Non Stationary Operator Selection with Island Models

Caner Candan
LERIA - Université d'Angers
Angers, France
caner.candan@univ-angers.fr

Adrien Goëffon
LERIA - Université d'Angers
Angers, France
adrien.goeffon@univ-angers.fr

Frédéric Lardeux
LERIA - Universityé d'Angers
Angers, France
frederic.lardeux@univ-angers.fr

Frédéric Saubion
LERIA - Université d'Angers
Angers, France
frederic.saubion@univ-angers.fr

ABSTRACT

The purpose of adaptive operator selection is to choose dynamically the most suitable variation operator of an evolutionary algorithm at each iteration of the search process. These variation operators are applied on individuals of a population which evolves, according to an evolutionary process, in order to find an optimal solution. Of course the efficiency of an operator may change during the search and therefore its application should be precisely controlled. In this paper, we use dynamic island models as operator selection mechanisms. A sub-population is associated to each operators and individuals are allowed to migrate from one sub-population to another one. In order to evaluate the performance of this adaptive selection mechanism, we propose an abstract operator representation using fitness improvement distributions that allow us to define non stationary operators with mutual interactions. Our purpose is to show that the adaptive selection is able to identify not only good operators but also suitable sequences of operators.

Categories and Subject Descriptors

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

Keywords

Island Models, Adaptive Operator Selection

1. INTRODUCTION

Basically, an evolutionary algorithm (EA) manages a population of individuals that encode possible configurations of the problem, in order to optimize a given fitness function. The individuals evolve by means of variation operators and selection processes. Although the efficiency of EAs is well-established on numerous optimization problems, their performance and robustness depend on the correct setting of its components by means of parameters. Parameter setting in EA has been extensively studied [LLM07]. According to usual taxonomies [EMSS07], we distinguish parameter tuning (off-line setting before solving) from parameter control (on-line setting during solving). Of course, different type of parameters can be considered: solving components (e.g., the selection process or the variation operators) or numerical parameters that modify the behavior of a given EA (e.g., the population size or the application rates of the variation operators). Most of the time, these parameters have strong interactions and it is very difficult to forecast how they will affect the performance of the EA. We focus here on the use of variation operators in population based algorithms: i.e., choosing at each iteration of the search process which variation operator should be applied.

Automatic tuning tools [NE07, HHKS09] have been developed to search for good parameters values in the parameters' space, which is defined as the crossproduct of the possible values of the parameters values and the possible instances of the problem. Specific heuristics are used in order to sample efficiently the possible values of parameters that are expected to have a significant impact on the performance. Another possible approach for parameter setting is to control the value of the parameters during the search. Adaptive operator selection (AOS) [Fia10] consists in providing an adaptive mechanism for selecting dynamically the suitable variation operator to apply on individuals at each iteration of the EA process. Recent approaches [FCSS08, MLS10] have proposed such adaptive mechanisms for general EAs with possibly many variation operators, whose behavior may be unknown. However, as suggested by the *No Free Lunch* theorems [WM97], it is difficult to anticipate the performance of an algorithm on any instances of a wide class of problems without preliminary experiment or learning process. Note that control could be consider as a dynamic tuning process: once the search has been achieved, the parameters values obtained by the control process can be synthesized and used for future runs. Of course, this dynamic approach allows the user to retrieve information on the temporal changes of the parameters' values.

In [CGLS12], we propose to use island models as an AOS mechanism. Island models [WRH98] provide a natural abstraction for dividing the population into several subsets, distributed on islands. An independent EA is run on each of these islands. Individuals are allowed to migrate from one island to another one in order to insure information sharing

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

during the resolution and maintain some diversity on each island thanks to incoming individuals. In order to use island models as an AOS method, the main principle consists in distributing each operator on a different island. A dynamic migration policy between the islands is used in order to move at each step the individuals to the most promising operators, i.e. supposed to have the best current utility[1].

Given an EA, it can be really difficult to assess the utility of the operators on real problems since it may depend on the individual but also on the previously applied operators. Problem-specific operators are often required (e.g., k-opt exchanges for the travelling salesman problems), which makes difficult comparisons between different problems. Suitable and reliable criteria must be defined as well as methods for recording and using their values. Therefore, artificial scenarios have been proposed in order to focus on the management of the operators. These scenarios may serve as a methodological basis for assessing the benefit of AOS procedures [Thi05, CFSS08]. The main idea is to use reward distributions to simulate the utility of operators — a reward corresponds to the current effect of the operator on the population. As mentioned above, the utility of the operators is likely to change along the solving process and thus these distributions are permuted regularly among the operators. In this paper, we also focus on such non stationary operators. We propose to induce more interaction between the operator by using distributions that change according to the history of the search, i.e. the operators that have been applied before. In particular, when managing the balance between exploration and exploitation in an EA, it is clear that intensification operators can only be used until they reach a minimal optimum, while similarly diversification operators cannot diversify the search forever. Roughly speaking, an intensification (resp. diversification) operator may clearly become less and less efficient as long as it is used. Similar situations certainly occur in presence of several operators who share the same search purpose and who have thus strong interactions. Our purpose is to assess the AOS mechanism based on island models is able to identify precisely operators whose efficiency may vary continuously and also sequences of such suitable operators.

2. ISLAND MODELS

Based on the main principles of EAs, island models consider a set of sub-populations, clustered on islands, which are evolving independently during some search steps and interacting periodically. This model provides an improved management of diversity and simplify the parallel implementation of EAs. Island models have mostly been used in a static way, with individuals that migrate from islands to islands according to a fixed predefined policy. Nevertheless, dynamic regulation policies allow the model to increase or decrease the migration probabilities during the evolutionary process according to the impact of previous migrations between islands.

In the remaining of this paper, we are interested in solving general optimization problems. An optimization problem is a pair $(f, \mathcal{S})$ where $\mathcal{S}$ is a search space whose elements represent potential solutions of the problem and $f : \mathcal{S} \to \mathbb{R}$ is a fitness function that associates a value. An optimal solution (maximization problem) is an element $s^* \in \mathcal{S}$ such that $\forall s \in \mathcal{S}, s \neq s^*, f(s^*) \geq f(s)$.

[1]The utility of an operator corresponds to its expected effect for the next steps of the solving process. Other terms can be found in the literature, e.g. efficiency or impact.

2.1 Basic Notions

An island model can be defined by the following elements:

- Dimensions: a size n, a set of islands $\mathcal{I} = \{i_1, \ldots, i_n\}$, a set of algorithms $\mathcal{A} = \{A_1, \ldots, A_n\}$, a set of populations $\mathcal{P} = \{P_1, \ldots, P_n\}$, each P_i is a subset of $\mathcal{S}$. Each population P_k is assigned to island i_k. Each algorithm A_k is assigned to island i_k.
- A topology: an undirected graph $(\mathcal{I}, V)$ where $V \subseteq \mathcal{I}^2$ is a set of edges between islands of $\mathcal{I}$.
- A migration[2] policy: a squared matrix M of size n, such that $M(i, j) \in [0, 1]$ represents the probability for an individual to migrate from island i to island j. This matrix is supposed to be coherent with the topology, i.e., $\forall (i, j) \in \mathcal{I}^2 \setminus V, M(i, j) = 0$.

Different choices can be made for the topology: complete graph, ring... Note that a node (i, i) in the graph will be used if one wants to allow individuals to have the possibility to stay on the same island. A basic island model algorithm can be defined as algorithm 1, where $A_i(P_i)$ is the population obtained after applying a basic iteration of algorithm A_i on population P_i. *best* is a function that computes the best individual of a set with regards to the objective function f. The stop condition corresponds to a limited number of iterations or the fact that an optimal solution has been found in the total population $\bigcup_{k \in 1..n} P_k$ (if the optimal bound is known). Note that the size of this global population is fixed along the process but the size of each P_k changes constantly according to the migrations.

input : a migration matrix M,
a set of islands $\mathcal{I} = \{i_1, \ldots, i_n\}$,
a set of algorithms $\mathcal{A} = \{A_1, \ldots, A_n\}$,
a set of populations $\mathcal{P} = \{P_1, \ldots, P_n\}$,
an optimization problem $(\mathcal{S}, f)$
output: a solution s^*

```
while not stop condition do
    for i ← 1 to n do
        P_i ← A_i(P_i);
        for s ∈ P_i do
            generate a random number rand;
            if ∃(i, j) ∈ V and rand < M(i, j) then
                P_j ← P_j ∪ {s};
                P_i ← P_i \ {s};
    s* ← best(∪_i(P_i));
```

Algorithm 1: Basic Island Model

2.2 Dynamic Migration Policy

Dynamic island models use migration policy that evolves during the search. Each island is equipped with two main processes that define how its population evolves and how the migration policy is controlled.

[2]Note that alternative island models use migration policies where migration probabilities depend on the characteristics of the individuals [SJ05].

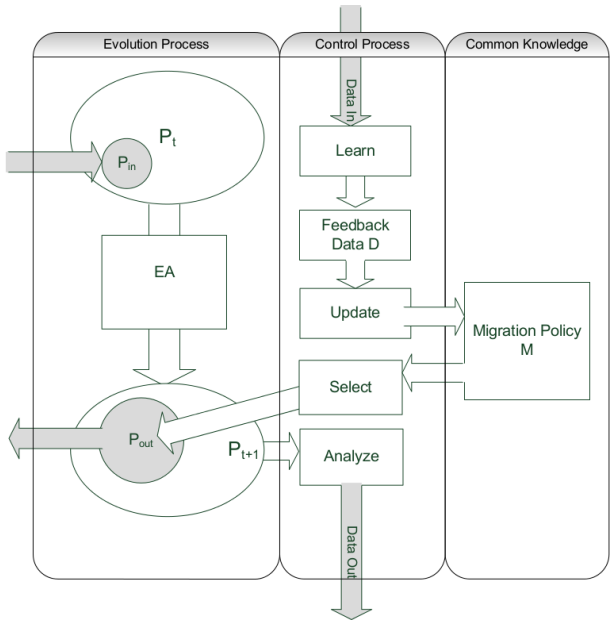

Figure 1: General Mechanism for an Island in a Dynamic Model

Figure 1 highlights that the migration policy is modified during the run according to incoming feedback received from other islands. This feedback allows the island to know how its individuals have been improved on other islands. The basic learning principle consists in sending more individuals to island that have been able to improve them significantly and less to islands that are currently less efficient for these individuals. To avoid brutal changes, these effects may be evaluated on a time sliding window. The *selection* component uses the migration matrix M to select the individuals that are sent to other islands. Of course, some individuals may stay on the same island. The *analyze* component aims to evaluate the improvements of the individuals after the EA has been applied. It sends this feedback information to the island these individuals were originated from. According to the previous notations, we define the following basic processes and data structures:

- A data matrix D is a square matrix of size n. $D(i,j)$ evaluates the improvement obtained by individuals of island i when they are processed on island j.
- An algorithm $A \in \mathcal{A}$ is a process that computes a new population P_{t+1} from a given population P_t by applying an evolution process.
- $learn(D, D_{in})$ is a process that modifies D according to information received in D_{in}. In fact this process can be seen as an operation $D \leftarrow D \oplus D_{in}$, which should be defined according to the problem.
- $update(M, D)$ is a process that modifies the migration matrix M according to the data matrix D.
- $select(P, M, j)$ is a process that selects a sets of individuals from the current population of the island P to be sent to another island j according to migration matrix M.
- $analyze(P)$ is a process that evaluates the improvement obtained in island i by individuals from island k during the last iteration of the algorithm ($D_{out}(k,i)$).

3. ADAPTIVE OPERATOR SELECTION

3.1 Operator Selection Mechanisms

Let us consider n operators $\{o_1, \ldots, o_n\}$ and associate an estimated utility $u_{i,t}$ of operator o_i at time t. The probability of selecting operator o_i at time t is $s_{i,t}$. In a static setting, one could of course use a predefined fixed roulette wheel selection $(s_1, \ldots, s_n)$, i.e. $s_{i,t}$ is constant during the execution of the algorithm. Note that these values can be determined by a tuning process. It is easy to check that a static roulette selection can be simulated in the island model. Given $r = (s_1, \ldots, s_n)$, we define a static transition matrix M whose n rows are all equal to r. Clearly $rM = r$ (r is a steady state vector, achieving just the expected application rates of the operators).

Contrary to a static tuning of the operator application rates, adaptive operator selection consists in selecting the next operator to apply at time $t+1$ by adapting the selection probability during the search. The utility has to be reevaluated at each time, classically using a formula $u_{i,t+1} = (1-\alpha)u_{i,t} + \alpha r_t$ where r_t is the reward associated to the application of operator i (immediate utility) and α is a coefficient that controls the balance between past and immediate utility, as in classic reinforcement learning techniques [SB98]. A classic mechanism is the probability matching selection rule:

$s_{i,t+1} = p_{min} + (1 - n.p_{min}) \frac{u_{i,t+1}}{\Sigma_{k=1}^{n} u_{k,t+1}}$

where a non negative p_{min} insures a non zero selection probability for all operators.

This selection rules are known to have drawbacks[3] and an alternative selection rule has been proposed in [Thi05], called adaptive pursuit (AP), that distinguishes the best current operator from the others:

$$\begin{cases} s_{i^*,t+1} = s_{i^*,t} + \beta(p_{max} - s_{i^*,t}) \\ s_{i,t+1} = s_{i,t} + \beta(p_{min} - s_{i,t}) \end{cases}$$

where $i^* = \text{argmax}\{u_{i,t}, i = 1..n\}$, $p_{max} = 1 - (n-1)p_{min}$ and β is a parameter to adjust balance of this winner-take-all strategy.

3.2 Island Models as Adaptive Operator Selection Mechanisms

The generic model can easily be instantiated in order to manage classical evolutionary algorithms like genetic, memetic or population-based local search algorithms. We describe the specialization of this scheme for the management of operators. Each island i is equipped with a particular operator o_i and has the following specifications:

- The learning process is very simple here since we just keep the last performance (time window of size 1), i.e. $[D \oplus D_{in}](i,j) = D(i,j)$.
- The algorithm A_i applies the operator o_i assigned to this island on every individuals of the population P_i: $A_i(P_i) = \{o_i(s) | s \in P_i\}$.

[3]The proportional selection gives more importance to operators that have a good average behavior compared to operator that can be very efficient but less frequently.

- The *analyze*(P) process computes the feedback information and sends it to each island (including its own). We may consider for instance the following functions:

  Mean $D_{out}(k,i) = \frac{\Sigma_{s \in P_i[k]} f(s)}{|P_i[k]|}$

  Max $D_{out}(k,i) = \max_{s \in P_i[k]}(f(s))$

  where $P_i[k] = \{s \in P_i | s \text{ comes from island } k\}$

- The *update* process uses the data matrix D (i.e., the feedback from other island) in order to modify the migration matrix. Of course, only the line corresponding to island i is modified. We compute an intermediate reward vector R. We propose to use an intensification strategy: only the island where individuals of i have obtained the best improvement is rewarded (note that there could be several such best islands).

$$R(k) = \begin{cases} \frac{1}{|B|} & \text{if } k \in B, \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{with } B = \underset{k}{\operatorname{argmax}}\, D(i,k)$$

  then the migration matrix is updated as:

  $M(i,k) = (1-\beta)(\alpha.M(i,k) + (1-\alpha)R(k)) + \beta N(k)$

  where N is a stochastic vector such that $||N|| = 1$. The parameter 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. The influence of these parameters has been studied in [CGLS12].

3.3 AOS as a Multi-Armed Bandits Problem

The initial multi-armed bandit problem was introduced in the context of the design of experiments by [Rob52]. It was formulated as the maximization of the total gain of a gambler who could make n tosses with two coins A and B with a gain of 1 for each head but nothing for tails. The biases of the coins are unknown. This problem is known as the *Two-armed Bandit* and has been extended to multi-armed bandit by Rodman [Rod78].

Let us consider K arms and an infinite sequence $\vec{x_1}, \vec{x_2}, \ldots$ of vectors $\vec{x}(t) = (x_1(t), \ldots, x_K(t))$, where $x_i(t) \in [0,1]$ is the reward obtained if the arm i is chosen at step t. At step $t+1$, the gambler only knows the rewards $x_{i_1}, \ldots, x_{i_t}$ obtained from the sequence $i_1, \ldots, i_t$, where $i_j \in \{1, \ldots, K\}$. A strategy is thus a sequence $I_1, \ldots, I_T$ with $I_j : (\{1, \ldots, K\} \times [0,1])^{j-1} \rightarrow \{1, \ldots, K\}$ whose cumulated reward is $R_S(T) = \Sigma_{t=1}^{T} x_{i_t}(t)$.

Optimal strategies have been initially proposed [Fel62, Git79] for this problem. Later, Auer [Aue02] has proposed to use this problem to manage the compromise between exploration and exploitation in optimization algorithms. The operator selection problem can be expressed as a multi-armed bandit problem [CFSS08]. An operator is associated to each arm. The rewards are the fitness improvements that are successively obtained. Upper bound confidence functions can be used to solve the associated bandit problem. The UCB1 [ACBF02] is defined as:

$$\bar{r}_{j,t} + \sqrt{\frac{2 \log \Sigma_k n_{k,t}}{n_{j,t}}}$$

where $\bar{r}_{j,t}$ is the average reward for arm j at iteration t and $n_{j,t}$ denote the number of times the j^{th} arm has been played. The arm that maximizes the previous expression is chosen for being played at iteration $t+1$. Here, the reward distribution of the operator are likely to change according to the state of the search. Therefore dynamic multi-armed bandit has to be considered and UCB have to be revised [CFSS08]. A standard test – known as Page Hinkley – for the change hypothesis is used. The resulting DMAB selection algorithm is thus based on UCB1 with a dynamic restart mechanism that is driven by a Page Hinkley test. Of course one has to consider carefully performance measures in order to assess the utility of the operators [MFS$^+$09]. Note that some hyper-heuristic approaches also use learning techniques (e.g., Markov chains in [MK11]) to learn good transitions between heuristics and attempt to model the best sequence of heuristics to apply.

4. USING NON STATIONARY OPERATORS

Most of the operators used in EAs have a behavior that depends on the individuals on which they are applied or, more generally, on the current state of the search. Therefore, it is likely that their impact on the exploration/exploitation balance will change over time. Moreover, the impact of an operator may also depend on the operators that have been previously applied on the individuals. In this case, the operator selection mechanism should be able to identify suitable sequences of operators instead of focusing only on the next operator to apply or on proportion of applications of each operators.

In the following abstract problem definition, operators will be represented by simple fitness improvements functions whose values may depend on the history of the individual. The history of an individual is the sequence of islands this individual has been on during the last t iterations of the algorithm.

Our purpose is to propose an abstract view of EAs in order to focus more on the operators than on the problem. Therefore, we will consider abstract operators that are independent from the genotypic description of the elements of the search space in order to be able to compute optimal or sub-optimal policies and to validate the properties of our AOS model against these policies.

A basic steady state process can be simulated by a tuple $(P_0, \mathcal{O}, \Omega, \sigma, \iota)$, including:

- an initial population $P_0 \subseteq 2^{\mathcal{S}}$.
- a set of variation operators $\mathcal{O}$ whose elements are relations $o \subseteq \mathcal{S} \times \mathcal{S}$. In practice, most operators cannot be expressed as functions $o : \mathcal{S} \rightarrow \mathcal{S}$ due to their non deterministic and stochastic nature. Therefore let us note $s \rightarrow_o s'$ the fact that $(s, s') \in o$.
- an operator selection function $\Omega : \mathbb{N} \rightarrow \mathcal{O}$ that select operator to apply at each iteration of the algorithm.
- a selection function $\sigma : 2^{\mathcal{S}} \rightarrow \mathcal{S}$ and an insertion function $\iota : 2^{\mathcal{S}} \times \mathcal{S} \rightarrow 2^{\mathcal{S}}$.

An execution of $A \equiv (P_0, \mathcal{O}, \Omega, \sigma, \iota)$ is thus a sequence $P_0, \ldots, P_T$, such that[4]

- $\forall 1 \leq i \leq T, P_i \subseteq 2^{\mathcal{S}}$,
- $\forall 0 \leq i \leq T-1, P_{i+1} = \iota(P_i, s)$ with $\sigma(P_i) \rightarrow_o s$ and $o = \Omega(i)$,
- T is the maximal number of allowed iteration.

We consider here fixed size populations, i.e., $\forall 1 \leq i \leq T, |P_i| = |P_0| = N$. Our purpose is to highlight on the operators during the search. Given the execution of an algorithm $P_0, \ldots, P_n$, we define a sequence $P_0^{\mathcal{O}}, \ldots, P_n^{\mathcal{O}}$, such that:

- $\forall 1 \leq i \leq T, P_i^{\mathcal{O}} \subseteq \mathcal{S} \times \mathcal{O}^*$, where $\mathcal{O}^*$ denotes the set of words constructed using operators in $\mathcal{O}$. An element of $P_i^{\mathcal{O}}$ is just a pair $(s, o_1 \ldots o_i)$ where s is an element of P_0 and $o_1 \ldots o_i$ is a sequence of operators that have been applied on it during the execution of the algorithm.
- $P_0^{\mathcal{O}} = \{\{(s, \epsilon) | s \in P_0\}\}$ is a multiset built with element of P_0 such that no operator has been applied on it yet.
- $\forall 1 \leq i \leq T, \Pi(P_i^{\mathcal{O}}) \supseteq P_i$, where $\Pi(P_i^{\mathcal{O}}) \equiv \{s' \in \mathcal{S} | (s, o_1 \ldots o_i) \in P_i^{\mathcal{O}}, s \rightarrow_{o_1} .. \rightarrow_{o_i} s'\}$

Due to the non-determinism of the operators, given a sequence of populations obtained by an algorithm $P_0, \ldots, P_n$ we cannot associate a unique sequence $P_0^{\mathcal{O}}, \ldots, P_n^{\mathcal{O}}$ as defined above (in fact $\forall 1 \leq i \leq T, \Pi(P_i^{\mathcal{O}}) \supseteq P_i$). In order to provide a general model for studying operators behavior, we abstract operators and individual according to the notion of fitness.

Given the execution of an algorithm $P_0, \ldots, P_T$, we associate a sequence $P_0^f, \ldots, P_T^f$, with $P_i^f \in \mathbb{R}^N$ and such that $\forall 1 \leq i \leq T, \forall s \in P_i, \exists v \in P_i^f, v = f(s)$. Therefore, we may now better link operators applications and successive populations. We define thus a relation $P_i \equiv_f P_i^{\mathcal{O}}$ if and only if there exists P_i^f, $\forall s \in P_i, \exists v \in P_i^f, v = f(s)$ and $\forall s \in \Pi(P_i^{\mathcal{O}}), \exists v \in P_i^f, v = f(s)$. At this stage, we do not need to consider the selection and insertion functions.

Operators can be abstracted by fitness improvements, given by a probabilistic distribution of their value. For maximization problem $(f, \mathcal{S})$, the effect of operators can be defined according to the evaluation function f. Given an element $s \in \mathcal{S}$, the variation of fitness f induced by an operator o is given as $\delta_f^o(s) = \frac{f(o(s)) - f(s)}{f(s)}$.

Abstracting operator by distributions has been investigated in [Thi05] where uniform distribution are associated to each operator in different overlapping intervals of values. Other distributions have been proposed in [CFSS08] with the Boolean and the outliers models. In these models, each operator o is associated a distribution (p_o, δ_o), i.e., $\forall s \in S, P(\delta_f^o(s) = \delta_o) = p_0 \wedge P(\delta_f^o(s) = 0) = (1 - p_0)$. The value of p_0 and δ_o are adjusted to insure some properties concerning the ordering of the operators according to expected rewards.

In these works, the distributions are fixed during a given number of iteration — epoch — and then are reassigned to operators according to a permutation. The utility of the operator is thus non stationary during the execution of the algorithm and the AOS mechanism has to discover the best operator to apply during each epoch.

Similarly, our purpose is to have operators whose behavior changes during the solving process in order to simulate real situation. We want to consider more continuous changes in the distributions. We consider possible interactions between the operators, i.e. the utility of an operator depends on the previously used operators on the same individual. Operators corresponds to fitness improvements functions whose values depend on the history of the initial individuals (i.e., the way it has been processed during solving). The idea is to provide a model where the utility of an operator decreases proportionally to its use. In such model, the AOS mechanism must not detect the best operator during an epoch but rather identify suitable sequences of operators.

Given an operator o with a distribution (p_o, δ_o), we extend our new operator to elements of $P_i^{\mathcal{O}}$ as a distribution $(p_o, \delta_o(1 - Occ(\omega, o)))$ with $Occ(\omega, o) = \frac{|\{i \in 1, \ldots, |\omega|, \omega_i = o\}|}{|\omega|}$ (i.e., proportion of operator o in ω). In practice, one has to choose the distribution but also a sliding window for recording not ω but only a subsequence of the operators that have been previously applied.

5. EXPERIMENTAL RESULTS

This section focuses on the capacity of the dynamic island model to detect a suitable scheduling of non stationary operators which maximizes the objective function. Two experiments have been conducted for observing the behavior of the model: the first one with equivalent operators and the second one with different operators.

5.1 Behavior with equivalent operators

We consider 3 equivalent operators associated to 3 islands and a population of 300 individuals uniformly distributed on each island. The parameters α and β of the *update* function (3.2) are respectively set to the default values 0.8 and 0.01. Each individual corresponds to $(s, |\omega|)$ (remind that we focus only on the fitness values, see Section 4) with a sliding window of length $|\omega| = 10$ in order to record the last visited islands. The fitness value of initial individuals is 0. Operators have similar distribution parameters $(p_o, \delta_o) = (1, 0.5)$. In this context, an expected application of the operators for an individual is to distribute them equitably in the window.

Representing simultaneously the evolution of all migration probabilities leads to unreadable figure. Nevertheless we systematically observe a fast stabilization of the model with a migration policy which simulate a ring topology, i.e. which follows a Hamiltonian circuit. Thus, in figure 2, we choose to only represent the sum of probabilities (in percent) corresponding to the two possible circuits (0,1,2) and (0,2,1).

In this figure, which represents a given run, one can observe a fast stabilization of the model with migrations corresponding to a circuit (0,1,2). Due to noise (parameter β), perturbations may appear at any time (for instance here between migrations 6000 and 8000) before stabilizing on a (possibly different) circuit.

Similar experiments were made with more than 3 islands and lead to same observations. For instance, figure 3 provides the complete graph representation of the migration probabilities between 10 islands, each having the same char-

[4]We must mention here that the P_i are indeed multisets of elements of $\mathcal{S}$. Nevertheless, in the following we will use the classic set notations in order to simplify the reading.

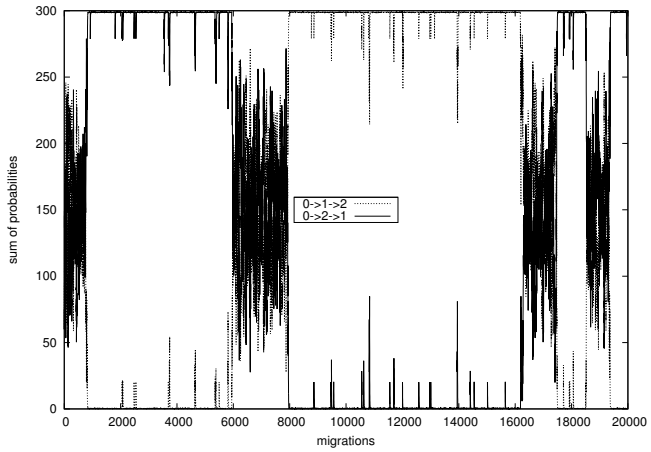

Figure 2: Stabilization periods on ring topology.

acteristics than in the previous experiment. As we have already mentioned, it is not possible to represent the evolution of the probabilities of migration but we developed a graphic application in order to observe the behavior of the model during the search. Figure 3 shows that a 10 islands circuit is found. More precisely, this figure corresponds to a state of the search in term of migration probabilities (represented by arrows). The animation shows an alternation between short perturbations periods and large stabilisation periods (with different circuits).

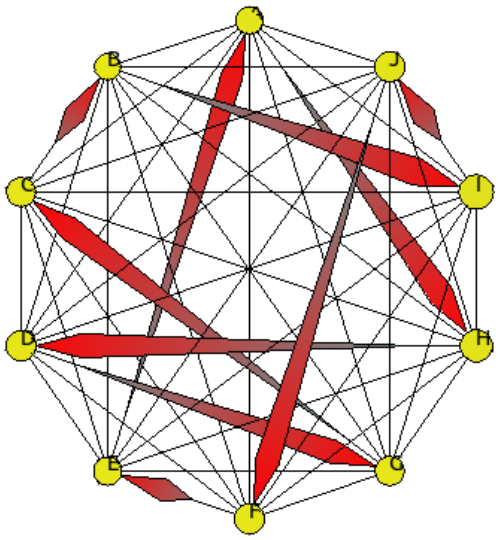

Figure 3: Circuit for 10 islands.

All experiments with equivalent operators show the ability of the dynamic island model to adapt its search strategy in order to make the search more efficient.

5.2 Comparison with other methods

Experiments with equivalent operators highlight the impact on migrations of non stationary operators. Now, we extend these experiments to a more general case with different operators. We first compare our model to simple roulette wheel selection mechanisms.

5.2.1 *Island Models vs.Roulette Wheel Selection*

Let us use 6 different sets of operators for our experiments with the next distribution parameters (with sliding window size equal to 10):

- set1: {(0.3,0.2), (0.8,0.2), (0.5,0.5), (0.1,0.9)}
- set2: {(0.7,0.7), (0.3,0.1), (0.8,0.1), (0.1,0.4), (0.8,0.9)}
- set3: {(0.9,0.1), (0.3,0.3), (0.2,0.7), (0.1,0.1), (0.6,0.3), (0.8,0.7)}
- set4: {(0.1,0.6), (0.5,0.7), (0.3,0.8), (0.4,0.6), (0.9,0.7), (0.2,0.3), (0.6,0.6)}
- set5: {(0.4,0.2), (0.6,0.1), (0.5,0.8), (0.7,0.3), (0.8,0.8), (0.1,0.1), (0.2,0.3), (0.9,0.6)}
- set6: {(0.1,0.9), (0.2,0.8), (0.3,0.7), (0.4,0.6), (0.5,0.5), (0.6,0.4), (0.7,0.3), (0.8,0.2), (0.9,0.1)}

The dynamic island model parameters are similar to those of the previous section. To assess the ability of the dynamic island model to well perform with different operators, we will compare its efficiency with other selection mechanisms: two different roulette and adaptive pursuit methods (see section 3.1).

- The *uniform roulette* is obtained using the island model defined in section 2 with $\alpha = 1$, $\beta = 0$ and an initial migration matrix M such that $\forall (i,j) \in \mathcal{I}^2, M(i,j) = 1/9$.
- The *tuned roulette* only differ on the initial migration matrix, which is initialized with probabilities given by an off-line tuning process. In our case, after a dynamic island model process, the average operators application rates are used to initialize the static migration matrix lines.

Table 1 shows results obtained by the 3 algorithms in term of average, standard deviation and best final value for 100 individuals and 2000 migrations.

We indicate in bold methods that are not statistically outperformed by any other one (w.r.t. Mann-Whitney test). One can observe that both tuned roulette and dynamic island model obtain the best results. However, remind that the tuned roulette requires first a dynamic island model run to set its value.

5.2.2 *Island Models vs.Other AOS*

Comparisons must also be performed with the AOS methods described in section 3.1. We consider here several possible scenarios in order to highlight the properties of the different methods. The scenarios are again defined by their corresponding sets of abstract operators. The length of the windows to compute the variation of fitness is still 10. We consider:

- A scenario with identical operators $s_1 \equiv \{(1,0.5),(1,0.5),(1,0.5)\}$
- Two scenarios with three different operators. $s_2 \equiv \{(0.9,1),(0.5,4),(0.2,9)\}$ where operators have close expected values and $s_3 \equiv \{(0.9,10),(0.5,10),(0.2,1)\}$ where these expected values are more different.

Set	Nb op.	Uniform Roulette			Tuned Roulette			Dynamic Island Model		
		avg	sd	best	avg	sd	best	avg	sd	best
set1	4	210.4	7.0	227.3	140.1	5.7	155.1	**235.7**	6.1	245.1
set2	5	134.6	10.0	164.2	598.2	10.2	628.8	**624.3**	9.8	658.4
set3	6	296.5	12.0	323.0	344.5	8.2	367.7	**369.6**	8.1	378.3
set4	7	475.4	12.1	512.8	**592.0**	10.7	625.1	**610.0**	10.7	636.4
set5	8	438.2	11.4	479.4	**680.6**	10.5	710.3	**711.1**	9.5	733.2
set6	9	326.1	9.5	355.0	**359.9**	10.0	391.4	**361.8**	9.8	394.4

Table 1: DIM vs. RW on different sets of operators.

- Two scenarios with a larger number of operators and roughly two types of operators: good ones and bad ones. $s_4 \equiv \{(0.9, 5), (0.2, 10), (0.1, 1), (0.1, 1), (0.1, 1)\}$ and $s_5 \equiv \{(0.9, 5), (0.4, 7), (0.3, 8), (0.2, 10), (0.2, 1), (0.2, 1), (0.1, 1), (0.1, 1), (0.1, 1), (0.1, 1)\}$.

The following AOS are used:

- OR is a *myopic* oracle that knows the values of the operators and the last applied operators in order to select the operator with the best expected value according to the variation of fitness. Note that this oracle does not compute the optimal solution for a given number of iterations since it does not take into account future operator applications.
- U is a uniform selection rule.
- ARW is a probability matching selection rule (adaptive roulette wheel). The minimal probability p_{min} is set to 0.01.
- DIM is our dynamic island model. The values of the parameters are $\alpha = 0.8$ and $\beta = 0.1$ except for s_2 where a very small number of individual requires a higher noise rate $\beta = 0.35$ to get acceptable results.
- AP is the adaptive pursuit method. The values of the parameters are $p_{min} = 0.1$ and $\beta = 0.8$.
- UCB uses the upper confidence bound UCB1
- $DMAB$ is the dynamic selection based on UCB1 with the Page Hinkley test. This method requires several parameters for the test. Note that, while the parameters values are quite intuitive for scenarios based on epoch as in [CFSS08], they are more difficult to set here and sometimes, $DMAB$ behaves worse or is equivalent to UCB. A deeper study of the parameter tuning should be conducted.

All these methods have been implemented in the same Scilab framework[5], each method is run 20 times with 1000 iterations each (i.e., 1000 applications of operators). For DIM the total number of individuals is denoted as *nbind*. The results correspond to the average value (rounded for sake of readability) and the (rounded) standard deviation obtained:

- by the best individuals of the *nbind* individuals of each run for DIM
- by the *nbind* best scores obtained over (20.*nbind*) runs for the other methods.

Note that our criteria do not favour DIM.

On table 2 we may remark that if U get good results as on s_1 then almost all methods are equivalent. AP provides good results for s_1 where indeed all operators are the same. Scenario s_2 shows that again, since the operators have close expected values, U obtains good results. Here, DIM requires a higher noise rate to balance the very few number of individuals. Keeping the same number of operators but inserting an operator that is clearly less efficient than the other ones is more difficult to detect for ARW. Of course U does not work well anymore. Note that increasing the number of individuals improves the results of DIM while increasing the number of runs has not necessarily an huge impact for all other methods, while the result computation favours these methods. This is especially clear for s_5. UCB and $DMAB$ are really efficient until the scenarios become larger. Even if it requires a sufficient number of individuals, DIM seems reliable and is able to improve its results if more computing resources are available. Note that more work should be done concerning the parameters of the methods. Moreover, additional criteria could be taken into account for assessing the utility of operators – e.g., entropy of the sub-populations or execution – in order to better manage the balance between intensification and diversification.

6. CONCLUSION

In this paper, the utility of non stationary operators is simulated by variable fitness improvement distributions. Contrary to previous artificial scenarios, the utility decreases proportionally to the previous applications of the operators and corresponds to situations where a single best operator cannot be used forever or for a sufficiently long number of iterations — e.g., always enforcing intensification. Adaptive operator selection aims thus at detecting suitable sequences of operators rather than identifying the best operator for a given epoch. In this context dynamic island models provide good results and can be used as an interesting alternative operator selection mechanism.

Acknowledgements : We are indebted to the anonymous referees for their valuable comments and suggestions. We also want to thank our students M. Dalle and Y. Hay for having provided the graphic interface for Figure 3. This work is partially supported by the Pays de la Loire Region (France) within the LigeRO project.

[5]http://www.scilab.org/

Set	Nb op.	Nb ind	OR		DIM		DMAB		ARW		U		AP		UCB	
			avg	sd	avg	sd	avg	sd	avg	sd	avg	sd	avg	sd	avg	sd
s_1	3	300	151	0	152	2	126	0	151	0	152	0	165	0	126	0
s_2	3	9	511	7	472	15	491	10	478	10	505	1	490	11	498	11
s_3	3	18	1477	12	1536	19	1533	25	1458	18	1482	19	1588	19	1588	30
s_3	3	99	1501	9	1604	13	1603	38	1489	11	1503	16	1629	13	1658	21
s_4	5	50	712	9	712	15	732	17	688	11	682	11	693	12	742	20
s_5	10	100	1301	15	1004	16	1082	29	1067	30	966	15	1030	13	1098	19
s_5	10	1000	1339	10	1226	15	1128	20	1141	16	998	9	1070	17	1138	19

Table 2: Results obtained with different AOS approaches on different sets of operators.

7. REFERENCES

[ACBF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[Aue02] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[CFSS08] L. Da Costa, Á. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 913–920. ACM, 2008.

[CGLS12] C. Candan, A. Goëffon, F. Lardeux, and F. Saubion. A dynamic island model for adaptive operator selection. In *Genetic and Evolutionary Computation Conference (GECCO'12)*, pages 1253–1260, 2012.

[EMSS07] A. E. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. 2007.

[FCSS08] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In *Parallel Problem Solving from Nature - PPSN X, 10th International Conference, Proceedings*, volume 5199 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2008.

[Fel62] D. Feldman. Contributions to the "two-armed bandit" problem. *The Annals of Mathematical Statistics*, 33(3):847–856, 1962.

[Fia10] Á. Fialho. *Adaptive Operator Selection for Optimization*. PhD thesis, Université Paris-Sud 11, Orsay, France, December 2010.

[Git79] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

[HHKS09] F. Hutter, H. Hoos, K.Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, September 2009.

[LLM07] F. Lobo, C. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, 2007.

[MFS+09] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, and M. Sebag. Compass and dynamic multi-armed bandits for adaptive operator selection. In *Proc. of IEEE Congress on Evolutionary Computation CEC*, pages 365–372. IEEE, 2009.

[MK11] K. McClymont and E. Keedwell. Markov chain hyper-heuristic (mchh): an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 2003–2010, New York, NY, USA, 2011. ACM.

[MLS10] J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *J. Heuristics*, 16(6):881–909, 2010.

[NE07] 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.

[Rob52] H. Robbins. Some aspects of the sequential desing of experiments. *Bulletin Amrican Mathematical Society*, (55):527–535, 1952.

[Rod78] L. Rodman. On the many-armed bandit problem. *The Annals of Probability*, 6(3):491–498, 1978.

[SB98] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[SJ05] Z. Skolicki and K. De Jong. The influence of migration sizes and intervals on island models. In *GECCO*, pages 1295–1302, 2005.

[Thi05] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Genetic and Evolutionary Computation Conference, GECCO*, pages 1539–1546. ACM, 2005.

[WM97] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation*, 1(1):67–82, 1997.

[WRH98] D. Whitley, S. Rana, and R. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47, 1998.