Natural Computing (2019) 18:163–179
https://doi.org/10.1007/s11047-017-9660-z

Migration policies in dynamic island models

Frédéric Lardeux[1] · Jorge Maturana[2] · Eduardo Rodriguez-Tello[3] · Frédéric Saubion[1]

Published online: 28 November 2017

Abstract
Dynamic island models are population-based algorithms for solving optimization problems, where the individuals of the population are distributed on islands. These subpopulations of individuals are processed by search algorithms on each island. In order to share information within this distributed search process, the individuals migrate from their initial island to another destination island at regular steps. In dynamic island models, the migration process evolves during the search according to the observed performance on the different 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. Therefore, our approach is related to the adaptive management of search operators in evolutionary algorithms. In this work, our main purpose is thus to precisely analyze dynamic migration policies. We propose a testing process that assigns gains to the algorithms applied on the islands in order to assess the adaptive ability of the migration policies, with regards to various scenarios. Instead of having one dynamic migration policy that is applied to the whole search process, as it has already been studied, we propose to associate a migration policy to each individual, which allows us to combine simultaneously different migration policies.

Keywords Adaptive operator selection · Adaptive evolutionary algorithms · Dynamic island model · Island models · Migration policies

1 Introduction

Island models (IM) have been introduced in evolutionary computation in order to avoid premature convergence in population-based algorithms when solving optimization problems (Whitley et al. 1998; Skolicki 2007). 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 (Luque and Alba 2011; Melab et al. 2005). Each sub-population evolves independently on each island and interacts periodically with other islands by means of migrations (Ruciski et al. 2010). Note that classic island models consider the same algorithm on all islands and use migration of good individuals that are duplicated from an initial island in order to disseminate such individuals of high quality among other destination islands.The impact of migration has been carefully studied already by Luque and Alba (2010) and Lässig and Sudholt (2013). Migrations may be actually used in order to reinforce the most efficient islands (Skolicki and De Jong 2005; Gustafson and Burke 2006; Araujo et al. 2009). Note that the impact of the frequency of migrations has been also studied more recently by Mambrini and Sudholt (2014). One may be interested in assessing the global convergence by evaluating two complementary aspects: (1) converge on each island (e.g., the ability to obtain the best individuals on all islands using for instance the notion of takeover time; Rudolph 2000; Luque and Alba 2010) and (2) ensure a good global diversity to avoid sub-populations from getting stuck in local optima and finally reach a global

✉ Frédéric Lardeux
frederic.lardeux@univ-angers.fr

Jorge Maturana
jorge.maturana@inf.uach.cl

Eduardo Rodriguez-Tello
ertello@tamps.cinvestav.mx

Frédéric Saubion
frederic.saubion@univ-angers.fr

[1] LERIA, University of Angers, Angers, France

[2] Universidad Austral de Chile, Valdivia, Chile

[3] CINVESTAV, Tamaulipas, Mexico, Mexico

optimum (e.g., using specific problems, difficult to handle with a single panmictic population; Lässig and Sudholt 2010).

1.1 Context

Classically, in the above-mentioned research, islands models use the same algorithm on each island and the islands differ only by their populations. Lardeux and Goëffon (2010) proposed to consider different algorithms on the islands—restricted in fact to a basic evolutionary algorithm with only one variation operator—and to define dynamic migration policies. In this approach, called Dynamic Island Models (DIM), migration probabilities are modified during the evolutionary process according to the impact of previous analogue migrations, by means of a learning process. Moreover, migrations are performed after each iteration of the algorithms on the islands. Compared to classic island models DIM has indeed been related to adaptive operator selection (AOS) techniques for evolutionary algorithms (Da Costa et al. 2008) since only one operator is used on each island. DIM should be able to identify a subset of islands that are currently the most suitable for improving individuals. It should also quickly react to changes when other operators become more efficient. Therefore, DIM has been compared to other classic operator selection mechanisms by Candan et al. (2012). Remark that this approach is also related to the Bucket Brigade Algorithms introduced in the context of classification (Wilson and Goldberg 1989). In particular the work of Dorigo (1991) handles the apportionment of credit problem with such techniques.

1.2 Contributions

The purpose of this paper is to carefully study different configurations of the DIM with dynamic migration policies, as well as their ability to adapt to changes during the search process. Such changes occur when this search process explores different areas of the search space. Therefore, the basic search operators (or heuristics) may become more or less efficient according to the current state of the search. Considering the case where each island may use its own specific search algorithm, we propose here different testing scenarios in order to simulate the evolution of the search efficiency on the islands. In such abstract scenarios, gains are associated to operators of the island in order to reflect their performances. Compared to previous testing scenarios (see for instance Thierens 2005; Da Costa et al. 2008; Candan et al. 2013), we consider here a gain matrix that takes into account all 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 complementary dependencies may occur between operators that intensify the search and operators that achieve diversification in order to escape from local optima. Similar complementarity may be observed when using local search operators that involve different neighborhoods. We consider in this work two types of scenarios: fixed and dynamic. The dynamic scenarios employed are designed to reflect different possible search processes.

Extending previous works (Lardeux and Goëffon 2010; Candan et al. 2012, 2013) where the same dynamic migration policy has been investigated, we propose here to study several possible configurations of the DIM, by considering more possible components, including learning and migration processes. Moreover, instead of having predefined migration policies, we propose to attach different policies to different individuals. This cooperative model allows us to use simultaneously several migration policies in order to benefit from their respective properties. Our study highlights that DIM is efficient for tracking interactions between islands. It also quickly react to efficiency changes during the search.

1.3 Organization of the paper

This paper is organized as follows. Section 2 presents the dynamic island model. Section 3 discusses how to measure the efficiency of migration policies. The experiments are presented in Sect. 4 and application to the NK-landscape problem is proposed Sect. 5. Finally conclusions and future research works are given in Sect. 6.

2 Dynamic island models

In this section we propose to generalize the definition of dynamic island models proposed in Lardeux and Goëffon (2010) by considering different possible options for the main components of the algorithm that manage the migration process. The purpose is to evaluate the respective advantages of the resulting possible configurations.

A Dynamic Island Model (DIM) is defined by:

- its size n that corresponds to the number of islands.
- a set of islands $\mathcal{I} = \{i_1, \ldots, i_n\}$ and a set of algorithms $\mathcal{A} = \{a_1, \ldots, a_n\}$. Each algorithm a_k is assigned to island i_k.
- a population $\mathcal{P} = \bigcup_{k \in 1,\ldots,n} p_k$. Each subpopulation p_k is a subset (in fact a multiset) 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 may change continuously according to the migrations. The subpopulation obtained by applying one iteration of the

algorithm a_k over the subpopulation p_k is referred to as $a_k(p_k)$.
- a topology given by an undirected graph $(\mathcal{I}, E)$ where $E \subseteq \mathcal{I} \times \mathcal{I}$ is a set of edges between islands (here we will consider a complete graph).
- a migration matrix M of size $n \times n$ with $M(i,j) \in [0,1]$. $M(i, j)$ corresponds to the probability of migration of an individual from island i to island j. At each iteration of the algorithm described below, each individual migrates from an initial island to a destination island according to these probabilities. M is supposed to be coherent with the topology, i.e., if $(i,j) \notin E$ then $M(i,j) = 0$, $\mathcal{M}$ is the set of migration matrices.
- a migration policy $\Pi : \mathcal{I} \times \mathcal{M} \rightarrow \mathcal{I}$ permits to select a destination island given an initial island and a migration matrix. This general notion of migration policy has to be implemented by means of different functions in a DIM algorithm.

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

Algorithm 1: Dynamic Island Model Behavior

```
Input  : a DIM, a gain function
Output: a solution s*
Local  : a reward matrix R of size n × n
1  s* ← best(P);
2  R ← 0;
3  Initialiaze(M);
4  while not stop condition do
5      for k ← 1 to n do
6          p_k ← a_k(p_k);
7          for s ∈ p_k do
8              i_l ← Migrate(i_k, M);
9              p_l ← p_l ∪ {s};
10             p_k ← p_k \ {s};
11     Reward(R, P);
12     Learning(M, R);
13     b ← best(P);
14     if b > s* then
15         s* ← b;
```

Overview of the algorithm: At each iteration of Algorithm 1, given an island i_k, the process is the following: (1) the corresponding algorithm a_k is applied to the population p_k, (2) individuals migrate from this current island to other island or possibly stay on the same island. The benefit of the migrations are then computed by the reward function and used in the learning function in order to update the migration matrix M. The contents of the following subsection is:

- Section 2.1: basic components and data structures of the algorithm, including the gain function used to evaluate the performance of the islands.
- Section 2.2: reward function (line 11) that evaluates the benefit of migrations. Two different functions are proposed.
- Section 2.3: learning function (line 12) that uses assigned rewards in order to update dynamically the migration matrix M. Different learning mechanism are presented.
- Section 2.4: different possible migration processes. Using previous functions, the *Migrate* function (line 8) implements in fact the migration policy Π as defined above.

2.1 Components of the algorithm

Note that we rather aim at testing scenarios in order to evaluate different possible settings of the DIM. Therefore, we define a notion of gain associated to each algorithm located on the islands. This gain simulates the effect of the application of the algorithm on the individuals. For instance, this gain can be the fitness improvement with regards to a classic optimization problem. Of course, this approach 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 in real problem. Nevertheless, such testing scenarios for EAs have been widely used for studying control of operators (Thierens 2005; Da Costa et al. 2008) and are indeed useful for evaluating general properties of these adaptive control mechanisms.

Definition 1 (Gain and fitness of individuals) We consider a function $gain : \mathcal{A} \times \mathbb{N} \rightarrow [0,1]$, such that $gain(a, t)$ corresponds to the improvement of an individual when processed by the algorithm a at iteration t of the DIM. Individuals may be abstracted by the sum of their successive obtained gains. For an individual $s \in p_i$ at iteration t, we define its value (fitness) as $v(s,t) = \Sigma_{\tau=1}^{t} gain(a_{s(\tau)}, \tau)$, where $s(\tau)$ is the island where s was located at iteration τ.

The fitness of an individual is related to the successive operators that have been applied to it. This generic gain function model will be refined in Sect. 3.

The value $R(i, j)$ of reward matrix R evaluates the benefit (in terms of fitness improvement) 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. Note that R is initialized with 0 values. M can be initialized with similar values for all $M(i, j)$ corresponding to equal probabilities of migration for any pair of islands.

Function best the best current individual of the whole population is selected by the function $best(\mathcal{P}) = s^*$, where $s^* = max_{s \in \mathcal{P}, t}(v(s,t))$.

Stop conditions as is usual in the literature the algorithm stops after a predefined limited number of iterations or when a certain prefixed solution cost has been found in the global population $\mathcal{P}$.

In our approach, 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.

2.2 Reward function

$R^{(t)}(i,k)$ 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 $R^{(t)}(i,k)$

$$Elitist\ Reward: R^{(t)}(i,k) = \begin{cases} \frac{1}{|B|} & \text{if } k \in B, \\ 0 & \text{otherwise,} \end{cases} \text{ with}$$

$$B = \underset{k \in \{1,\ldots,n\}}{\operatorname{argmax}} (\{v(s,t) - v(s,t-1) | s(t) = k, s(t-1) = i\})$$

where 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. Remind that $v(s, t)$ correspond to the fitness of individual s at iteration t (Definition 1).

$$Average\ Reward: R^{(t)}(i,k) = \frac{\Sigma_{s \in K} max(v(s,t) - v(s,t-1), 0)}{|K|},$$
$$\text{with } K = \{s \in p_k^{(t)} | s(t-1) = i\}$$

Note that K is the set of the individuals of the island k at iteration t that were on island i at iteration $t-1$. The fitness improvement obtained on destination island i by individuals coming from previous island k is computed by $max(v(s,t) - v(s,t-1), 0)$. Note that this improvement belongs to interval [0, 1] in order to be compatible with learning functions.

2.3 Learning function

2.3.1 Basic reinforcement learning function

The basic learning principle consists in sending more individuals to the destination islands that have previously improved individuals coming from the current island and less to the destination 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:

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

where $N^{(t)}$ is a stochastic noise vector and parameters α and β are constant values in the interval [0, 1]. 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 previously studied by Candan et al. (2012).

2.3.2 QLearning based function

In this work, we introduce a classic QLearning (see Sutton and Barto (1998) for more details) algorithm in order to update the transition matrix. Compared to the previous learning function, QLearning takes into account an estimation of the future expected value that can be obtained after a migration has been performed (i.e., examining next destination j from k by using max_j, in case of k is selected as destination from i).

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

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

2.4 Migrate function

We consider four possible migration functions:

- *Elitist migration*: individuals from island i migrate to the island j that has the highest value in row i of M, i.e., $\operatorname{argmax}_j M^{(t)}(i,j)$. Such migration policy promotes intensification of the search process toward the most efficient islands.
- *Proportional migration*: each individual s on island i migrates according to a probability on the *ith* row of $M^{(t)}$. Note that M is normalized in order to insure good probability properties (in particular the sum of the value of a given line should be equal to 1).
- *Uniform migration*: individuals from island i randomly migrate to the island j (uniform probability).
- *Upper Confidence Bound (UCB) based migration*: the selection of the next possible migration can be considered as a reinforcement learning problem itself. Therefore, we consider the destination island selection as a multi-armed bandit problem (see Cesa-Bianchi and Lugosi 2006 for instance)and we select the next migration using the following formula:

$$UCB(i,j) = M^{(t)}(i,j) + \sqrt{\frac{2\log(\sum_{1 \le j \le n} nb_i^{(t)}(k))}{nb_i^{(t)}(j)}} ls$$

where $nb_i^{(t)}(k)$ is the number of individuals that have migrated from i to k at iteration t. Note that we use here the migration matrix as an estimation of the gain that has been obtained by previous migrations from i to other islands. The island j with maximal $UCB(i, j)$ value is selected for next migration. Informally, UCB ensures a trade-off between exploitation by choosing the best current destination j (i.e., greedy choice of the best value $M^{(t)}(i,j)$) and exploration of other possible destinations j (when j has been less considered, the second term of the sum increases).

2.5 Configurations of the DIM: setting the policy

While in previous works (Lardeux and Goëffon 2010; Candan et al. 2012, 2013), the proposed DIM used a single migration policy (i.e., choice of reward, learn and migrate functions). In this work, we study different configurations of the DIM in order to highlight their respective behaviors. In particular we consider an alternative learning mechanism based on QLearning, as well as alternative migration functions. A configuration of the DIM defines indeed a migration policy.

Instead of considering policies at the algorithm level, we want to fully benefit from the collaborative model induced by the DIM and we propose to link 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, we previously observed that different policies leads to different search behavior and we propose here to check that cooperation may lead to better results.

We may first remarks that different migration functions may be used in the same DIM while its 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. 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 learning functions (see Sect. 2.3), $type_l \in \{C, Q\}$, (C)lassic or (Q)learning
- $type_r$ corresponds to the type of reward functions (see Sect. 2.2), $type_r \in \{E, P\}$, (E)litist or (P)roportional
- $type_r$ corresponds to composition of the population concerning the migration functions (see Sect. 2.4) and is a tuple $(Elit, Prop, Unif, UCB)$ with $Elit, Prop, Unif, UCB \in [0, 1]$. n fact $type_r$ can be a stochastic vector, satisfying that the sum of its components is equal to 1. Nevertheless in the following, in order to simplify the configurations, we choose to use a discrete $\{0, 1\}$ notation indicating that a migration function is used or not. These possible migration functions are then equally distributed on the individuals. For instance, (1, 1, 0, 1) means that one third of the individuals use elitist migration, one third use proportional migration, and one third UCB based migration. Of course, some ill-formed configurations are not allowed, such as (0, 0, 0, 0).

Observe that we may consider here pure policies, i.e., policies that use only one type of individuals as well as mixed policies were the migration policy is not the same for all individuals. For instance, the policy $(C, E, (0, 1, 0, 0))$ corresponds to the basic dynamic island model that has been previously studied in Candan et al. (2012, 2013).

Note that DIM migration policies have been previously compared to adaptive selection operator policies based on reinforcement learning technique, considering the adaptive operator selection problem as a multi-armed bandit problem. These previous works (Goëffon et al. 2016; Candan et al. 2012, 2013) have shown that DIM is an efficient alternative to these approaches. In this paper, since we consider gains that depend from previously visited island, classic adaptive operator selection approaches, as fully described by Maturana et al. (2012), are not efficient (as we have checked experimentally). As baseline for comparisons, we consider the following policies:

- A myopic oracle (Oracle) which knows the hidden matrix and selects, at iteration $t + 1$, $\text{argmax}_j H^{(t+1)}(i, j)$ if action a_i has been selected at iteration t.
- A uniform selection (U) that selects uniformly an action at each iteration.

3 Assessing the efficiency of the policies

In this section, we focus on assessing the ability of a DIM to dynamically select the most promising islands according to the current state of the search. We want to assess that migration policies are able to adapt to changes that may occur during the search process, which will be simulated by scenarios. In order to define these scenarios, we are particularly interested in two main aspects:

- introducing changes in the efficiency of the islands, which corresponds to the fact that different exploration and exploitation stages are usually required along a search process, involving different search operators

– taking into account dependencies between islands in order to discover possible cooperative sequences, which corresponds to the fact that, when using different operators (e.g., metaheuristics or hyperheuristics), they can be combined to achieved an efficient search (e.g., due to complementary effects). For instance, after having reached a local optimum with a given search operator, one may use another operator to escape from this local optimum (e.g., using different neighborhoods that correspond to different local move operators).

The purpose of this paper is to assess the efficiency of different dynamic migration policies using scenarios that simulate both above-mentioned aspects. Given a DIM, the efficiency of the migration policy can be estimated according to the fitness of the individuals that are generated during the search process. As mentioned before, we use here generic testing scenarios (i.e., not related to a specific optimization problem) for assessing this efficiency.

Given a DIM and a time horizon T, the efficiency of its migration policy is defined by the value obtained by its best individual $v(s^*, T)$ after T iterations (see Definition 1). In this context, an optimal policy corresponds to the best sequence of applications of algorithms $a_{i(1)}, \ldots, a_{i(T)}$ (that also corresponds to the best visiting sequence of islands $i(1), \ldots, i(T)$).

In order to assess the ability to adapt the migration policy to changes, we introduce a hidden gain matrix that defines a specific gain function.

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

The gain obtained by algorithm a_k depends on the algorithm a_j that has been previously applied to a given 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 DIM. 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 behavior in our model, H will be a dynamic in our experiments, with changing values $H^{(t)}$.

Moreover, in order to simulate the fact that search operators can be often stochastic, we will consider two different types of gains, extending Definition 2:

– fixed gain functions, where $gain(a_k, t)$ is always equal to the value $H^{(t)}(j,k)$ as mentioned in Definition 2.
– probabilistic gain functions, where $gain(a_k, t) = 1$ according to the probability defined by $H^{(t)}(j,k)$. For this reason, the values of H will always belong to interval [0, 1].

Therefore a scenario is fully defined by a sequence of hidden matrices $H^{(0)}, H^{(1)}, \ldots, H^{(T)}$.

4 Experiments

The purpose of this section is to evaluate and to analyze the respective performances of the different possible configurations of the DIM, that implement different migration policies, on various possible testing scenarios. According to our evaluation methodology described in Sect. 3, we consider both static and dynamic hidden gain matrices. The goal is to provide representative scenarios, where interaction between islands may be stable or may evolve during the search process. Dynamic scenarios are defined by alternating different matrices selected from a set of predefined fixed matrices, at different change frequencies. We use also Markov inspired scenarios in order to simulate a search process that changes progressively and reach a stable state after a given time. Remind that our main motivation is to focus on the management of the migration of individuals within the DIM, whose efficiency is assessed by scenarios that take into account the benefit of visiting the islands (and thus being processed by their corresponding algorithms).

4.1 Fixed hidden matrices

We consider three basic 10×10 gain matrices A, B and C. These matrix simulate situations with different types of dependencies between islands. Based on A, B and C, we define constant gain matrices or changing matrices. The gains are illustrated on Fig. 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 edge between some islands, migrations are possible but with a null gain $H^{(t)}(i,j)$, which means that no benefit is obtained by using the operator j after the operator i (e.g., if operators have opposite or incompatible effects).

– Matrix A has two possible gain paths of length 3 that provide a total gain of 2, namely $(1-2-4-6)$ and $(1-2-3-5)$. Nevertheless, once reaching the end of these paths , i.e., either island 6 or 5, one needs to restart again from island 1 without any gain leading to a total number of 4 migrations. Therefore, using one of

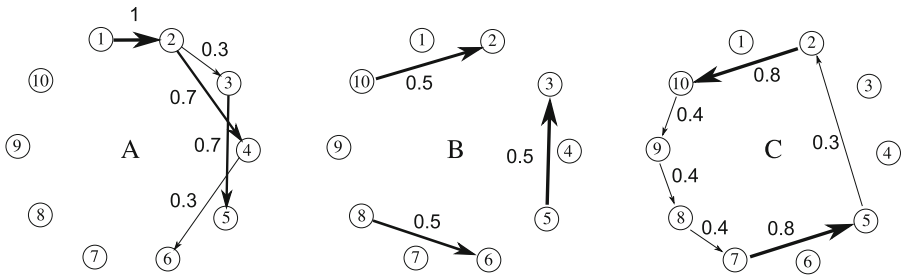

Fig. 1 Representation of possible gains of H

these paths is suboptimal compared to a cycle $(1-2-4-1)$ whose gain is thus 1.7 but in only 3 migration steps. This matrix has been defined to check if the policy is able to avoid being trapped into suboptimal paths.

- Matrix B is introduced to checked that with only 3 efficient paths, the policy is able to identify shortest gain paths (e.g., performing a cycle $(10-2-10)$).
- Matrix C has a clear gain cycle $(2-10-9-8-7-5-2)$ with no null gain transition. However this cycle is suboptimal compared to the optimal shortest cycle $(2-10-7-5-2)$.

4.2 Dynamic hidden matrices

In order to define a dynamic hidden matrix it is possible to use a sequence of fixed matrices or to use a matrix based on Markov chain transitions.

- Dynamic hidden matrix based on fixed matrices (Hidden Dyn) In this model, we alternate the three previous matrices A, B and C at defined time steps. Given a change frequency FC and a time horizon T the Hidden Dyn matrix can be defined as:

$H^{(t)} = A$ if $(FC > 1 \wedge (t \text{ div } FC) \bmod 3 = 0)$ or $(FC = 1 \wedge t \bmod 3 = 1)$
$H^{(t)} = B$ if $(FC > 1 \wedge (t \text{ div } FC) \bmod 3 = 1)$ or $(FC = 1 \wedge t \bmod 3 = 2)$
$H^{(t)} = C$ if $(FC > 1 \wedge (t \text{ div } FC) \bmod 3 = 2)$ or $(FC = 1 \wedge t \bmod 3 = 0)$

The previous three rules have been defined in order to modify the gain model using the three previously defined matrices A, B and C. Therefore, we define three possible states according to the current iteration and a frequency of change.

- Dynamic hidden matrix based on Markov chain transitions (Hidden Markov) We use a Markov chain transitions matrix (see Kemeny and Snell 1960) as dynamic hidden gain matrix.

$$HMarkov^{(0)} = \begin{bmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.3 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.5 & 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.2 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.5 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.4 & 0.5 \end{bmatrix}$$

This matrix has been defined in order to include different possible gain cycles, similar to the previous fixed matrices.

As for dynamic hidden matrix based on fixed matrices, we also use a frequency of changes FC. At each change time t, an update of the hidden matrix is performed such that $H^{(t)} = H^{(t-1)} \times H^{(0)}$ (otherwise the matrix remains identical). This Hidden Markov matrix stabilizes after 50 changes. Therefore, according to the time horizon and FC value, this stable state may be reached sooner or not.

We consider two possible types of gains: either constant gains, in this case as mentioned in Sect. 3, $gain(a_k, t) = H^{(t)}(j,k)$ if $a_{i(t-1)} = a_j$ or probabilistic gains, in this case $gain(a_k, t) = 1$ with a probability equal to $H^{(t)}(j,k)$. This allows us to simulate scenarios where the effect of an operator can be subjected to stochastic phenomena, which is specially the case in metaheuristics that use generic operators (e.g., uniform crosser or basic mutation operators in evolutionary algorithms).

4.3 Test instances and parameters of the configurations

Using the previous matrices, we define the following set of test scenarios over a time horizon $T = 600$.

- Changing dynamic hidden gain matrix with fixed gains and change frequencies 1, 5, 10, 25, 50, 100 and 200
- Markov hidden gain matrix with fixed gains and change frequencies of 1, 5, 10, 25, 50, 100 and 200

Table 1 QLearning configurations—scenarios with fixed gains—one individual

Strategies	Fixed matrices						
	A	B	C				
Q1000	165.80	64.33	328.10				
Q0100	125.20	50.53	223.42				
Q0010	19.31	9.18	20.26				
Q0001	57.31	21.65	74.64				
Strategies	Hidden dyn with different change frequencies						
	1	5	10	25	50	100	200
Q1000	102.74	47.82	41.80	46.56	67.30	125.05	163.10
Q0100	51.60	44.42	43.08	49.99	57.91	85.44	128.74
Q0010	16.80	16.49	15.92	17.25	15.90	16.90	16.63
Q0001	18.70	17.04	19.18	26.96	27.57	39.30	66.65
Strategies	Hidden Markov with different change frequencies						
	1	5	10	25	50	100	200
Q1000	146.35	158.86	167.15	185.18	198.10	222.17	256.54
Q0100	130.43	131.18	132.93	135.56	143.72	159.30	184.50
Q0010	60.79	60.70	60.47	60.64	60.41	60.36	60.20
Q0001	83.41	83.44	83.48	83.89	84.63	90.63	106.20

- Changing dynamic hidden gain matrix with probabilistic gains and change frequencies of 1, 5, 10, 25, 50, 100 and 200
- Markov hidden gain matrix with probabilistic gains and changes frequencies of 1, 5, 10, 25, 50, 100 and 200
- Fixed matrices A, B and C with fixed gains
- Fixed matrices A, B and C with probabilistic gains

We consider thus 34 scenarios for each configuration of the DIM. Each configuration is evaluated on 20 independent runs for each scenario.

Note that the performance of the learning functions presented in Sect. 2.3 can be affected by the choice of their parameter values. The literature offers different possibilities to find a combination of parameter values allowing a given algorithm to achieve its best possible performance (Birattari 2009; Hutter et al. 2009; Adenso-Diaz and Laguna 2006; Gunawan et al. 2011).

In this work, the most appropriate parameter values for the learning functions were found by using an iterated racing procedure called I/F-Race (Balaprakash et al. 2007). This offline automatic configuration procedure is implemented in R as part of the irace package (López-Ibáñez et al. 2011) and has been successfully used in several research projects (López-Ibáñez et al. 2016).

A sample of 6 representative scenarios was used to run the irace package. 1000 independent runs were executed for each scenario and for each studied learning function. The selected parameters are α and β for the basic reinforcement learning function, and δ and γ for QLearning. Each of these four parameters can take values in the range [0.0, 1.0]. As a result, the following parameter configurations were obtained: $\alpha = 0.8$ and $\beta = 0.01$ for the basic reinforcement learning function, and for QLearning, $\delta = 0.1$ and $\gamma = 0.8$. Therefore, these parameter configurations are used in the experimentations reported hereafter.

4.4 Statistical testing methodology

A statistical significance analysis was performed for the experiments presented in this paper. Each analysis was conducted using the following methodology. First, the *Shapiro-Wilk* test was used to evaluate the normality of data distributions. For normally distributed data, either *ANOVA* or the *Welch's t* parametric tests were used depending on whether the variances across the samples were homogeneous (*homoskedasticity*) or not. This was investigated using the *Bartlett's* test. For non-normal data, the nonparametric *Kruskal–Wallis* test was adopted. A significance level of 0.05 has been considered.

4.5 Experimental results

The experimental results are presented according to the previously described scenarios on the different configurations/policies of the DIM, with regards to our taxonomy

Table 2 $Q1000$—scenarios with fixed gains—different population sizes

nb ind	Fixed matrices						
	A	B	C				
1	165.80	64.33	328.10				
4	266.48	71.73	345.24				
10	285.13	74.39	353.65				
20	292.99	76.37	353.81				
50	295.43	77.30	360.81				
100	297.00	78.96	365.94				
400	297.76	79.95	390.00				
nb ind	Hidden dyn with different change frequencies						
	1	5	10	25	50	100	200
1	146.35	158.86	167.15	185.18	198.10	222.17	256.54
4	193.23	211.27	215.45	224.48	235.66	258.63	292.62
10	223.17	233.99	236.93	240.27	254.33	284.98	327.23
20	241.84	247.34	250.43	251.38	263.74	296.93	339.57
50	250.05	251.92	252.54	253.96	269.89	310.78	364.17
100	253.14	253.50	254.13	256.44	270.75	310.77	362.44
400	255.65	256.03	256.50	257.92	274.71	327.93	395.20
nb ind	Hidden Markov with different change frequencies						
	1	5	10	25	50	100	200
1	102.74	47.82	41.80	46.56	67.30	125.05	163.10
4	136.01	90.15	74.12	97.93	153.16	180.51	209.37
10	144.88	105.75	100.74	125.75	176.00	200.90	232.96
20	146.01	134.55	113.53	131.48	190.27	218.73	243.21
50	134.32	135.21	121.66	132.30	195.63	223.57	245.90
100	130.66	147.71	127.72	134.09	196.28	226.87	248.31
400	137.92	149.15	127.07	132.08	175.98	225.61	250.79

(see Sect. 2.5). In the tables, the final gains are mentioned (i.e., the sum of all successive gains obtained by the best individual according to our previous definitions). Therefore, higher values correspond to better performances.

4.5.1 Configurations based on QLearning

In this paper, we introduce QLearning as a possible learning technique in order to update the migration matrix M as defined in Sect. 2.3. Therefore, we first study this new configuration of the DIM before going on into more complete comparisons.

According to Sect. 2.5, this learning approach can be combined with two possible reward functions providing a set of possible configurations of the form $QE*$ or $QP*$, where $*$ is any vector $\{0,1\}^4$. We have run the previously defined scenarios and we observed that both configurations provide similar results for each evaluated scenario (there was not a significant performance difference between them). Therefore, in the following, we only consider one type of configuration, which will be called $Q*$, with the different values of $*$ as above.

Concerning the QLearning $Q*$ configurations, we aim first at evaluating different configurations with only one individual. To this aim, we run single individual versions of the following four configurations $Q1000$, $Q0100$, $Q0010$, $Q0001$. Each configuration has been run 20×20 times for each scenario and the best results of each sequence of 20 consecutive runs has been used to compute the mean value. In this case, we compare the mean value with the results obtained with other configurations with

20 individuals. Table 1 shows that the configuration $Q1000$ provides the best results, followed by $Q0100$, which means that for QLearning, as expected, a greedy based strategy is useful for selecting next island.

Once having compared the respective performances of the different "pure" configurations based on QLearning, we aim at studying the influence of using several individuals with different migration policies, instead of a single one, i.e. assessing the benefit of using a cooperative management of the migration process and its influence on the knowledge that is collected by a population of individuals. We focus thus on the configuration $Q1000$ and run several experiments with different population sizes. Note that for all the experiments, the same computation power (total number of executions) is assigned to each configuration. For instance, when one individual is used, 400 executions are needed to compute the mean values, as mentioned above. The number of individuals ranges from 1 to 400 for the different scenarios with fixed gains.

The results in Table 2 show that the performance increases with the number of individuals. This is interesting, since from the QLearning point of view, it means that using a DIM with several individuals, which share their learned information by means of the migration matrix M, may improve the global learning process and, consequently, improves the management of the migrations. It also appears that, when using more than 20 individuals, improvement slows down. Therefore, we consider that using 20 individuals for the next experiments constitutes a good compromise.

In this section, we have restricted the presentation of our experiments to fixed gain scenarios, since results obtained with probabilistic scenarios are rather similar. We have only assessed the benefit of using several individuals instead of a single one for improving the performance of configurations that use only one type of migration policy. In next section, we will consider general configurations using several type of individuals—and thus several migration policies—simultaneously.

Table 3 Basic fixed matrix (A–B–C)—fixed gain—top 12 configurations

Fixed matrices					
A		B		C	
Q1001	(44)	Q1000	(40)	Oracle	(46)
Q1101		Oracle	(39)	CP1001	(43)
Q1000	(43)	Q1110	(33)	Q1001	
Q1011	(41)	Q1100		Q1010	(37)
Q1100		CP1111		Q1100	
Q1111	(40)	Q1010	(32)	Q1101	(36)
Q1110	(38)	Q1011	(31)	Q1011	
Q0101		Q0110	(30)	Q1110	
Q0100	(36)	Q0100	(29)	Q1111	
Q1011		C P 0011		C P1011	(30)
Q1101	(35)	Q0111	(27)	CE1101	(28)
Q1111		CP 0111	(26)	CE1111	(27)

4.5.2 Comparison of the different configurations

According to previous observations made in Sect. 4.5.1, we consider 15 configurations for QLearning (different combinations types, ignoring the reward function as already mentioned), 30 configurations for the classic learning mechanism C associated to all possible rewards and combinations of migration functions, the uniform blind migration policy (U) and the myopic oracle (Oracle). Hence, 47 configurations are evaluated on 34 scenarios. A statistical significance analysis was performed for each possible pair of configurations, say (x, y), by using the methodology described in Sect. 4.4. Then, a *global score* of the configurations was computed as follows. If a significant performance difference exists between configurations x and y for a particular scenario, the global score of either x or y is increased by one point depending on whether such a difference favors one of these configurations. If there is no significant difference between the compared configurations their global scores remain unaltered. Thus, the higher the global score of a configuration, the better its overall performance is. In order to provide a more readable analysis of the results, we decided to focus on the 12 best performing configurations according to its score (the first quartile).

In the rest of the tables presented in this section (Tables 3, 4, 5, 6, 7 and 8), the data presented in each column is sorted in descending order according to the *global score* (shown between parentheses) obtained by the configurations in the statistical significance analysis. Configurations with a similar *global score* value are grouped. In Tables 4, 5, 7 and 8 the frequency at which the hidden matrix is changed is represented by FC.

We first analyze the results for Fixed, Hidden Dyn and Hidden Markov matrices using fixed gains.

Table 3 shows that depending on the structure of the hidden matrix that may contain improvement cycles or not as well as suboptimal gain cycles, the performance of the policies may differ. In particular, it is interesting to remark that the Oracle may be confused with matrix A, where there exist two possible gain paths (no cycle), one being suboptimal. $Q*$ policies are more efficient to detect such gain cycles and to identify the best ones. Since on matrix B no cycle occurs, classic $C*$ policies can be efficient to detect short gain paths or cycle paths, as those present in matrix C. Of course it is noticeable that $Q*$ policies provide good

Table 4 Hidden dyn—fixed reward—top 12 configurations

Frequency of changes (FC)													
1		5		10		25		50		100		200	
Q1100	(45)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Q1000	(42)	Q1000	(45)
Q0101	(38)	Q1100	(41)	Q0101	39	Q1010	(45)	Q1110	(45)	Q1100		Q1100	
Oracle	(37)	Q1101		Q1001	(38)	Q1110	(40)	Q1010	(40)	Q1101		Q0100	(42)
Q1000	(33)	Q0100	(39)	CP1010	(33)	Q0110		Q1011		Q0100	(41)	Q1110	
Q1101		CP1001	(38)	Q1101		Q0111		Q1111		Q1011	(40)	Q1101	(41)
Q0100	(31)	Q0101		Q1000	(32)	Q1011		Q1000		Q1111	(39)	Q1001	(40)
Q1111		Q1000	(37)	Q0110	(30)	Q1111		Q0100		Oracle	(38)	Q1111	
Q1001	(30)	CP0001	(35)	Q1010		Q1000	(36)	Q0110	(36)	Q1110		Q1011	(39)
CP1010		Q1001	(34)	Q1100	(29)	Q0101	(33)	Q1100	(33)	Q1010	(37)	Q0110	(35)
CP1011		Q1110	(33)	Q1011		Q1100		Q1101		CP0111	(33)	Q0101	
CP1111		Q1111		Q1110		Q0011	(31)	CP0111	(31)	Q0111	(32)	Q0111	

Table 5 Hidden Markov—fixed gains—top 12 configurations

Frequency of changes (FC)													
1		5		10		25		50		100		200	
Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)
Q1101	(38)	CP1001	(37)	Q1001	(40)	Q1001	(39)	Q1101	(34)	Q1101	(36)	Q1110	(29)
CE1001		Q1001		CE1001	(39)	CP1001	(38)	Q1100	(33)	Q1100	(35)	CP1011	
Q1001		CE1001		CP1001	(36)	CE1001		Q1001		Q1110	(34)	Q1101	(28)
CP1001		Q1101	(36)	Q1101		Q1101	(35)	Q1010	(31)	Q1011	(33)	Q1100	(27)
CE1011	(35)	Q1011	(35)	CE1011	(33)	Q1011	(34)	Q1011		Q1010	(32)	Q1001	
Q1011		CE1011	(32)	Q1011		CE1011	(30)	CP1001		Q1001		Q1011	
CE1101	(34)	Q1100		Q1100	(32)	CE1101		Q1110		Q1111		Q1111	
Q1100	(32)	CE1101		CE1101	(30)	Q1100		Q1111		CE1111	(26)	CP1111	
Q1110		Q1111		Q1111		Q1111		CE1101	(30)	CP1011		Q1010	(26)
Q1111		CE1111	(30)	Q1110		Q1110	(29)	Q1000	(29)	CE1011	(25)	Q0011	
Q1010	(31)	Q1110		Q1010	(29)	Q1010	(28)	CE1001	(27)	CE0011		CE0111	(25)

performance as soon as they use either elitist or proportional migration to promote greedy choices.

We consider now the dynamic hidden matrices based on alternating between A, B and C, again with fixed gains. These matrices will be called Hidden Dyn in the remaining of this paper. We may observe in Table 4 that, if the changes are too frequent then, since the learning process is unable to detect any stable knowledge, most of the policies are statistically equivalent. Nevertheless, note that, if we considered more than the 12 best policies it will not be still the case. Since the Oracle was rather efficient for the fixed matrices case, it still have good properties. In most of these experiments, classic learning processes (either CP or CE) exhibit good results. It is also interesting to mention that, in these experiments, configurations $Q*$ that use several types of individuals obtain better results than configurations using only one type of individual.

In Table 5, results are more contrasted among the different policies. When the frequency of changes if high (i.e., lowest values), then the matrix stabilizes sooner (remind that the matrix is stable after 50 changes). Therefore, methods that were efficient for fixed matrices are also efficient here (e.g., $CP*$ or $Q*$). It is interesting to see that when the changes are less frequent, due to the structure of the initial matrix, which does not have specific gain paths or cycles, $Q*$ are really efficient and may become even better than the Oracle. This phenomena is certainly related to the successive states reached by the matrix during its stabilization process.

Table 6 Basic fixed matrix (*A*–*B*–*C*)—probabilistic gain—top 12 configurations

Fixed matrices					
A		B		C	
Q1000	(41)	Oracle	(39)	Oracle	(46)
Q1001		Q1000		CP1001	(37)
Q1011	(39)	Q1110	(35)	Q1001	(36)
Q1100		Q1100	(34)	Q1010	(35)
Q1101		Q0100	(30)	Q1100	
Q1111		Q0110		Q1101	(33)
Q1110	(37)	Q1010		Q1011	(32)
Q0100	(36)	CP0011	(28)	Q1110	
Q0101		CP0100		Q1111	
Q1010		CP1111		CP1111	(28)
Q0110	(35)	Q1011		CE1111	(26)
Q0111		CP0111	(24)	CE0011	(25)

The following Tables 6, 7 and 8 present the results obtained with probabilistic gains. These results are quite similar for many policies. Nevertheless, when using probabilistic gains, $Q*$ policies have better properties than $C*$ policies.

In order to give a clearer overview of our experiments, we present in Figs. 2 and 3, a global comparison of the methods that have been ranked in the top 12 for all previous experiments with respect to their global score, distinguishing between fixed and probabilistic gains. The y-axis represents the sum of the scores that have been obtained by the policies according to their rankings (higher scores correspond to better performances). We may draw the following conclusions from these experiments:

- The newly introduced $Q*$ policies provide good results allowing to reach even better adaptive control than a myopic Oracle. In presence of fixed gains, classic configurations $C*$ are still interesting. Note that all the best configurations use individuals that migrate according to an elitist choice (i.e. greedy choice of the best possible next decision). When this migration component is not present, it can be compensated by a proportional migration that allows the individuals to benefit from a less elitist migration process. The *UCB* migration policy mixing exploitation (greedy choice) and exploration (random choice) seems to be also an interesting diversification mechanism, if combined with more exploitative strategies, while not being efficient if used alone (as seen in Sect. 4.5.1).
- Using simultaneously several types of individuals is really beneficial. In fact, some individuals play the role of explorers: even if they obtain poor gains, they contribute to the global learning process, by updating the reward matrix and, consequently, the migration matrix. Individuals that focus on elitism can be considered as "champions", whose role is to collect the best possible scores, according to the current knowledge with regards to estimated expected gains. Therefore, instead of achieving the classic trade-off between exploration and exploitation at the algorithm level, it is performed by means of the cooperation between individuals. In particular, let us note that $C*$ configurations using several types of individuals achieve better results than the classic initial DIM (i.e., *CE*0100). In general, good results can be obtained

Table 7 Hidden dyn—probabilistic gains—top 12 configurations

Frequency of changes (FC)													
1		5		10		25		50		100		200	
Q0101	(38)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Oracle	(46)	Q1000	(42)	Q1000	(45)
Oracle	(37)	CP1001	(42)	Q0101	(40)	Q1010	(44)	Q1010	(42)	Q1100	(41)	Q0100	(43)
Q0100		Q1100	(41)	Q1001		Q1110		Q1011		Q0100	(40)	Q1100	
Q1000		Q0100	(40)	CP1010	(32)	Q0110	(40)	Q1110		Q1101		Q1110	(42)
Q1001	(36)	Q1101	(38)	Q1000		Q0111		Q1111		Q1110		Q1101	(41)
Q1101		CP1010	(36)	Q1101		Q1011		Q1000	(38)	Q1111		Q1001	(40)
Q1100	(34)	Q0101		Q0100	(31)	Q1111		Q0100	(37)	Oracle	(38)	Q1111	
CP1001	(30)	Q1001		Q1100		Q1000	(38)	Q0110		Q1011		Q1010	(38)
Q1110		CP0001	(35)	Q0110	(30)	Q1101	(36)	Q0111		Q1010	(36)	Q1011	
Q1111		CP1011	(34)	Q1010		Q0011	(33)	Q1100		CP0011	(32)	Q0110	(37)
CP1101	(29)	Q1000		Q1011		Q0100		Q1101	(35)	Q0101		Q0101	(35)
CP1010	(27)	CP0011	(32)	Q1110		Q0101		CP0111	(32)	Q0111		Q0111	

Table 8 Hidden Markov—probabilistic gains—top 12 configurations

Frequency of changes (FC)													
1		5		10		25		50		100		200	
Q1101	(39)	Oracle	(37)	Oracle	(39)	CE1011	(35)	Oracle	(43)	Oracle	(45)	Oracle	(46)
Oracle	(38)	CE1011	(36)	CE1011	(34)	Oracle		Q1100	(40)	Q1100	(42)	Q1100	(38)
CE1101	(36)	Q1101	(35)	CE1101		Q1101		Q1000	(33)	Q1010	(38)	Q1001	(37)
CE1001	(34)	CP1001	(34)	CP1001		CP1001	(34)	Q1010		Q1101		Q1010	
CE1011		Q1100	(33)	Q1101		CE1001	(33)	Q1101		Q1001	(35)	Q1101	(34)
Q1001		Q1011		CE1001	(33)	CE1101		Q1011	(32)	Q1011		Q1011	(32)
Q1011	(32)	Q1010		Q1001		Q1001		CE1011	(29)	Q1110		Q1110	(31)
CP1001	(31)	Q1001		Q1010		Q1011		CP1001	(27)	Q1111	(28)	Q1111	(30)
CP1101		CE1101		Q1011		Q1100		Q1001		CE1011	(26)	Q0011	(24)
Q1010		CE1001		Q1100	(32)	Q1010	(32)	Q1110		Q1000	(25)	CP0011	(23)
Q1100		Q1111	(30)	CP1011	(31)	CP1011	(30)	CE1101	(26)	CE0011	(21)	CE0111	(18)
CP1011	(30)	CP1101		CP1101		Q1000		CE0011	(23)	CE0111	(20)	Q1000	

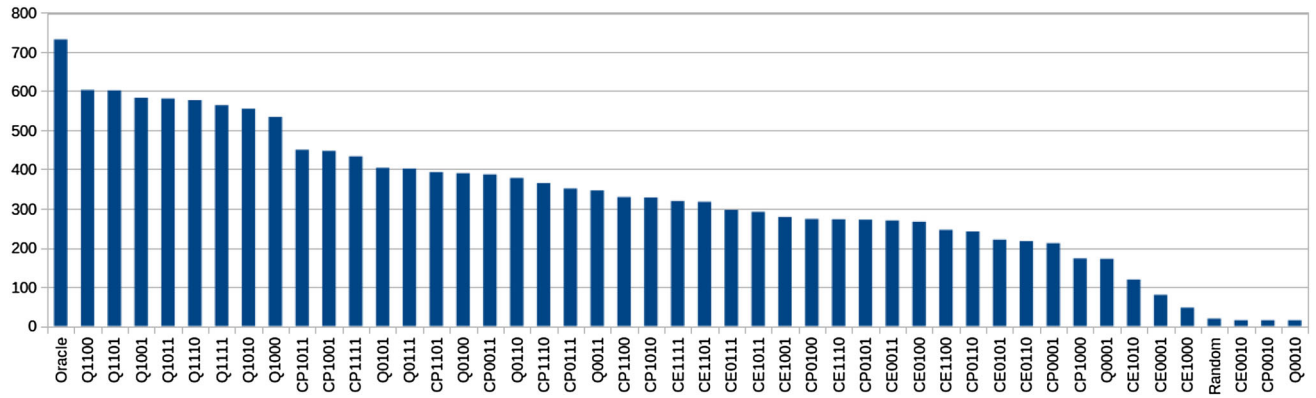

Fig. 2 Ranking of configurations for fixed gains based on their global score computed from data obtained from a statistical significance analysis (see Sect. 4.4)

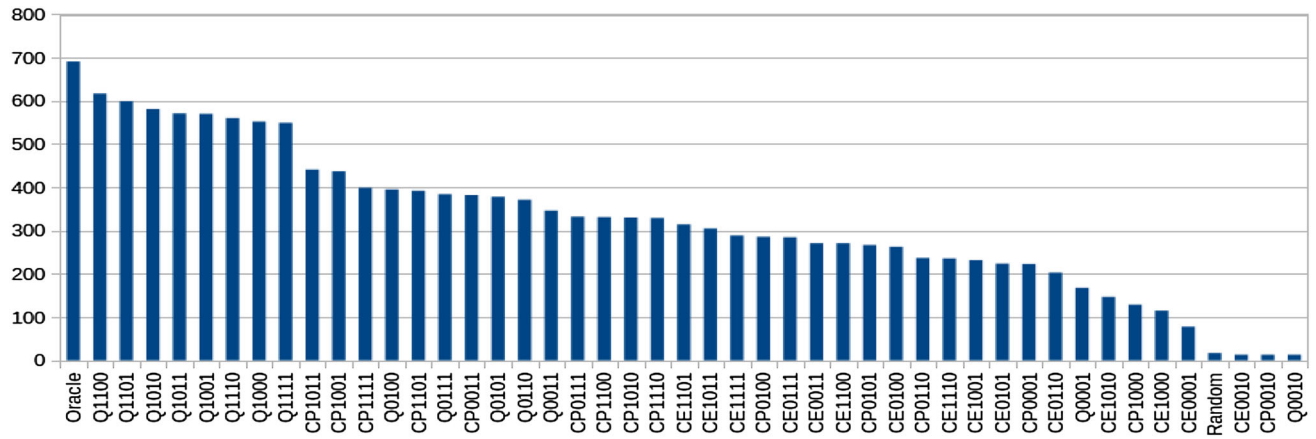

Fig. 3 Ranking of configurations for probabilistic gains according to their global score computed from data obtained from a statistical significance analysis (see Sect. 4.4)

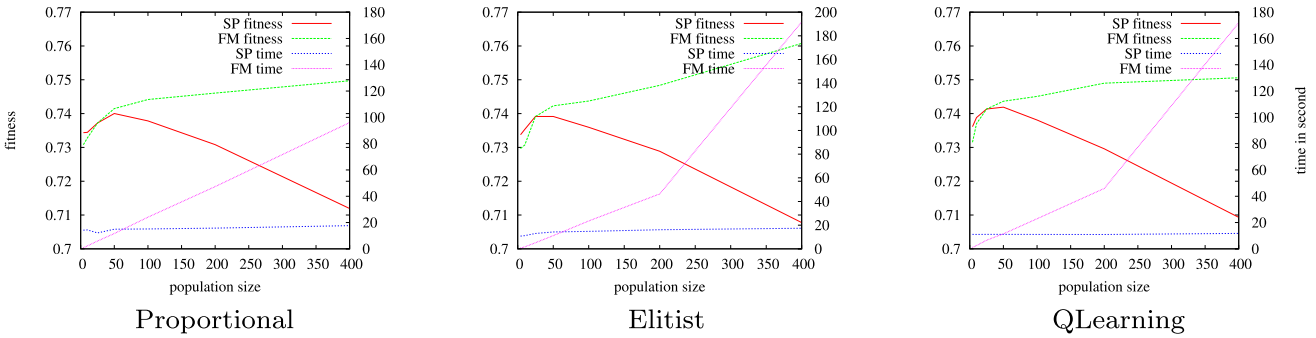

Fig. 4 Impact of population size on fitness and computation time

when individuals using an elitist migration policy are associated to individuals using a policy that permits more or less diversification (from uniformly random choice to proportional choice).

– Concerning QLearning based configurations, remark that, of course, this learning process is really interesting for controlling migrations in DIM, which had not been experimented before. Moreover, it is interesting to remark that using simultaneously different type of individuals provide an increase of efficiency of the learning process compared to a pure strategy (e.g., $Q1000$). In particular, it should be noticed that the configuration $Q1100$, that mixes elitist and proportional migration, always belong to the 12 best configurations for the 34 scenarios.

5 Application to the NK-landscape problem

In this section, we propose to observe the behavior of migration policies on a test proble that is often used in evolutionary computation.

5.1 Description of the problem

The Nk-landscape problem (Kauffman 1993) 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).

5.2 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 Sect. 5.3. The parameters of the Learn functions have been set using as in previous section. In this section, our purpose is to study experimentally the following properties of DIM:

– Impact of the population size: since we introduce different types of individuals, each one defining its own migration policy and collaborating through the learning mechanism, we propose to evaluate the influence of the number of individuals on the quality of the learning process.
– 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 with respect to other learning techniques.

In previous section, we have observed that several migration functions have indeed related effects. In particular, UCB that can be viewed as an mix between elitist choice and random choice. Nevertheless, observing results in Tables 7 and 8, we may see that the effects of UCB are not really significant when combined with other functions and is inefficient when used alone. Therefore, in this section in order to propose a more compact study, we reduce the possible migrations functions to: Elitist, Proportional and Uniform. The configurations are thus defined by triple.

5.3 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

Table 9 Comparison of different configurations

Population	Learn and reward functions							
Elit, *Prop*, *Unif*	CE		CP		Q		Rand	
1, 0, 0	5	(3)	6	(2)	7	(1)	0	(4)
0, 1, 0	8	(1)	5	(3)	8	(1)	0	(4)
1, 1, 0	9	(1)	7	(2)	5	(3)	0	(4)
1, 0, 1	8	(1)	6	(2)	6	(2)	0	(4)
0, 1, 1	6	(2)	6	(2)	7	(1)	0	(4)
1, 1, 1	6	(2)	7	(1)	6	(2)	0	(4)
Average sum	7	(1)	6.17	(3)	6.5	(2)	0	(4)
Average rank	1.67		2		1.67		4	

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 4 presents the results on the 128_8.0 nk-landscape instance and configurations CP, CE and Q, where individuals are equally dispatched between the different migration functions, i.e. (1, 1, 1) configurations. 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 appears that a population size around 20–25 individuals constitutes a good compromise.

5.4 Evaluating different policies

Table 9 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 10,000. These experiments use a population of 25 individuals with different configurations of migration functions.

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. 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.

We observe here results that have some common points with results obtained for fixed matrices with fixed gain in Table 3. Here, classic learning methods (CP and CE) have interesting results.

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.

6 Conclusions

In this work, we propose to study the management of migrations in islands models. In Dynamic Island Models, the migration mechanism is updated during the search in order to better adapt to the current state of the search process and therefore to improve its overall performance. DIM are using different sub-populations that are processed by different search operators or algorithms. In order to provide a wide range of possible search situations, we propose to use search scenarios that take into account possible interactions between the operators as well as the dynamic evolution of their efficiency during the search. Based on a generic DIM, we propose a rather complete set of possible components that allows us to define different configuration of the DIM, involving different management of the individuals and of the migration processes. Instead of having one dynamic migration policy that is applied to the whole search process, as it has already been studied, we propose here to associate this policy to each individual, which allows us to use simultaneously different migration policies within the same DIM. Therefore, these individuals cooperate and share their information into a common migration matrix. We also consider in this work, 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 scenarios. Our results highlights 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.

References

Adenso-Diaz B, Laguna M (2006) Fine-tuning of algorithms using fractional experimental designs and local search. Oper Res 54(1):99–114

Araujo L, Merelo JJ, Mora A, Cotta C (2009) Genotypic differences and migration policies in an island model. In: Proceedings of the 11th annual conference on genetic and evolutionary computation, GECCO, ACM, Montreal, Québec, Canada, pp 1331–1338. https://doi.org/10.1145/1569901.1570080

Balaprakash P, Birattari M, Stützle T (2007) Improvement strategies for the F-race algorithm: sampling design and iterative refinement. Springer, Berlin, pp 108–122. https://doi.org/10.1007/978-3-540-75514-2_9

Birattari M (2009) Tuning metaheuristics: a machine learning perspective. Springer, Berlin. https://doi.org/10.1007/978-3-642-00483-4

Candan C, Goëffon A, Lardeux F, Saubion F (2012) A dynamic island model for adaptive operator selection. In: Proceedings of the 14th annual conference on genetic and evolutionary computation, GECCO, Philadelphia, Pennsylvania, USA, pp 1253–1260. https://doi.org/10.1145/2330163.2330337

Candan C, Goëffon A, Lardeux F, Saubion F (2013) Non stationary operator selection with island models. In: Proceedings of the 15th annual conference on genetic and evolutionary computation, GECCO, Amsterdam, The Netherlands, pp 1509–1516. https://doi.org/10.1145/2463372.2463559

Cesa-Bianchi N, Lugosi G (2006) Prediction, learning, and games. Cambridge University Press, New York

Da Costa L, Fialho A, Schoenauer M, Sebag M (2008) Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the 10th annual conference on genetic and evolutionary computation, GECCO, ACM, Atlanta, GA, USA, pp 913–920. https://doi.org/10.1145/1389095.1389272

Dorigo M (1991) Message-based bucket brigade: an algorithm for the apportionment of credit problem. Springer, Berlin, pp 235–244. https://doi.org/10.1007/BFb0017018

Goëffon A, Lardeux F, Saubion F (2016) Simulating non-stationary operators in search algorithms. Appl Soft Comput 38:257–268. https://doi.org/10.1016/j.asoc.2015.09.024

Gunawan A, Lau H, Lindawati L (2011) Fine-tuning algorithm parameters using the design of experiments approach. Springer, Berlin, pp 278–292. https://doi.org/10.1007/978-3-642-25566-3

Gustafson S, Burke E (2006) The speciating island model: an alternative parallel evolutionary algorithm. J Parallel Distrib Comput 66(8):1025–1036. https://doi.org/10.1016/j.jpdc.2006.04.017

Hutter F, Hoos HH, Leyton-Brown K, Stützle T (2009) Paramils: an automatic algorithm configuration framework. J Artif Intell Res 36(1):267–306

Kauffman S (1993) The origins of order: self-organization and selection in evolution, 1st edn. Oxford University Press, Oxford

Kemeny JG, Snell JL (1960) Finite Markov chains. D. Van Nostrand, Princeton

Lardeux F, Goëffon A (2010) A dynamic island-based genetic algorithms framework. In: Proceedings of the Asia–Pacific conference on simulated evolution and learning. Springer, lecture notes in computer science, vol 6457, pp 156–165. https://doi.org/10.1007/978-3-642-17298-4

Lässig J, Sudholt D (2013) Design and analysis of migration in parallel evolutionary algorithms. Soft Comput 17(7):1121–1144. https://doi.org/10.1007/s00500-013-0991-0

Lässig J, Sudholt D (2010) The benefit of migration in parallel evolutionary algorithms. In: Proceedings of the 12th annual conference on genetic and evolutionary computation, GECCO, ACM, pp 1105–1112. https://doi.org/10.1145/1830483.1830687

López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, Stützle T, Birattari M (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002

López-Ibáñez M, Dubois-Lacoste J, Stützle T, Birattari M (2011) The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium. http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf

Luque G, Alba E (2011) Parallel genetic algorithms, theory and real world applications, studies in computational intelligence, vol 367. Springer, Berlin. https://doi.org/10.1007/978-3-642-22084-5

Luque G, Alba E (2010) Selection pressure and takeover time of distributed evolutionary algorithms. In: Proceedings of the 12th annual conference on genetic and evolutionary computation, GECCO, ACM, Portland, Oregon, USA, pp 1083–1088, https://doi.org/10.1145/1830483.1830684

Mambrini A, Sudholt D (2014) Design and analysis of adaptive migration intervals in parallel evolutionary algorithms. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation, GECCO, ACM, Vancouver, BC, Canada, pp 1047–1054. https://doi.org/10.1145/2576768.2598347

Maturana J, Fialho A, Saubion F, Schoenauer M, Lardeux F, Sebag M (2012) Autonomous search. In: Adaptive operator selection and management in evolutionary algorithms. Springer, pp 161–190. https://doi.org/10.1007/978-3-642-21434-9_7

Melab N, Mezmaz M, Talbi E (2005) Parallel hybrid multi-objective island model in peer-to-peer environment. In: Proceedings of the 19th IEEE international parallel and distributed processing symposium, IEEE, p 190, https://doi.org/10.1109/IPDPS.2005.327

Ruciski M, Izzo D, Biscani F (2010) On the impact of the migration topology on the island model. Parallel Comput 36(10–11):555–571. https://doi.org/10.1016/j.parco.2010.04.002

Rudolph G (2000) Takeover times and probabilities of non-generational selection rules. In: Proceedings of the 2nd annual conference on genetic and evolutionary computation, GECCO, Morgan Kaufmann, Las Vegas, Nevada, USA, pp 903–910

Skolicki ZM (2007) An analysis of island models in evolutionary computation. Ph.D. thesis, George Mason University, Fairfax

Skolicki Z, De Jong K (2005) The influence of migration sizes and intervals on island models. In: Proceedings of the 7th annual conference on genetic and evolutionary computation, GECCO, Washington DC, USA, pp 1295–1302, https://doi.org/10.1145/1068009.1068219

Sutton R, Barto A (1998) Reinforcement learning: an introduction. MIT Press, London

Thierens D (2005) An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th annual genetic and evolutionary computation conference, GECCO, ACM, Washington DC, USA, pp 1539–1546. https://doi.org/10.1145/1068009.1068251

Whitley D, Rana S, Heckendorn RB (1998) The island model genetic algorithm: on separability, population size and convergence. J Comput Inf Technol 7(1):33–47 http://cit.fer.hr/index.php/CIT/article/view/2919/1783

Wilson SW, Goldberg DE (1989) A critical review of classifier systems. In: Proceedings of the 3rd international conference on genetic algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 244–255. http://dl.acm.org/citation.cfm?id=645512.657260