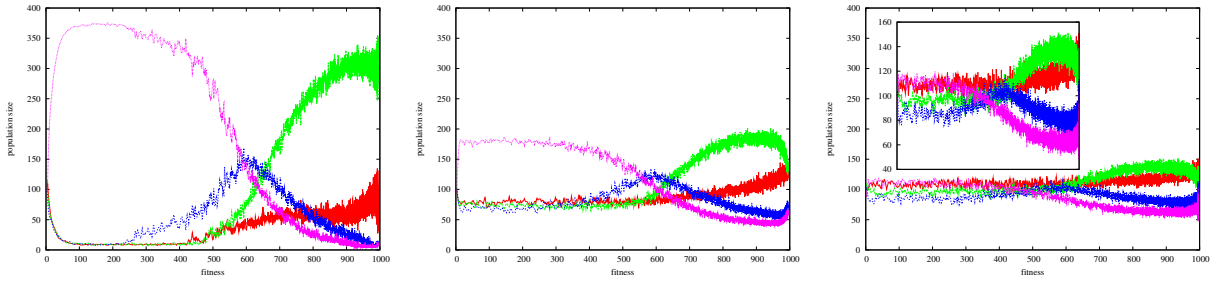


Figure 8: Evolution of the transition probability to stay in the same island with respect to the migrations. The left figure shows an initialization with the same value for all the transition probabilities and the right figure shows an initialization with an identity transition matrix.



(a) (α,β)=(0.4,0.01) (b) (α,β)=(0.8,0.01) (c) (α,β)=(0.95,0.01)

Figure 9: Changing the value of α: less or more inertness makes the model more stable but does not modify the global distribution of individuals



(a) (α,β)=(0.8,0.01) (b) (α,β)=(0.8,0.2) (c) (α,β)=(0.8,0.5)

Figure 10: Changing the value of β: more noise makes the distribution of individuals more uniform

5. CONCLUSION

This paper presents an original generic model for dynamic islands model and shows that this may constitute an efficient approach for the adaptive selection of operators in EAs. Each island is assigned to a single variation operator EA, and the dynamic regulation of migrations distributes the individuals on the most promising islands according to recent background information. At each stage of the search, the more efficient operators receive the greatest part of the computational resources, while the model is able to auto-adapt the attractive power of each islands. To assess the real efficiency of the model, we used an experimental protocol, comparing a theoretically optimal selection scheme of operators, for the One-Max problem, to empirical obtained values. The next step is to apply this operator selection strategy to more difficult problems and to compare it with other adaptive operator selection methods.

6. REFERENCES

[1] L. Araujo, J. J. M. Guervós, A. Mora, and C. Cotta. Genotypic differences and migration policies in an island model. In *GECCO*, pages 1331–1338, 2009.

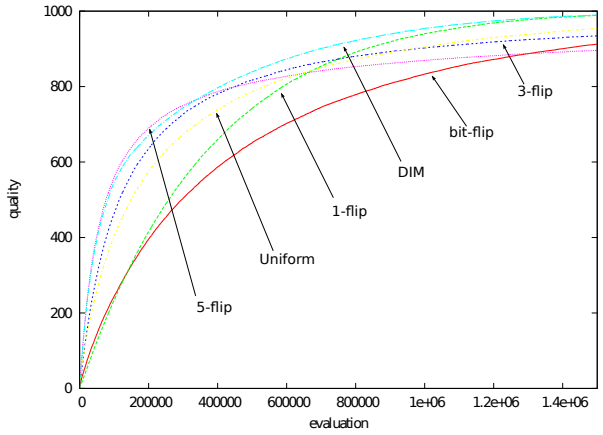


Figure 11: Quality of the best individual of the population with respect to the evaluations number for 5 algorithms: bit-flip, 1-flip, 3-flip, 5-flip, uniform selection and DIM selection.

[2] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7(4):311–334, 2001.

[3] A. Coelho, I. Ricarte, A. Yamakami, E. Noda, and A. Freitas. Devising adaptive migration policies for cooperative distributed genetic algorithms. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 438–443, 2002.

[4] K. De Jong. *Evolutionary computation: a unified approach*. MIT Press, 2006.

[5] J. Denzinger and J. Kidney. Improving migration by diversity. In *IEEE Congress on Evolutionary Computation (1)*, pages 700–707, 2003.

[6] B. Derbel and S. Verel. DAMS: Distributed Adaptive Metaheuristic Selection. In *Proceedings of the annual conference on Genetic and evolutionary computation (GECCO'11)*, pages 1–18, United Kingdom, July 2011.

[7] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2003.

[8] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. 2007.

[9] T. Eldos. A new migration model for the distributed genetic algorithms. In *Proceedings of the International Conference on Scientific Computing, Las Vegas, NV*, pages 26–29.

[10] Á. Fialho. *Adaptive Operator Selection for Optimization*. PhD thesis, Université Paris-Sud 11, Orsay, France, Dec. 2010.

[11] Á. Fialho, L. D. Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In *PPSN*, pages 175–184, 2008.

[12] S. Gustafson and E. K. Burke. The speciating island model: An alternative parallel evolutionary algorithm. *J. Parallel Distrib. Comput.*, 66(8):1025–1036, 2006.

[13] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, Sept. 2009.

[14] F. Lardeux and A. Goëffon. A dynamic island-based genetic algorithms framework. In *SEAL*, pages 156–165, 2010.

[15] F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, 2007.

[16] J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *J. Heuristics*, 16(6):881–909, 2010.

[17] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[18] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 975–980, 2007.

[19] M. Rucinski, D. Izzo, and F. Biscani. On the impact of the migration topology on the island model. *CoRR*, abs/1004.4541, 2010.

[20] Z. Skolicki and K. A. D. Jong. The influence of migration sizes and intervals on island models. In *GECCO*, pages 1295–1302, 2005.

[21] R. Tanese. Distributed genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 434–439, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[22] D. 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.

[23] R. K. Ursem. Diversity-guided evolutionary algorithms. In J. J. M. Guervos, P. Adamidis, H.-G. Beyer, J. L. F.-V. Martin, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Granada, Spain, September 7-11, 2002, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 462–474. Springer, 2002.

[24] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.

[25] D. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation*, 1(1):67–82, 1997.