On Migration Policies in Dynamic Island Models

Jorge Maturana, Frédéric Lardeux, Frédéric Saubion

On Migration Policies in Dynamic Island Models

Jorge Maturana[1], Frédéric Lardeux[2], and Frédéric Saubion[2]

[1] Universidad Austral de Chile
[2] LERIA, University of Angers (France)

Abstract. Dynamic island models are population-based algorithm where individuals are located on islands that executes a different algorithm. The individuals are managed by a migration process that evolve during the search according to the observed performance on the islands. We propose a testing framework that assigns gains to the algorithms applied on the island in order to assess the adaptive ability of the migration policies with regards to various situations.

1 Introduction

Island models (IM) [WRH98,Sko07] have been introduced in evolutionary computation in order to avoid premature convergence in population-based algorithms when solving optimization problems. The main idea of IM is to use a set of subpopulations instead of a panmictic one, in order to improve the performance of the evolutionary process. Classically, islands models use the same algorithm on each island and the islands differ only by their populations. In [LG10] it has been proposed to consider different algorithms on the islands - restricted in fact to a basic variation operator - and to define dynamic migration policies. In this approach, called Dynamic Island Models (DIM), migration probabilities change during the evolutionary process by means of a learning process. Compared to classic island models, since only one operator is used on each island, DIM is indeed related to adaptive operator selection techniques for evolutionary algorithms [DFSS08]. DIM should be able to identify a subset of islands that are currently appropriate for improving individuals, but also to quickly react to changes when other operators become more beneficial [CGLS12]. DIM has been compared to other adaptive operator selection policies [LG10,CGLS12].
The purpose of this paper is to carefully study different configuration of the DIM with dynamic migration policies, as well as their ability to adapt to changes during the solving process. Such changes occur when the solving process explores different areas of the search space. Therefore, the basic search heuristics or operators may become more or less efficient according to the current state of the search. We propose here a testing model in order to simulate the evolution of the search efficiency on the islands. In such (surrogate) models, gains are associated to operators of the island in order to reflect their performances. Compared to previous models, we consider a gain matrix that take into account possible interactions between operators, i.e., the efficiency of an operator applied on a given individual may depend on the previous operators applied on it. This is motivated

by the fact that, in search processes, such dependencies may occur between operators alternating intensification and diversification stages or operators using complementary neighbourhoods.
Our study highlights that DIM is efficient for tracking interactions between islands and to quickly react to efficiency changes during the search. We introduce a new configuration of the DIM, called Specialized Collaborative Model, using simultaneously several settings of the dynamic migration process by means of different types of individuals, which improves the performance.

2 Dynamic Island Model

A Dynamic Island Model (DIM) [LG10] is defined by:

- its size n
- a set of islands $\mathcal{I} = \{i_1, \cdots, i_n\}$ and a set of algorithms $\mathcal{A} = \{a_1, \cdots, a_n\}$. Each algorithm a_k is assigned to island i_k.
- a set of populations $\mathcal{P} = \{p_1, \cdots, p_n\}$. Each population is a subset of individuals. Each population p_k is assigned to island i_k. The size of the entire population is fixed but the size of each p_k changes continuously according to the migrations. $a_k(p_k)$ is the population obtained after applying algorithm a_k on population p_k.
- a topology given by an undirected graph $(\mathcal{I}, V)$ where $V \subseteq \mathcal{I} \times \mathcal{I}$ is a set of edges between islands (here we will consider a complete graph).
- an initial migration matrix M of size $n \times n$ with $M(i,j) \in [0..1]$. M is supposed to be coherent with the topology, i.e., if $(i,j) \notin V$ then $M(i,j) = 0$, $\mathcal{M}$ is the set of migration matrices.
- a migration policy $\Pi : \mathcal{I} \times \mathcal{M} \to \mathcal{I}$ that selects a migration island given an initial island and a migration matrix.

Description of the components of the algorithm :

- In this paper, we define a notion of gain associated to each algorithm located on the islands that simulates the effect of its application on the individuals of the population. For instance, this gain can be the fitness improvement with regards to a classic optimisation problem. Of course, this does not take into account the fact that the performance of an algorithm a depends, most of the time, on the semantics - phenotype and/or genotype - of the individuals. Nevertheless, such testing scenarios for EAs have been widely used for studying control of operators [Thi05,DFSS08]. We consider a function $gain : \mathcal{A} \times \mathbb{N} \to \mathbb{R}$, such that $gain(a,t)$ is the gain of algorithm a when processed at iteration t of the DIM. Individuals may be abstracted by the sum of their successive gains. For an individual $s \in p_i$ at iteration t, we define its value at iteration t $v(s,t) = \Sigma_{\tau=1}^{t} gain(a_{s(\tau)}, \tau)$, where $s(\tau)$ is the island where s was located at iteration τ.
- The value $R(i,j)$ of reward matrix R evaluates the benefit (by means of rewards) of sending individual from island i to island j. R is used to update the migration matrix M by means of a reinforcement learning based process.

```
input  : a DIM, a gain function
output: a solution s*
local  : a reward matrix R of size n × n
s* ← best(P);
R ← 0;
while not stop condition do
    for k ← 1 to n do
        Update(R, p_k);
        p_k ← a_k(p_k);
        for s ∈ p_k do
            i_l ← Π(i_k, M);
            p_l ← p_l ∪ {s};
            p_k ← p_k \ {s};
        Learning(M, R);
    b ← best(P);
    if b > s* then
        s* ← b;
```

Algorithm 1: Dynamic Island Model

- The function *best* computes the best current individual of the whole population, $best(\mathcal{P}) = best(\cup_{i\in\mathcal{I}}(p_i))$, according to their values.
- The stop condition is, as usual, a limited number of iterations or the fact that an optimal solution has been found in the global population.

Since M and R will be changed at each iteration of the algorithm, let us denote $M^{(t)}$ and $R^{(t)}$ the value of these matrices at iteration t of the algorithm.

Reward function: Note that $Reward(R, p_i)$ is performed for each island i and will affect the i^{th} line vector R_i of R. $R_i^{(t)}(k)$ corresponds to the reward assign to individuals that were on island i at iteration $t-1$ and that have been processed on island k at iteration t. We consider two possible reward functions for computing $R_i^{(t)}(k)$.

$$\text{Elitist Reward: } R_i^{(t)}(k) = \begin{cases} \frac{1}{|B|} & \text{if } k \in B, \\ 0 & \text{otherwise,} \end{cases} \text{ with}$$
$$B = \underset{k\in\{1,\ldots,n\}}{\operatorname{argmax}} \left(\{v(s,t) - v(s,t-1) | s(t) = k, s(t-1) = i\}\right)$$

B is the set of the indices of the islands k where individuals coming from i at iteration $t-1$ have obtained the best gain improvements at iteration t.

$$\text{Proportional Reward: } R_i^{(t)}(k) = \frac{\Sigma_{s\in K} v(s,t)}{|K|}, \text{with } K = \{s \in p_k^{(t)} | s(t-1) = i\}$$

Note that K is the set of the individuals of the island k at iteration t that were on island i at iteration $t-1$.

Learning Function: The basic learning principle consists in sending more individuals to the islands that have previously improved individuals coming from

the current island and less to the islands that are currently less efficient. The learning process is achieved by an adaptive update of the migration matrix at iteration t, $M^{(t)}$, performed as:

$$M^{(t+1)}(i,k) = (1-\beta)(\alpha.M^{(t)}(i,k) + (1-\alpha)R_i^{(t)}(k)) + \beta.N^{(t)}(k)$$

where $N^{(t)}$ is a stochastic noise vector. The parameter α represents the importance of the knowledge accumulated (inertia or exploitation) and β is the amount of noise, which is necessary to explore alternative actions. The influence of these parameters has been studied in [CGLS12].

Migration Policies: We consider two possible migration policies Π,

- Proportional migration: for each individual s on island i the classic migration process consists in sending this individual according to a probability on line vector $M_i^{(t)}$. Note that M is normalized in order to insure good probability properties.
- Elitist migration policy: individuals from island i migrate to the island j that has the highest value in line vector, i.e. $\text{argmax}_j\, M^{(t)}(i,j)$. Such migration policy promotes intensification of the search process toward the most efficient islands.

Configurations of the DIM:

- CIM is a classic DIM that uses the elitist reward and proportional migration as proposed in previous works [LG10,CGLS12].
- $SCoM$ (Specialized Collaborative Model) is a new model using a proportional reward for the update of the matrix. Compared to previous dynamic migration methods, it allows to benefit from several possible migration policies in the same DIM. It uses two types of individuals: *champions* (C), that migrate with an elitist migration and *proportionals* (P) that migrate with a proportional rule. Preliminary experiments have been performed and show us the need to exclude *champions* to contribute to update matrix M because an excessive reinforcement of first-found migrations discourage the exploration of other alternatives. We have also tested the combinations "proportional reward - proportional mutation" and "elitist reward - champions and proportionals", which achieved very poor results and are thus not presented here.

3 Assessing the Efficiency of Migration Policies

In this section, we are interested in two main aspects: introduce changes in the efficiency of the islands and take into account dependencies between islands in order to discover possible cooperative sequences
Given a DIM and a time horizon T, the efficiency of its migration policy is defined by the value $\Sigma_{t=1}^{T} gain(a_{i(t)}, t)$ obtained by its best individual s^* after T iterations, where $a_{i(t)}$ is the algorithm that has been applied on this individual

on island $i(t)$ at iteration t. In this context, an optimal policy corresponds to the best sequence $a_{i(1)}, \cdots, a_{i(T)}$ (that also corresponds to the best visiting sequence of islands $i(1), \cdots, i(T)$). In order to assess the ability to adapt the migration policy to changes, we introduce a hidden gain matrix.

Definition 1 (Hidden Gain Matrix). *Given a DIM we define a sequence of matrices $H^{(t)}$, for each iteration t of the algorithm, of size $|\mathcal{A}|^2$ such that $gain(a_k, t) = H^{(t)}(j, k)$ if $a_{i(t-1)} = a_j$ (i.e., gain from j to k).*

In this model, the gain obtained by action a_k depends on the action a_j that has been previously applied on the considered individual. This general model allows us to take into account dependencies between search operators that should be used sequentially. Of course, H is not known by the IM. Note that while $H^{(t)}$ encodes gains, $M^{(t)}$ encodes migration probabilities. Nevertheless, the accuracy of the learning process will be easy to assess by comparing the structures of H and M. Note that, for an individual s, $v(s,t) = \Sigma_{k=2}^{t} H^{(k)}(s(k-1), s(k))$. When solving real problems, the gains associated to the application of the search operators are likely to change over time. In order to simulate this behaviour in our model, H will be a dynamic in our experiments, with changing values $H^{(t)}$. As base line for comparisons, we consider the following policies:

- A myopic oracle (OR) which knows the hidden matrix and selects, at iteration $t+1$, $\text{argmax}_j\, H^{(t+1)}(i,j)$ if action a_i was selected at iteration t.
- An optimal oracle (OPT) which selects the best sequence with a global view (i.e., computes the best possible score).
- A uniform selection (U) that selects uniformly an action at each iteration.

4 Experiments

Three basic 10×10 gain matrices A, B and C have been defined. These matrix represent typical situations with different types of dependencies between islands. Based on A, B and C, we will define either constant H such that $H^{(t)}$ is always equal to one of these matrix or changing H. The gains are illustrated on Figure 1. The thickness of the arrows from i to j is proportional to the associated gain $H^{(t)}(i,j)$. Of course, even if there is no line between some islands, migrations are always possible but with a null gain in $H^{(t)}$, which means that no benefit is obtained by using the operator j after the operator i (e.g., if operators have opposite effects). Note that C has a gain cycle $(2-10-9-8-7-5-2)$, however it is suboptimal compared to the optimal cycle $(2-10-7-5-2)$.

4.1 Results Using Constant Hidden Matrices

All the different policies presented above are tested on three constant hidden matrix independently. For the DIM algorithms described in Section 2, we use 20 individuals, 30 runs of 600 iterations. The baseline policies described in Section 3 are tested on 20×30 runs of 600 iterations.

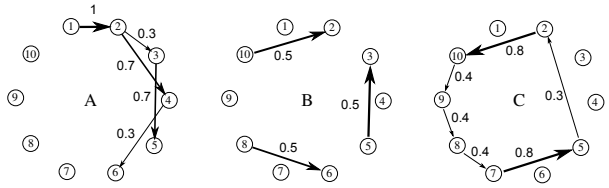

Fig. 1. Representation of possible gains of H.

Parameters The parameters of these methods have been obtained using a methodological tuning approach based on F-Race [BSPV02]. $SCoM$ uses 3 Champions and 17 Proportionals, only the later contributing to the update of M (learning process). Both CIM and $SCoM$ use $\alpha = 0.8, \beta = 0.01$ as proposed in [CGLS12].

Results Table 1 shows the mean value of the total gain and its standard deviation for the 30 runs. We also provide a measure of global performance, called *Performance Percentage* (PP), defined as $PP = 1/N \sum_i^N g_i/g_i^*$ where N is the number of iterations, g_i is the gain obtained by the best individual in the population at iteration i and g_i^* is the optimal mean gain for the current hidden matrix H at iteration i. This criterion is adapted to other selection policies that do not use populations by considering the best score among 30 runs at each iteration. CIM and $SCoM$ get significantly better results than the other ones, according to a T-test at 95% confidence, which constitutes a good preliminary result before exploring their behaviour when H changes over time.

Method	$H = A$			$H = B$			$H = C$		
	Mean	SD	PP	Mean	SD	PP	Mean	SD	PP
OPT	340.00	–	1.00	300.00	–	1.00	315.00	–	1.00
OR	155.23	4.37	0.46	137.57	3.48	0.45	239.90	0.00	0.76
U	24.83	1.85	0.07	22.98	2.35	0.08	21.27	1.41	0.07
CIM	113.78	4.80	0.33	103.64	5.04	0.35	215.73	3.62	0.68
SCoM	150.60	4.03	0.47	131.76	3.24	0.41	237.18	1.52	0.75

Table 1. Results obtained by different methods when solving a constant H

4.2 Results Using Changing Hidden Matrices

We study how the different policies react to changes of the hidden matrix. We simply change $H(t)$ using sequentially A-B-C every 100 iterations, and compare the performances of the policies.
OR is very efficient on C, that has a clearly well defined path, but it get lost on A and B (no clear path to follow). The result is certainly due to the fact that OR, by definition, checks only one step forward. CIM obtains similar results as

Meth.	Parameters	Mean	SD	PP
OPT	–	318.33	–	1.00
OR	–	178.16	2.92	0.56
SCoM	C3P17-P	136.68	18.78	0.43
CIM	$\alpha = 0.8, \beta = 0.01$	127.88	4.52	0.40

Table 2. Results obtained when changing H during the run for initial comparisons.

in the fixed matrix case (see Table 1). This also seems coherent since, if elitist reward and proportional migration are applied on a clear path, then it is likely that the DIM will be able to identify it.
Besides OR, $SCoM$ obtains the best results for the initial comparison (according to a T-test at 95% confidence). Nevertheless, it is interesting to note that, while $SCoM$ obtains good results the first time it solves B, it is different for the second time. Looking at the population, remark that since *champions* do not perform exploration, the *proportional* individuals will not be able to efficiently update the migration matrix M. There is thus a need for increasing the exploration ability of *SCoM*, as proposed in the next subsection. Let us also note that $SCoM$ and CIM are both faster than baseline methods with a factor 4 and 8, respectively.

4.3 Improving the SCoM Policy

As mentioned before, a solution for improving policies consists in increasing their exploration ability (i.e., well known exploration vs. exploitation dilemma in reinforcement learning). Since $SCoM$ obtains the best results, we focus now on this DIM. Remind that $SCoM$ uses two types of individuals (champions and proportionals). A third type could now be considered, called *Explorer* (E), that chooses uniformly the next island to migrate on.
We define different configurations of $SCoM$, mainly by changing the composition of the population (C, P and E individuals) and selecting whether all or just one type of individual will be used to update M. Table 3 shows the results of a selected number of combinations of $SCoM$. The names correspond to the configurations, for instance, *P17E3-E* corresponds to a population of 17 *proportionals* and 3 *explorers*, when only the latter contribute to update M.
By looking at the results on Table 3, the following conclusions may be drawn:

- *Champions* often produce better results than Proportionals especially in static conditions, but they are unable to escape from a well-defined migration path even if its efficiency decreases, due to a change in H.
- Even though *Proportionals* obtain slightly less gain than Champions, the fact that they use a proportional migration provides them with an opportunity to escape from local optima when gains decrease (acting as "open-minded champions").
- *Explorers* never obtain good results, but they are a key feature in order to obtain a comprehensive and efficiently updated M.

Meth.	Parameters	Mean	SD	PP
SCoM	C10P10-P	156.17	4.47	0.50
SCoM	C10E10-E	173.98	13.74	0.55
SCoM	C17E3-E	143.81	24.97	0.46
SCoM	P10E10-All	189.92	7.08	0.61
SCoM	P10E10-E	126.56	10.76	0.40
SCoM	P17E3-E	79.36	9.10	0.25
SCoM	P3E17-E	152.50	7.52	0.48
SCoM	P17E3-All	161.83	17.48	0.52

Table 3. Results obtained when changing H during the run.

Remark that a combination of *Proportionals* and *Explorers* outperforms other policies that include *Champions*. It seems that the gain obtained by *Proportionals* are good enough to prescind from *Champions*. The prominence of policies where all individuals contribute to M (*-All*) supports the idea of using *Proportionals* either to gather high gains and to explore the search space.

References

[BSPV02] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[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.

[DFSS08] Luis Da Costa, Álvaro Fialho, Marc Schoenauer, and Michèle Sebag. Adaptive operator selection with dynamic multi-armed bandits. In M. Keijzer et al., editor, *Proc. GECCO'08*, pages 913–920. ACM Press, 2008.

[LG10] Frédéric Lardeux and Adrien Goëffon. A dynamic island-based genetic algorithms framework. In *SEAL*, pages 156–165, 2010.

[Sko07] Zbigniew Skolicki. *An Analysis of Island Models in Evolutionary Computation*. PhD thesis, George Mason University, 2007.

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

[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.