Modèles en Iles Dynamiques

Adrien Goëffon et Frédéric Lardeux

LERIA - Université d'Angers

{prenom.nom}@univ-angers.fr

Résumé

Dans ce papier, nous proposons un cadre de travail pour les modèles en iles dont les topologies sont représentées par des graphes complets étiquetés. À chaque arc est associée une probabilité de migration entre deux iles. Nous montrons ici comment faire évoluer ces probabilités dynamiquement au cours de la recherche en fonction de l'impact des dernières migrations, ce qui permet d'améliorer le paramétrage et l'efficacité des algorithmes génétiques et évolutionnaires. Deux types d'application de ce cadre de travail sont détaillés ici : des modèles en iles auto-adaptatifs et des algorithmes évolutionnaires autonomes.

Abstract

In this paper, we present a framework for island-based evolutionary algorithms, using a complete graph modeling. Each directed edge is labeled by a migration probability between two islands. We show that probabilities can evolve dynamically during the search with respect to the last migrations impact, in order to improve the efficiency of evolutionary algorithms and to make their tuning easier. Two applications of this framework are detailed here : auto-adaptive island models and autonomous evolutionary algorithms.

1 Introduction

Les algorithmes génétiques (AG) [10] sont largement utilisés pour la résolution approchée de problème NP-difficiles. Ils sont relativement simples à concevoir et peuvent produire sur de nombreux problèmes d'optimisation de très bons résultats en terme de qualité des solutions et de robustesse de l'algorithme. Néanmoins, leur efficacité dépend d'un certain nombre de facteurs, de la représentation des configurations [18] à la fonction de fitness utilisée [29], en passant par les opérateurs de mutation et de croisement définis [21] ou encore des paramétrages plus généraux (taille de la population, fréquence des mutations, contrôle de la diversité, sélections des individus, élitisme, *etc.*) [12, 11, 25]. Même avec un effort particulier pour adapter un AG efficace à un problème particulier, on observe rapidement sur certaines instances plus critiques ou difficiles un certain nombre de limitations en terme de performances générales ou de passage à l'échelle de l'algorithme.

Afin de rendre les AG plus efficaces et puissants, les techniques habituelles consistent à les hybrider avec des algorithmes de recherche locale (algorithmes mémétiques [19]), et/ou d'utiliser des modèles en ile permettant à la fois de combiner plus efficacement l'intensification et la diversification de la recherche, mais aussi de rendre l'algorithme plus facilement parallélisable [28].

Depuis une vingtaine d'années et l'avènement des premiers algorithmes évolutionnaires distribués [26], les modèles en ile sont de plus en plus étudiés dans la communauté évolutionnaire. La principale problématique est de définir à la fois la topologie du modèle (*où peut migrer tel individu ?*) et les politiques de migration (*quel individu doit migrer ?*) de manière à ralentir la convergence générale de la population tout en favorisant le brassage global des bons individus. Araujo *et al.* ont dressé dans [2] un solide état de l'art des modèles en ile, concernant notamment la question des politiques de migration. On peut observer qu'un nombre important de topologies (on retrouve dans [14, 22] de nombreux types de modèles comme les chaines, anneaux, hypercubes et bien d'autres) et de politiques [3, 6, 27, 5, 7, 1] ont été définies, grandement basées sur des mesures de diversité locales et globales (par population / sur l'ensemble des iles). Dans tous les cas, la fréquence et la quantité des migrations demeure difficile à déterminer [24], d'autant plus que les interactions entre les deux niveaux d'évolution (intra- et inter-iles) sont très importantes [23]. Idéalement, une stratégie d'évolution globale intelligente devrait tirer un avantage de ces interactions et maximiser le bénéfice des migrations. Mais à partir de simples indicateurs intra- et inter-iles (typiquement des mesures de diversité et de fitness), il est difficile de prédire à quel moment des individus doivent migrer, lesquels, où

et pour quel impact.

Ces considérations nous ont motivé à développer un modèle en ile dynamique dont le but est d'auto-adapter sa topologie et ses politiques de migration au cours de la recherche en fonction d'indicateurs définis en paramètre. Cela peut être les propriétés des sous-populations, mais surtout et dans le cadre d'une approche originale et plus efficace, les effets des précédentes migrations. La première application est donc de définir dynamiquement une politique optimale de croisements entre individus dans les AG pour une meilleure efficacité. Dans un second temps, nous étendons le champ d'application de notre modèle dynamique, non plus pour optimiser les croisements entre individus, mais pour sélectionner automatiquement, dans le cadre d'un algorithme évolutionnaire, les opérateurs de recherche locale les plus pertinents au cours de la recherche. Dans cette extension, les iles ne constituent plus qu'un simple partitionnement entre les individus, mais disposent chacunes de leur propres règles et paramétrages, par exemple un opérateur de recherche locale différent.

Ce papier est structuré en trois parties. Dans la section 2, nous décrivons de manière générale le modèle en ile dynamique, en le présentant comme un cadre de travail. La section 3 se focalise sur l'application principale d'un tel modèle, à savoir la modification dynamique de sa topologie [16]. Des résultats sur les problèmes 0/1KP et MAX-SAT montreront le bien-fondé de cette approche. Dans la section 4, nous montrons comment une telle modélisation peut être utilisée pour définir un algorithme évolutionnaire autonome, choisissant par exemple les opérateurs de recherche locale les plus pertinents au cours de la recherche.

2 Généralisation des modèles en iles

2.1 Cadre de travail

Comme le rappellent les auteurs de [2], plusieurs paramètres définissent un modèle en iles :

- le nombre d'individus pouvant migrer,
- la politique de sélection des individus migrants,
- la politique de remplacement des individus migrants,
- la topologie de la communication entre sous-populations, et
- la nature synchrone ou asynchrone de la connexion entre sous-populations.

Nous proposons ici une modélisation qui généralise tous ces paramètres, et tout en permettant de rendre le modèle en iles dynamique.

Nous représentons ce dernier par un graphe où les nœuds représentent les iles (les sous-populations) et les arcs les possibilités de migrations. Chaque arc possède une valeur qui représente la probabilité pour un individu de migrer d'une ile vers une autre. L'auto-adaptation de ce modèle est gérée à l'aide d'un mécanisme de récompense/pénalité. Les probabilités de migration (valeurs des arcs) sont mises à jour après chaque cycle de migration en fonction des derniers effets de la migration. Si l'ile qui accueille un individu voit sa population améliorée (resp. détériorée) alors la probabilité de migration correspondante est augmentée (resp. diminuée). Ici, la qualité des populations est évaluée en terme de fitness moyenne. diversity of the population if the modeling imposes it.

Le contrôle dynamique des paramètres (comme le taux de migration) rend possible de manipuler des iles de différentes tailles (sauf si nous décidons de l'interdire). Ce mécanisme empêche les sous-populations de faible qualité de dépenser autant d'effort de calcul que celles de bonne qualité et permet de gérer la fusion des populations. Si chaque ile est associée à un opérateur de mutation différent, un effort de recherche locale différent ou à une paramétrisation locale, alors l'algorithme pourra dynamiquement fournir une répartition bien adaptée des individus durant tout le processus de recherche.

2.2 Mise à Jour Dynamique du Modèle

La figure 1 est un exemple de modèle avec trois iles ($i1$, $i2$ et $i3$). La figure 1.a représente les iles avec leurs individus ainsi que les valeurs de migration (probabilités) d'une ile à une autre. Dans la figure 1.b, la destination de chaque individu est choisie en fonction des valeurs des probabilités de migration. La plupart des individus restent sur la même ile (les valeurs des boucles sont généralement plus élevées) mais plusieurs individus migrent vers d'autres iles (deux individus vont de $i1$ à $i2$ et un de $i3$ à $i2$). Après ces migrations, sur chaque ile, les opérateurs tels que le croisement, la mutation ou la recherche locale sont appliqués sur les individus. On met ensuite à jour les valeurs de migration en fonction de la provenance des parents dont les fils (*i.e.* individus obtenus par croisement) améliorent (resp. détériorent) la population. Pour chaque parent, la valeur de l'arc entre la dernière ile qu'il a visitée et son ile actuelle est augmentée (resp. diminuée) pour prendre en compte l'impact de la migration. Par exemple, sur la figure 1.c, si nous observons seulement $i2$ et que la valeur de récompence/pénalité est fixée à 5 points (± 0.05), plusieurs valeurs sont mises à jour :

- ($i1 \rightarrow i2$) diminue de 0.50 à 0.40 car deux individus venant de $i1$ ont produit un individu fils détériorant la population de $i2$;
- ($i2 \rightarrow i2$) diminue de 0.95 à 0.85 car les individus de l'ile $i2$ n'améliorent pas la population de $i2$;
- ($i3 \rightarrow i2$) augmente de 0.20 à 0.30 car l'individu venant de $i3$ a produit un individu fils améliorant la population de $i2$;
- dû à la normalisation, d'autres valeurs en relation avec $i2$ doivent être ajustées.

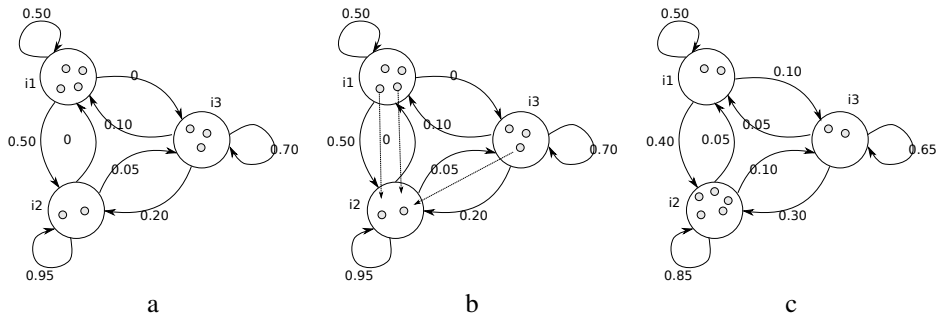

FIGURE 1 – Communication entre les sous-populations avec une représentation en graphe complet

3 Modèles en iles auto-adaptatifs

Dans cette section nous proposons de mesurer l'efficacité générale du schéma dynamique de modèle en iles (MID), appliqué à deux problèmes d'optimisation classiques : le problème du sac à dos 0/1 (0/1 KP) et le problème MAX-SAT. Pour cette étude, l'objectif principal n'est pas de proposer un algorithme puissant qui égale les meilleures méthodes connues mais plutôt de mesurer la pertinence d'un tel modèle. C'est pour cette raison que nous avons déconnecté les opérateurs de recherche locale et que nous nous intéressons uniquement à un AG classique partitionné en iles.

Pour ces deux problèmes, nous comparons les performances de quatre configurations basiques d'algorithmes génétiques :

- Un AG classique d'une seule ile (AG),
- Un algorithme MID standard ($\mathrm{MID}_{\mathrm{stand}}$),
- Un aglorithme MID paramétré avec une topologie en anneau unidirectionnel $\mathrm{MID}_{\mathrm{ring}}$ qui simule un modèle en ile classique avec des migrations circulaires (à chaque processus de migration, les meilleurs individus migrent vers les populations suivantes),
- Un AG parallèle ($\mathrm{AG}/\!\!/$) avec plusieurs iles mais pas de migration.

Notons que l'AG classique correspond à un modèle en iles où l'intervalle des migrations est minime tandis, que l'AG parallèle est un modèle en iles sans aucune migration (ou avec un intervalle de migration infini).

3.1 Paramétrage des algorithmes génétiques

Les quatre configurations des AG ont deux types de caractéristiques. Toutes les valeurs numériques sont empiriquement obtenues et confirmées par la méthode REVAC [20].

1. Caractéristiques intra-iles :
   - type d'AG : steady-state
   - élitisme : oui
   - sélection : tournoi
   - mutation : aléatoire sur les fils avec une probabilité de 0.5
   - croisement : uniforme
2. Caractéristiques inter-iles :
   - nombre d'iles : 20
   - nombre total d'individus : 600
   - répartition de départ : équilibré (30 individus par ile)
   - nombre total de croisements : 216000 (360×600 individus)
   - probabilité initiale de migration : voir section suivante
   - récompense : 5 points
   - penalité : 5 points

Le nombre de croisements pour une ile entre chaque migration est proportionelle à son nombre d'individus. Ce choix assure le même taux de croisements pour chaque ile quelle que soit sa taille.

Probabilités initiales de migration Pour donner la même puissance d'attractivité à chaque ile, les probabilités initiales de migrations doivent être symétriques. Au départ, la probabilité de rester sur la même ile est beaucoup plus élevée que celle de migrer vers une autre ile afin d'exploiter les populations initiales. Par exemple, la matrice initiale correspondant à un modèle en iles avec trois iles pourrait être le suivant :

		Destination		
		$i1$	$i2$	$i3$
Source	$i1$	0.75	0.125	0.125
	$i2$	0.125	0.75	0.125
	$i3$	0.125	0.125	0.75

3.2 Conditions expérimentales

Les algorithmes utilisés dans nos expériences sont appliqués 10 fois pour chaque instance. Pour s'assurer que la différence de comportement n'est pas due à la population initiale, le même jeu de 10 graines aléatoires sont utilisées

par tous les algorithmes. Les résultats présentés dans les tables sont des moyennes. Les écarts-types étant tous très faibles, ils ne sont pas reportés spécifiquement.

3.3 Problème du sac à dos 0/1

Le problème du sac à dos 0/1 est un problème combinatoire bien connu. Ayant n objets dont le poids w_i et la valeur v_i sont connus ($x_i \in \{1, \ldots, n\}$), le but est de trouver un sous-ensemble d'objets de valeur maximale tel que le poids total ne dépasse pas une capacité donnée W. Dans le problème binaire 0/1, chaque objet ne peut être sélectionné qu'une seule fois ($x_i \in \{0, 1\}$, où x_i est le nombre de copies sélectionnées pour l'objet i).

Plus précisément, le problème du sac à dos 0/1 est formulé comme un problème d'optimisation par :

$$\text{maximise} \sum_{i=1}^{n} v_i x_i, \text{ t.q.} \sum_{i=1}^{n} w_i x_i \leq W, \; x_i \in \{0, 1\}$$

Pour plus d'informations sur ce problème, nous invitons le lecteur à se référer à [15].

Les algorithmes à base d'iles $\mathrm{MID}_{\mathrm{stand}}$ et $\mathrm{MID}_{\mathrm{ring}}$ ainsi que les autres algorithmes $\mathrm{GA}_{\mathrm{classic}}$ et $\mathrm{AG}/\!\!/$ ont été testés sur cinq instances du problème. Ces instances ont été générées en se basant sur la définition donnée par [4] et le générateur proposé dans [17] avec les paramètres suivants :

- nombre d'objets $\in \{100, 250, 500, 1000, 2000\}$
- coefficient : 10000
- type : avis subset-sum
- nombre de tests en séries : 1000

Les expériences ont montré que seules les trois dernières instances (celles avec 500, 1000 et 2000 objets) sont représentatives pour comparer les algorithmes. Les deux premières ne présentent pas de difficulté particulière et les résultats sont donc similaires quel que soit l'algorithme. Nous nous focaliserons donc sur les trois instances aléatoires : n500, n1000 et n2000.

La table 1 montre l'efficacité de chaque méthode sur ces intances. Il n'est pas surprenant que le modèle en iles dynamique soit plus performant que les AG traditionnels. Cependant, les différences de performances sont assez importantes sachant que les dernières améliorations sont souvent plus dures à trouver pour le problème du sac à dos. Un point intéressant dans cette expérience est que le schéma de rotation classique ($\mathrm{MID}_{\mathrm{ring}}$) n'est pas compétitif ; comparativement, l'AG classique fournit souvent de meilleurs résultats pour les deux instances les plus difficiles. La principale raison provient probablement de la taille relativement petite des iles (20 individus) qui est adaptative au cours du processus de recherche pour $\mathrm{MID}_{\mathrm{stand}}$ alors qu'elle reste inchangée pour $\mathrm{MID}_{\mathrm{ring}}$.

3.3.1 Le Problème MAX-SAT

Pour une formule booléenne donnée en CNF (conjonctions de clauses qui sont des disjonctions de littéraux), le problème MAX-SAT consiste à fournir une affectation aux variables booléennes telle que le nombre de clauses vraies soit maximum. La formule est satisfiable si et seulement s'il existe une affectation qui rend vraies toutes les clauses.

Trois instances satisfiables sont utilisées pour nos expériences :

- f600 : instance aléatoire avec 600 variables et 2550 clauses,
- f1000 : instance aléatoire avec 1000 variables et 4250 clauses, et
- qg1-7.suffled : instance de carré latin avec 686 variables et 6816 clauses.

Dans la table 2 nous observons que $\mathrm{MID}_{\mathrm{stand}}$ fournit les meilleurs résultats pour les trois instances. $\mathrm{AG}/\!\!/$ est le moins performant et AG obtient des résultats légèrement moins bons que $\mathrm{MID}_{\mathrm{ring}}$. Les résultats pour $\mathrm{MID}_{\mathrm{stand}}$ et $\mathrm{MID}_{\mathrm{ring}}$ sont calculés avec la meilleure fréquence de migration empiriquement trouvée (voir ci-après).

3.3.2 Discussion

Comme nous l'avons vu précédemment, $\mathrm{MID}_{\mathrm{stand}}$ obtient des résultats encourageants comparativement aux autres modèles d'algorithmes génétiques testés. Pourtant, la différence entre $\mathrm{MID}_{\mathrm{stand}}$ et $\mathrm{MID}_{\mathrm{ring}}$ provient uniquement du type de migration, tandis que les différences entre $\mathrm{AG}/\!\!/$, AG et $\mathrm{MID}_{\mathrm{stand}}$ se fait sur la fréquence des migrations. Cette fréquence, ou plutôt période dans ce cas, est donnée par un nombre moyen de croisements par individu. En conséquence, le nombre de croisements entre deux phases de migrations diffère d'une ile à l'autre et dépend de leur taille (*i.e.* de leur nombre d'individus).

L'algorithme génétique classique AG peut être considéré comme un modèle en ile avec une période de migration très faible ; l'algorithme parallèle $\mathrm{AG}/\!\!/$ aurait lui une période de migration maximale, telle qu'il n'y ait aucune migration durant l'intégralité de la recherche (on rappelle qu'une périodicité faible correspond à une fréquence élevée). Entre ces deux algorithmes, un modèle en ile dynamique peut utiliser des fréquences différentes, qui engendreront des comportements différents. On peut observer figure 2 l'impact des fréquences de migrations sur l'ensemble des instances 0/1 KP et MAX-SAT mentionnées dans cet article.

Il est intéressant de noter que, lorsque la période est supérieure à 8 (*i.e.* 8 croisements par individu en moyenne sur chaque ile), $\mathrm{MID}_{\mathrm{stand}}$ et $\mathrm{MID}_{\mathrm{ring}}$ fournissent des résultats équivalents. Une explication possible est que la population a le temps de converger suffisamment avant la première phase migratoire. En effet, dans nos expérimentations, 240 croisements (période : 8, individus par ile : 30

Instance	AG	$\mathrm{MID}_{\mathrm{stand}}$	$\mathrm{MID}_{\mathrm{ring}}$	AG//
n500	755 626.54	**760 620.50**	755 818.38	748 230.25
n1000	1 485 393.86	**1 502 549.22**	1 483 248.36	1 465 757.47
n2000	2 866 752.92	**2 910 891.46**	2 853 072.50	2 832 290.37

TABLE 1 – Comparaisons entre MID et des AG classiques

Instance	Nb Clauses	AG	$\mathrm{MID}_{\mathrm{stand}}$	$\mathrm{MID}_{\mathrm{ring}}$	AG//
f600	2550	2513.80	**2533.40**	2518.70	2357.20
f1000	4250	4174.20	**4208.90**	4174.10	3890.50
qg1-7.shuffled	6816	6756.30	**6787.00**	6776.50	6211.70

TABLE 2 – Comparaison entre MID et les AG classiques

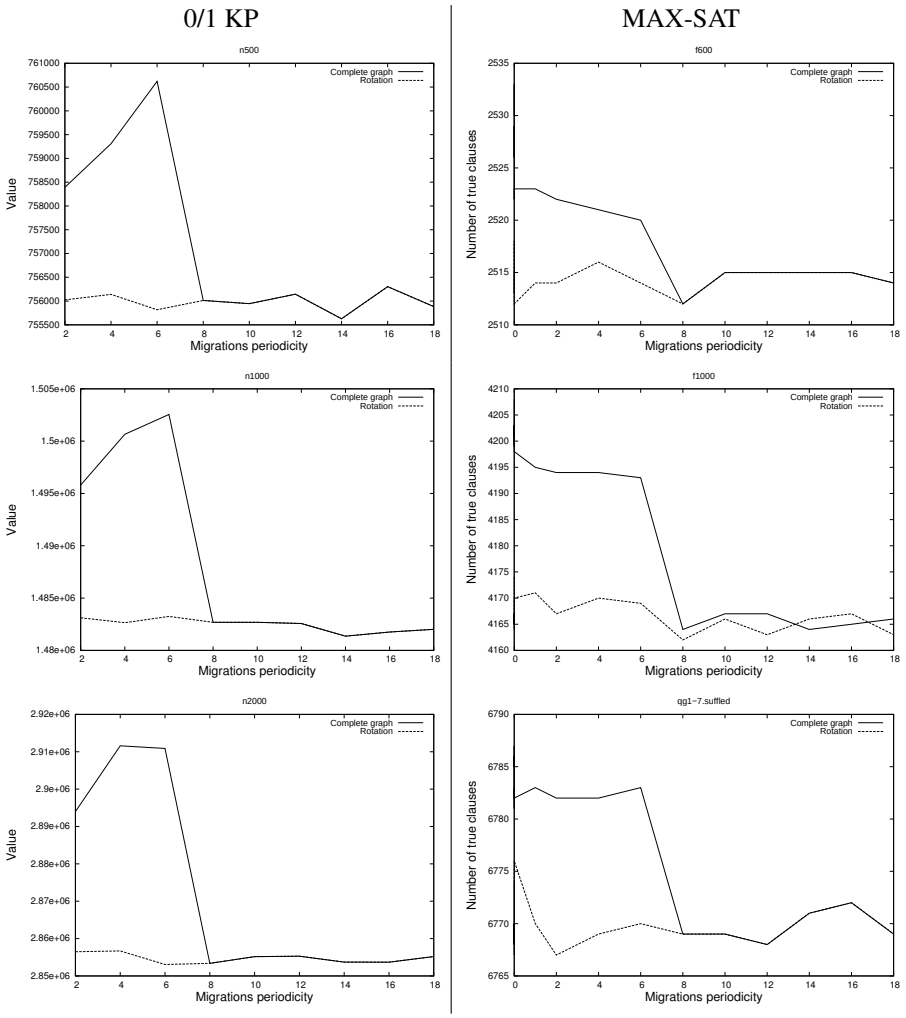

FIGURE 2 – Puissance de l'algorithme MID en fonction de la fréquence de migration (période élevée = migrations plus espacées)

au départ) sont appliqués sur la population de chaque ile avant la première migration. Si chaque population contient seulement des clones ou des individus très proches après cette première phase évolutionnaire, alors un schéma rotatif des migrations produira le même effet réduit qu'un modèle en ile basé sur un graphe complet. Précisons que

pour cette étude, nous avons délibérément retiré tout contrôle de diversité au sein des populations, afin d'observer plus précisément le comportement des différents modèles. C'est pour cette raison que les scores atteints sont moins bons que ceux du tableau 2.

Une autre observation est la faible variabilité des performances de $\mathrm{MID}_{\mathrm{ring}}$ lorsque l'on fait varier la périodicité des migrations. Hormis pour l'instance `qg1-7.suffled`, où les résultats sont meilleurs avec une faible périodicité, ce paramètre ne semble pas déterminant dans le cadre d'un schéma rotatif des migrations.

Avec une périodicité des migrations inférieure à 8, le schéma en graphe complet est clairement meilleur que le classique schéma rotatif. Pour les instances du 0/1 KP, une valeur fixée à 6 semble être la plus appropriée dans notre étude, tandis qu'une valeur plus faible fournira de meilleurs résultats pour les instances MAX-SAT. En fonction des problèmes, des instances ou des algorithmes évolutionnaires utilisés, une plus grande vitesse de convergence des individus profitera de phases migratoires plus rapprochées, mais une simple régulation de la convergence réduira ces différences d'efficacité. Il est également possible d'auto-réguler le rythme des fréquences migratoires, soit explicitement en faisant varier le paramètre durant la recherche à l'aide d'un contrôleur adapté, soit implicitement dans la définition des récompenses et pénalités. En effet, en renforçant ou relâchant les probabilités de migration des boucles (probabilité qu'un individu reste sur son ile lors d'une phase migratoire), on obtient un plus faible ou plus fort taux de migration dans le modèle.

4 Algorithmes évolutionnaires autonomes

Dans la section précédente, on présentait une évolution dynamique de la topologie du modèle, ce qui a pour effet de déterminer à tout moment des probabilités de migration pertinentes entre les différentes populations, dans le cadre d'un algorithme génétique classique partitionné.

Dans cette section, nous utiliserons le même cadre de travail pour sélectionner de manière autonome les opérateurs de recherche locale au sein d'un algorithme évolutionnaire classique. Pour mesurer de manière précise les apports de ce modèle, nous l'utiliserons au sein d'un algorithme de recherche locale à population et sans croisement. À chaque ile sera associé un opérateur de recherche locale particulier. L'objectif est ici non pas d'anticiper les meilleurs croisements entre les individus, mais de détecter à tout moment de la recherche le ou les opérateurs de recherche locale le(s) plus pertinent(s). Cette utilisation du cadre de travail constitue une approche originale dans la définition d'algorithmes autonomes [13, 8].

La mise à jour dynamique des probabilités de migrations entre les iles suivra le même modèle que dans la section 2.2, à ceci près qu'il faut estimer la pertinence des opérateurs de recherche locale et non des croisements. Dans la section précédente, nous évaluions les fils des individus immigrants afin de récompenser ou pénaliser les différents flux migratoires entrants. Dans cette section, le poids des arcs sera cette fois impacté en fonction du bénéfice obtenu du point de vue de l'individu immigrant (sur lequel l'opérateur de recherche locale propre à l'ile aura été appliqué). C'est-à-dire qu'à chaque stade de la recherche, les arcs entrants des iles comportant les meilleurs opérateurs de recherche locale vont se renfoncer et ainsi attirer une plus grande quantité d'invividus, ce qui va accélérer la recherche d'un point de vue global. Afin de rendre chaque opérateur toujours accessible (certains opérateurs seront plus efficaces en fin de recherche qu'au début, et réciproquement), il est impératif soit d'imposer un poids minimal à chaque arc du modèle, soit d'introduire un bruit dans la mise à jour des poids. Ces condisérations seront prises en compte lors du paramétrage du modèle, comme le fait de rendre l'algorithme plus ou moins intensif et réactif, à la manière des algorithmes de colonies de fourmis.

Nous proposons de mesurer l'intérêt d'une telle approche en l'appliquant au problème ONE-MAX. Il s'agit d'un problème très simple, dont on connait théoriquement les meilleurs opérateurs de recherche locale à appliquer en fonction de l'avancée de la recherche.

4.1 Le Problème ONE-MAX

Le problème ONE-MAX consiste à retrouver plus ou moins à l'aveugle une chaine de bits ayant un maximum de 1. Notre problématique consiste à débuter la recherche avec une chaine de bits initialisée par une suite de 0 pour la terminer avec une suite de 1, en utilisant uniquement des séries de flips aléatoires. En début de recherche, les meilleurs opérateurs de recherche locale sont ceux permettant de *flipper* un maximum de bits, tandis qu'en fin de recherche, les opérateurs effectuant un nombre réduit de flips sont censés être les plus efficaces.
Concrètement, nous allons travailler avec une chaine de l bits (dans notre expérience 1000) et nous utiliserons 4 opérateurs différents :

- *1-flip*, opérateur flippant 1 bit sélectionné aléatoirement parmi les l ;
- *3-flips*, opérateur flippant 3 bits sélectionnés aléatoirement parmi les l ;
- *5-flips*, opérateur flippant 5 bits sélectionnés aléatoirement parmi les l ;
- *1/l-flip*, opérateur flippant chacun des l bits avec une probabilité de $1/l$.

L'algorithme proposé peut donc être vu comme un algorithme de descente avec sélection du premier mouvement améliorant. Son originalité est que l'opérateur de recherche

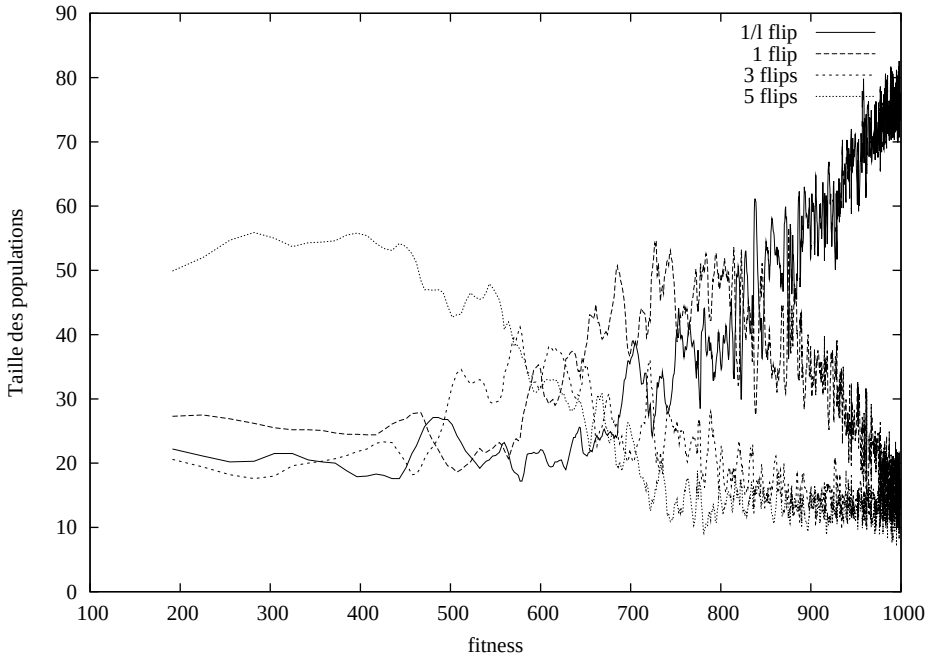

FIGURE 3 – Evolution de la taille de la population des iles en fonction de la fitness globale

locale appliqué à un individu change en fonction de sa fitness. La figure 3 montre l'évolution de la taille des populations des iles correspondant aux opérateurs en fonction de la valeur de la meilleure fitness obtenue depuis le début de la recherche.

4.2 Observations

Nous pouvons observer que l'ile associée à l'opérateur *5-flips* attire le plus les individus en début de recherche. Lorsque la fitness atteint 60% de la fitness optimale (1000 dans notre cas), l'opérateur *3-flips* devient le plus efficace et la probabilité de migration vers son ile associée est donc augmentée (ce qui engendre l'accroissement de la population de son ile). Ensuite, très rapidement, l'opérateur *1-flip* est le plus améliorant et enfin, lorsqu'il devient très difficile d'améliorer les individus, l'opérateur *1/l-flip* apparaît être le plus efficace. Toutes ces observations sont cohérentes avec les résultats déjà observés dans des papiers étudiant l'impact des opérateurs de recherche locale pour le problème ONE-MAX [9]. Notre cadre de travail nous a donc permis de mettre en place un algorithme adaptatif de manière très simple. Pour tester d'autres opérateurs, l'ajout d'autres iles peut être envisagé.

5 Conclusion

Ce travail présente un cadre de travail pour utiliser des modèles en iles dynamiques, permettant tout d'abord de généraliser différents schémas topologiques des modèles classiques, mais surtout d'autoréguler les migrations. Cette modélisation sous forme de graphe complet étiqueté permet de définir tout type de topologie, des graphes sans arc (tous les poids sont nuls, c'est le cas des AG parallèles) aux modèles en iles fréquemment utilisés comme les anneaux ou les treillis. Si la régulation dynamique est activée, alors la topologie est modifiée durant la recherche grâce aux récompenses et pénalités calculées en fonction de l'impact des précédentes migrations. Contrairement aux politiques migratoires des modèles en iles traditionnels, où les migrations sont évaluées *a priori* en mesurant la divergence entre les individus (qui est alors davantage une répartition guidée des individus visant à produire une certaine diversité, mais un non-sens pour un algorithme dit *inspiré par la nature*), nous encourageons (resp. dissuadons) les mouvements migratoires qui ont donné lieu dans un passé proche à de bonnes (resp. mauvaises) descendances, en terme de fitness et / ou de diversité. Durant la recherche, cette autorégulation des probabilités de migrations rend le modèle plus ou moins dynamique en terme de quantités de migrations, ce qui favorise soit la diversification, soit l'intensification (ou les deux, à des endroits différents du modèle).

Les expérimentations réalisées sur les problèmes 0/1 KP et MAX-SAT ont montré qu'un tel schéma dynamique, même avec une paramétrisation basique, apporte de bons résultats, en particulier en comparaison avec les modèles en iles statiques classiques de type anneau unidirectionnel.

Un aspect particulièrement intéressant de notre modèle

est qu'en paramétrant différemment chaque ile, il va affecter de manière autonome ses ressources (individus) sur les iles produisant de meilleurs résultats, c'est-à-dire ayant un paramétrage plus adéquat *à un certain moment de la recherche*. Par exemple, si à chaque ile on associe un opérateur de recherche locale différent, alors le modèle en ile dynamique s'apparentera à un algorithme autonome sélectionnant les opérateurs les plus pertinents au cours de la recherche. C'est ce que nous avons montré empiriquement dans la section 4 sur un problème éprouvé (ONE-MAX), où l'algorithme parvient à répartir les individus sur les opérateurs les plus efficaces, et en étant bien plus réactif que d'autres algorithmes autonomes basés sur l'apprentissage. C'est ce dernier point que nous mesurerons plus précisément dans le cadre de futurs travaux.

Remerciement

Nous souhaitons remercier Sébastien Verel pour ses conseils sur le problème ONE-MAX, ainsi que pour son aide et ses commentaires précieux à propos de la mise à jour des poids dans le cadre de la sélection autonome des opérateurs.

Références

[1] Lourdes Araujo, Juan Julián Merelo Guervós, Carlos Cotta, and Francisco Fernández de Vega. Multikulti algorithm : Migrating the most different genotypes in an island model. *CoRR*, abs/0806.2843, 2008.

[2] Lourdes Araujo, Juan Julián Merelo Guervós, Antonio Mora, and Carlos Cotta. Genotypic differences and migration policies in an island model. In *GECCO*, pages 1331–1338, 2009.

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

[4] Vasek Chvátal. Hard knapsack problems. *Operations Research*, 28 :1402–1411, 1980.

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

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

[7] Taisir Eldos. A new migration model for the distributed genetic algorithms. In *Proceedings of the International Conference on Scientific Computing (CSC ?06), Las Vegas, NV*, pages 26–29.

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

[9] Álvaro Fialho, Luís Da Costa, Marc Schoenauer, and Michèle Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *LION*, pages 176–190, 2009.

[10] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, Mass., 1989.

[11] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[12] David E. Goldberg, Kalyanmoy Deb, and James H. Clark. Genetic algorithms, noise, and the sizing of populations. *COMPLEX SYSTEMS*, 6 :333–362, 1991.

[13] W. Gong, Á. Fialho, and Z. Cai. Adaptive strategy selection in differential evolution. In *Genetic and Evolutionary Computation Conference,GECCO*, pages 409–416. ACM, 2010.

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

[15] Michail G. Lagoudakis. The 0-1 knapsack problem – an introductory survey. Technical report, University of Southwestern Louisiana, 1996.

[16] Frédéric Lardeux and Adrien Goëffon. A Dynamic Island-Based Genetic Algorithms Framework. In K. Deb et al., editor, *Simulated Evolution and Learning*, volume 6457 of *Lecture Notes in Computer Science*, pages 156–165. Springer, 2010.

[17] Silvano Martello, David Pisinger, and Paolo Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Manage. Sci.*, 45(3) :414–424, 1999.

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

[19] Pablo Moscato. Memetic algorithms : a short introduction. pages 219–234, 1999.

[20] Volker Nannen, Selmar K. Smit, and Agoston E. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. In *Proceedings of the 10th*

international conference on Parallel Problem Solving from Nature, pages 528–538, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] J. N. Richter. *On Mutation and Crossover in the Theory of Evoutionary Algorithms*. PhD thesis, Montana State University, 2010.

[22] Marek Rucinski, Dario Izzo, and Francesco Biscani. On the impact of the migration topology on the island model. *CoRR*, abs/1004.4541, 2010.

[23] Zbigniew Skolicki. *Linkage in Island Models*, chapter Studies in Computational Intelligence, pages 41–60. Springer Berlin / Heidelberg, 2008.

[24] Zbigniew Skolicki and Kenneth A. De Jong. The influence of migration sizes and intervals on island models. In *GECCO*, pages 1295–1302, 2005.

[25] William M. Spears. *Evolutionary Algorithms : The Role of Mutation and Recombination*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

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

[27] Rasmus K. Ursem. Diversity-guided evolutionary algorithms. In Juan J. Merelo Guervos, Panagiotis Adamidis, Hans-Georg Beyer, Jose Luis Fernandez-Villacanas Martin, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII, 7th International Conference, Granada, Spain, September 7-11, 2002, Proceedings*, volume 2439 of *Lecture Notes in Computer Science*, pages 462–474. Springer, 2002.

[28] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm : On separability, population size and convergence. *Journal of Computing and Information Technology*, 7 :33–47, 1998.

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