Learning and Migration Processes for Island Models

Frédéric Lardeux, Frédéric Saubion
Universit d'Angers
LERIA
Angers, France
{firstname.lastname}@univ-angers.fr

Jorge Maturana
Instituto de Informática, Universidad Austral de Chile (Chile)
jorge.maturana@inf.uach.cl

***Abstract*—Dynamic island models are population-based algorithms for solving optimization problems, where the individuals of the population are distributed on islands. The purpose of this dynamic/adaptive management of the migrations is to send the individuals to the most promising islands, with regards to their current states. In this paper, we propose to investigate Q-Learning as migration policy.**

***Index Terms*—Island Models; Adaptive Operator Selection**

I. Introduction

Island models (IM) [1], [2] 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 sub-populations instead of a panmictic one, in order to improve the performance of the evolutionary search process. IM are thus closely related to parallel evolutionary computation [3]. Each sub-population evolves independently on each island and interacts periodically with other islands by means of migrations [4].

The impact of migration has been carefully studied [5]. Migrations may be actually used in order to reinforce the most efficient islands [6]. Considering the same algorithm on all islands, one may be interested in assessing the convergence ability by evaluating two complementary aspects: (1) the ability to converge on each island (e.g., the ability to obtain the best individuals on all islands using for instance the notion of takeover time [5]) and (2) the ability to ensure a good global diversity to avoid sub-populations to get stuck in local optima and finally reach a global optima/.

While, in classic islands models, the same algorithm is executed on each island, other island models consider different algorithms on the islands (see for instance [7]). In particular in Dynamic Island Models (DIM) [8], the islands are restricted in fact to a basic evolutionary algorithm with only one variation operator and dynamic migration policies are introduced. These policies are achieved by means of migration probabilities that are modified during the evolutionary process according to the impact of previous analogue migrations, using a learning process. Therefore DIM have be related to adaptive operator selection techniques for evolutionary algorithms [9], since only one operator is used on each island. A DIM should be able to identify a subset of islands that are currently the most suitable for improving individuals, but also to quickly react to changes when new operators become more efficient.

In this paper we propose to improve migration policies by considering more possible configurations. In previous works [8], [10], migrations are performed according to a migration matrix M, where $M(i, j)$ is the probability for an individual to migrate from island i to j. This matrix is updated by a simple reinforcement learning process that takes into account the quality improvement obtained of island j by individuals from island i in order to regulate beneficial migrations. Therefore, in DIM, the key action is to migrate from an island to another one according to an expected gain, which can also be handled by Q-learning [11], [12], which is a method for finding optimal action policies. Note that Qlearning has already been used for controlling and tuning parameters in evolutionary algorithms [13], [14].

In this context, given a migration matrix, an individual may choose its destination according to several strategies: greedy (higher value), proportional choice, random choice... Therefore, the are many possible combination of basic components that lead to different migration processes. While in previous works, the same dynamic migration policy has been investigated, we propose here to study several possible configuration of the DIM, by considering more possible components, including learning and migration processes. Instead of selecting one setting of the DIM, we propose thus to attach the policy to the individuals , in order to take advantage of our multi-individuals algorithm. This cooperative model allows us to use simultaneously several migration policies in order to benefit from their respective properties. Our experiments show that:

- using QLearning is interesting for managing migrations in dynamic island models,
- using a population of individuals that cooperate by exchanging informations improves the QLearning performance for scheduling the operators that must be used along the search compared to a single learning process,
- considering simultaneously different type of individuals that use different migration policies is also beneficial and leads to good results compared to previously proposed dynamic migration approaches in the context of our search scenarios.

II. Dynamic Island Models

A Dynamic Island Model (DIM) is defined by 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. Each island i_k is equipped with a population p_k and $\mathcal{P} = \{p_1, \cdots, p_n\}$ is the global population. The size of $\mathcal{P}$ 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 is defined by an undirected graph $(\mathcal{I}, V)$ where $V \subseteq \mathcal{I} \times \mathcal{I}$ is a set of edges between islands (here we consider a complete graph).

The migration of individuals between islands are performed according to a migration matrix M of size $n \times n$ with $M(i,j) \in [0..1]$. M is assumed to be coherent with the topology, i.e., if $(i,j) \notin V$ then $M(i,j) = 0$. A migration policy $\Pi : \mathcal{I} \times \mathcal{M} \to \mathcal{I}$ selects a migration island given an initial island and a migration matrix.

Given a DIM, its computational behaviour is described by Algorithm 1.

input : a DIM, a fitness function, an intial migration matrix M
output: a solution s^*
local : a reward matrix R of size $n \times n$

$s^* \leftarrow best(\mathcal{P})$;
$R \leftarrow \mathbf{0}$;
$Initialize(M)$;
while *not stop condition* **do**
  for $k \leftarrow 1$ **to** n **do**
    $Reward(R, p_k)$;
    $p_k \leftarrow a_k(p_k)$;
    for $s \in p_k$ **do**
      $i_l \leftarrow Migrate(i_k, M)$;
      $p_l \leftarrow p_l \cup \{s\}$;
      $p_k \leftarrow p_k \setminus \{s\}$;
    $Learn(M, R)$;
  $b \leftarrow best(\mathcal{P})$;
  if $b > s^*$ **then**
    $s^* \leftarrow b$;

Algorithm 1. Dynamic Island Model

Description of the components of the algorithm:

- R is the reward matrix whose values $R(i,j)$ evaluate the benefit (by means of fitness improvement) of sending individual from island i to island j. R is used to update the migration matrix M, using reinforcement learning. Note that R is initialized with 0 values. M can be initialized with uniform values for all $M(i,j)$ corresponding to equal probabilities of migration for any pair of islands.
- The function $best$ returns the best individual of the current global population, $best(\mathcal{P}) = best(\cup_{i \in \mathcal{I}}(p_i))$.
- The stop condition is a limited number of iterations or the fact that an optimal solution has been found in the global population $\mathcal{P}$.

Let us now focus on the most important components used in the migration process. 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.

A. Reward Function

$R_i^{(t)}(k)$ corresponds to the reward assigned 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 $R_i^{(t)}(k)$: either only the most improved individuals are considered for assessing the benefit of the migration or all individuals are taken into account according to the relative improvement.

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

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

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

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

B. Learn Function

In DIM, we expect to learn from previous migrations in order to adapt the migration process. This problem is clearly a reinforcement problem, which consists in, given an individual, selecting the most suitable migration (action) from an island i to an island j. We recall a simple reinforcement learning function used in [8] and we also propose to consider Qlearning technique [15].

Basic Reinforcement 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 [8].

QLearning Based Function

Since one main challenge in DIM is to manage efficiently the migration of individuals from island i to island j according to collected feedback information, learning the most suitable migration actions at each step of the search seemed fully relevant to Qlearning techniques. We use a classic QLearning (see [12] for more details) algorithm in order to update the transition matrix. Compared to previous function, QLearning takes into account an estimation of the future optimal value that can be obtained after a migration has been performed.

$$M^{(t+1)}(i,k) = M^{(t)}(i,k) + \delta(R_i^{(t)}(k) + \gamma max_j M^{(t)}(k,j) - M^{(t)}(i,k))$$

where δ is the learning rate and γ is a discount factor that allows to control the importance of the estimation of expected future gains.

C. Migrate Function

Once matrix M has been updated, several migration functions can be considered to select for individuals from island i their next island j. We consider here three possible choice, using different degree of greediness.

- *Elitist migration*: 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 promotes intensification of the search process toward the most efficient islands.
- *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.
- *Uniform migration*: individuals from island i migrate to the island j at random uniformly.

D. Configurations of the DIM: Setting the Policy

We propose to link migration policies to individuals, allowing thus the DIM to use simultaneously different policies within the same search process. This is motivated by the fact that the DIM is particularly well suited to the management of collaborative policies. Moreover, this generic approach allows us to consider and compare various settings of the DIM.

We may first remarks that different migrate functions may be used in the same DIM. But it is not possible to use different Learn and Reward functions at the same time since they manage the matrices R and M differently, involving incompatible update processes. Based on the previously described components, we propose the following taxonomy in order to define different configurations of the DIM. A policy for a DIM will be described by a tuple $(type_l type_r type_m)$ where

- $type_l$ corresponds to the type of learn functions (see Section II-B), $type_l \in \{C, Q\}$, (C)lassic or (Q)learning
- $type_r$ corresponds to the type of reward functions (see Section II-A), $type_r \in \{E, P\}$, (E)litist or (P)roportional
- $type_r$ corresponds to composition of the population concerning the migration functions (see Section II-C) and is a tuple $Elit - Prop - Unif$ with $Elit, Prop, Unif \in \{0, 1\}$. For instance, $1-1-1$) means that one individual uses elitist migration, one uses proportional migration, and one uses uniform migration.

Note that we may consider here pure policies, i.e., configurations that use only one type of individuals as well as mixed configurations were the migration policy is not the same for all individuals. For instance, $CE0-1-0$ corresponds to the basic DIM that has been previously studied in [8]. Note that we use a * to denote generic configuration, e.g. CE * will represent any configuration that uses C as learn function and E as reward function.

III. Experiments

A. Description of the Problem

The Nk-landscape problem [16] is a problem-independent model for constructing multi-modal landscapes that can gradually be tuned from smooth to rugged. The parameter of this model are N, the number of (binary) genes, and k, the number of genes that influence a particular gene. By increasing the value of k from 0 to $N-1$, Nk-landscapes can be tuned from smooth to rugged. The k variables that form the context of the fitness contribution of gene s_i can be chosen according to different models.

In our experiments, we use as set of 8 instances of Nk-landscape of sizes $128, 256$ and 512 and different values of K from 2 to 8. Since this is a binary problem, we consider here 4 classical binary mutation operators: *bit-flip* that flip each bit of an individual with a probability $1/N$ and *p-flip* operators with $p \in \{1, 3, 5\}$ that change randomly p bits in an individuals. The goal is to find the best possible solution. Note that only improving mutations are taken into account (i.e., the individual is not replaced by a mutated individual of lower fitness).

B. Experimental Settings and Method

Each configuration has been run 20 times on the 8 problem instances. The size of the population will be studied in Section III-D. The parameters of the Learn functions have been set using recommended default values $\alpha = 0.8, \beta = 0.01, \delta = 0.8, \gamma = 0.1$. Preliminary experiments have been performed to assess the validity of these parameter setting. Our purpose is to study experimentally the following properties of DIM:

- Impact of QLearning: since Qlearning is introduced in this work as an alternative learning technique for managing the migration matrix, we want to study its efficiency by comparing different possible Qlearning based configurations.
- Impact of the population size: since we introduce different types of individuals, each one defining its own migration policy and collaborating through the learning

	Q $(1-\epsilon)\times 20$		Q 10-0-10		Q 10-10-0		Q $(1-\epsilon)$	
Instance	avg (rk)	std	avg (rk)	std	avg (rk)	std	avg (rk)	std
128_2.0	95.491 (1)	0.168	**95.554 (1)**	0.310	95.504 (1)	0.266	95.063 (4)	0.752
128_4.0	96.075 (1)	0.532	96.233 (1)	0.741	**96.372 (1)**	1.122	94.782 (4)	1.832
128_8.0	95.150 (3)	0.964	95.600 (1)	0.926	**95.667 (1)**	0.696	94.363 (4)	2.007
256_2.0	186.416 (4)	0.805	186.822 (1)	0.924	186.701 (1)	0.910	**186.858 (1)**	1.119
256_4.0	189.000 (3)	1.023	**189.597 (1)**	1.716	189.115 (1)	1.144	186.892 (3)	2.575
256_8.0	188.412 (1)	2.881	**189.120 (1)**	1.443	188.364 (3)	1.286	185.953 (4)	2.585
512_2.0	371.875 (2)	1.641	**372.522 (1)**	1.097	371.321 (3)	1.555	370.852 (4)	2.020
512_4.0	377.828 (1)	3.477	**378.218 (1)**	1.522	377.115 (3)	1.518	373.699 (4)	4.002
Avg Rank	2		1		1.75		3.5	

TABLE I
DIFFERENT CONFIGURATIONS WITH QLEARNING ON NK-LANDSCAPE INSTANCES WITH RANKING PERFORMED ACCORDING TO A T-TEST (1%).

mechanism, we propose to evaluate the influence of the number of individuals on the quality of the learning process.
- Impact of using several types of individuals against a single migration policy: we propose to highlight the improvement in terms of operator management due to the use of mixed policies.

C. Impact of Qlearning

Our first investigations consist in assessing the performance of Qlearning in DIM. Qlearning techniques often choose the next action by means of a ϵ-greedy strategy (i.e., choose the best next move according to the transition matrix learned from formula presented in Section II-B with probability $(1-\epsilon)$ and choose a random action with probability ϵ). We thus consider a $(1-\epsilon)$ migration function. In table I, Q $(1-\epsilon)\times 20$ is a DIM with $(1-\epsilon)$ migration function using 20 individuals while Q $(1-\epsilon)$ uses only one individuals but benefit from 20 times more runs, in order to ensure computational fairness. ϵ is set to 0.05 (using 0.01 provides similar results, while higher values provide poor results).

In order to simulate similar policies, we consider here two configurations of the DIM with 20 individuals: Q 10-10-0, Q 0-10-10. These two configurations involve individuals that use elitist migrations with individuals that may diversify the exploration of islands (proportional or uniform). We restrict our choice to these two configurations, other Qlearning configurations will be considered and compared in Section III-E. Note that, for Qlearning, we have performed tests showing that the reward function has no influence on the results. Therefore, we omit this function in the configurations, which are just called Q*.

Table I presents the results obtained on the different Nk-landscape instances. Results in bold are the best results for each instance. For each configuration, a ranking is performed according to a T-test: two methods that obtain results that are not statistically different according to a T-test at 1% are assigned to the same rank.

We observe that using several individuals (i.e., Q $(1-\epsilon)\times 20$) instead of a single one (here, Q $(1-\epsilon)$ may be seen as an adaptive local search with different operators) improves the results. Moreover using combinations of different types of individual in order to have a policy similar to $(1-\epsilon)$ is also interesting. Note that in Q 10-0-10, contrary to Q $(1-\epsilon)$ policy, the uniform migration is separated from greedy migration, since linked to different individuals.

Therefore, exploration performed by uniform migration does not impact the greedy choice performed by elitist migration, but rather attempts to improve the global learning process. Note also that using a number of individuals that use random migration close to the proportion ϵ (i.e., Q 19-0-1) provides similar results, which shows the reliability of our DIM with several types of individuals.

D. Impact of the population size

DIM takes advantage of the successive generation of individuals that contribute to the migration matrix and, of course, the number of individuals is an important parameter. The main question is: "Given a computational budget, is it more beneficial to use more individual with fewer runs of the algorithm or more runs with fewer individuals ?" Figure 1 presents the results on the 128_8.0 nk-landscape instance and Q, where individuals are equally dispatched between the different migration functions. Similar results were observed for other configurations and instances.

FM fitness (resp. FM time) represents the evolution of the fitness (resp. time) with regards to the population size (on X axis), when the same number of migrations (set ot $10,000$) is used for each population size. SP fitness (resp. SP time) represents the evolution of the fitness (resp. time) when the same computation time is used (SP time curve is thus constant).

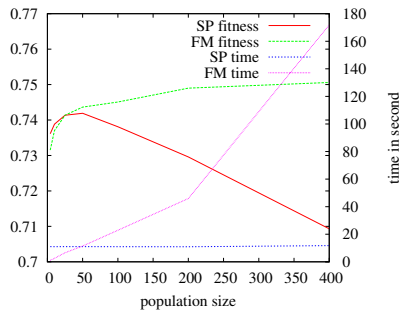
Let us remark that, even if the number of allowed migration is fixed, increasing the size of the population improve the results up to a given limit (SP fitness curve). Of course with more individuals (FM fitness) and the same number of migration results may be improved but with a drastic increase of computation time. Therefore, it seems that a population size around $20-25$ individuals constitutes a good compromise.

E. Evaluating Different Configurations

Table II is a compilation of tests realized on the 8 Nk-landscape instances for all learning and reward functions with a number of migrations fixed to 10000. These experiments

Population	Learn and reward functions							
$Elit, Prop, Unif$	CE		CP		Q		Rand	
25,0,0	128	(3)	104	(2)	71	(1)	158	(4)
0,25,0	54	(2)	125	(3)	50	(1)	158	(4)
12,13,0	57	(1)	63	(2)	103	(3)	158	(4)
12,0,13	35	(1)	51	(2)	63	(3)	158	(4)
0,12,13	69	(3)	56	(2)	47	(1)	158	(4)
8,8,9	67	(3)	50	(1)	59	(2)	158	(4)
average	68.33	(2.17)	74.83	(2)	65.5	(1.83)	158	(4)
average sum	68.33	(2)	74.83	(3)	65.5	(1)	158	(4)
average rank	2.17		2		1.83		4	

TABLE II
COMPARISONS OF DIFFERENT CONFIGURATIONS



Qlearning

Fig. 1. Impact of population size on fitness and computation time

use a population of 25 individuals with different repartitions of migration types (uniform repartition on 1, 2 and 3 migration types).

We consider also pure random migration policies (i.e., $*$ 0-0-n) that are simply named as Rand since, in this case the learning process is not used. For each population and each instance, results are sorted and a rank is assigned to the configuration. This rank uses a statistical student T-Test as in Section III-C. Each cell corresponds to the sum of the 8 ranks obtained on the 8 instances. Values in bracket are the rank for each line (i.e., each repartition of individuals types). Last line provides the average rank. These results highlight that combining several types of individuals within the same DIM generally improves the migration process (results in column for each learn/reward combinations. Concerning Qleaning, it seems, as expected useful to introduce a compromise between exploitation (elitist) and exploration. Note that using proportional migration seems very efficient. We may also remark that no combination of learn/reward function obtains the best results for all combinations of types of individuals. Nevertheless, using Qlearning seems to be a reasonable choice.

IV. CONCLUSION

In this work, we propose to study the management of migrations in islands models. We consider a QLearning approach in order to learn what are the best migration choices for individuals at a given state of the search. We evaluate these possible configurations of the DIM on a set of NK-Landscape benchmarks. Our results highlights that Qlearning is an efficient learning process for managing migration in island models. Moreover, considering a population of individuals that use different type of migration is efficient compared to previous global dynamic migration policies. In particular, Qlearning with a population of individuals that use different type of migration is interesting compared to a classic Qlearning approach with an ϵ-greedy strategy.

REFERENCES

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

[2] Z. Skolicki, An analysis of island models in evolutionary computation, Ph.D. thesis, George Mason University (2007).

[3] N. Melab, M.-S. Mezmaz, E.-G. Talbi, Parallel hybrid multi-objective island model in peer-to-peer environment, in: 19th International Parallel and Distributed Processing Symposium (IPDPS 2005), IEEE Computer Society, 2005.

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

[5] G. Luque, E. Alba, Selection pressure and takeover time of distributed evolutionary algorithms, in: M. Pelikan, J. Branke (Eds.), Genetic and Evolutionary Computation Conference, GECCO 2010, ACM, 2010, pp. 1083–1088.

[6] Z. Skolicki, K. D. Jong, The influence of migration sizes and intervals on island models, in: GECCO, 2005, pp. 1295–1302.

[7] C. Jankee, S. Vérel, B. Derbel, C. Fonlupt, Distributed adaptive metaheuristic selection: Comparisons of selection strategies, in: 12th International Conference, Evolution Artificielle, EA, 2015, pp. 83–96.

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

[9] L. Da Costa, A. Fialho, M. Schoenauer, M. Sebag, Adaptive operator selection with dynamic multi-armed bandits, in: M. Keijzer et al. (Ed.), Proc. GECCO'08, ACM Press, 2008, pp. 913–920.

[10] A. Goëffon, F. Lardeux, F. Saubion, Simulating non-stationary operators in search algorithms, Appl. Soft Comput. 38 (2016) 257–268.

[11] C. Watkins, Learning from delayed rewards, Ph.D. thesis, University of Cambridge, England (1989).

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

[13] A. E. Eiben, M. Horváth, W. Kowalczyk, M. C. Schut, Reinforcement learning for online control of evolutionary algorithms, in: Engineering Self-Organising Systems, 4th International Workshop, ESOA, Vol. 4335 of LNCS, Springer, 2006, pp. 151–160.

[14] G. Karafotias, Á. E. Eiben, M. Hoogendoorn, Generic parameter control with reinforcement learning, in: Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014, ACM, 2014, pp. 1319–1326.

[15] C. J. C. H. Watkins, P. Dayan, Technical note q-learning, Machine Learning 8 (1992) 279–292.

[16] S. A. Kauffman, The Origins of Order: Self-Organization and Selection in Evolution, 1st Edition, Oxford University Press, USA, 1993.