Algorithme évolutionnaire adaptatif : une rétrospective

Mariam Bouzid[1]

Université d'Angers, France,
mariam.bouzid@etud.univ-angers.fr

Résumé La difficulté à laquelle se heurte les algorithmes évolutionnaires – en vertu du No-Free-Lunch Theorem – est leur spécialisation et la détermination d'un paramétrage optimal pour un problème en particulier. Pour ôter l'aspect arbitraire à la configuration des paramètres, il a été proposé des approches de contrôle dynamique. Parmi ces méthodes se trouve la sélection adaptative d'opérateurs utilisée conjointement à l'apprentissage par renforcement. Ce dernier est un modèle de prise de décision autonome, adaptée et conditionnée. Son fondement provient de l'abstraction du système de récompense inspiré de travaux sur les processus cognitifs. Cette approche apporte des résultats très encourageants, est souple en terme de représentatibilité mais génère des difficultés d'interprétabilité et de validation. Nous proposons ici une rétrospective sous la forme d'une implémentation d'un algorithme évolutionnaire adaptatif et d'une étude comparative à travers le problème canonique du *One-max*. Ces observations nous permettrons une validation du modèle. Enfin, nous l'appliquerons au problème du voyageur de commerce, un problème $\mathcal{NP}$ Complet...

Keywords: contrôle de paramètres, algorithme évolutionnaire adaptatif, apprentissage par renforcement

Table des matières

1 Introduction

Avant d'évoquer les aspects techniques et formels du problème nous pouvons avoir quelques intuitions sur l'architecture d'un algorithme adaptatif.

Le concept essentiel est l'**apprentissage**, lié à la notion d'**adaptation**. Lors de l'apprentissage, pour faire analogie avec la pensée, il y a progression d'une raison sensitive à une raison intellectuelle.

Le principe du **système de récompense** reprend ces observations : une entité sensitive perçoit une stimulation ou une inhibition et réagit en conséquence au moyen d'un mécanisme rationnel, en résulte alors une **décision**. Cette interaction entre sens et raison est l'essence même de l'apprentissage puisqu'on a alors un comportement conditionné et l'apparition d'une mémoire associative.

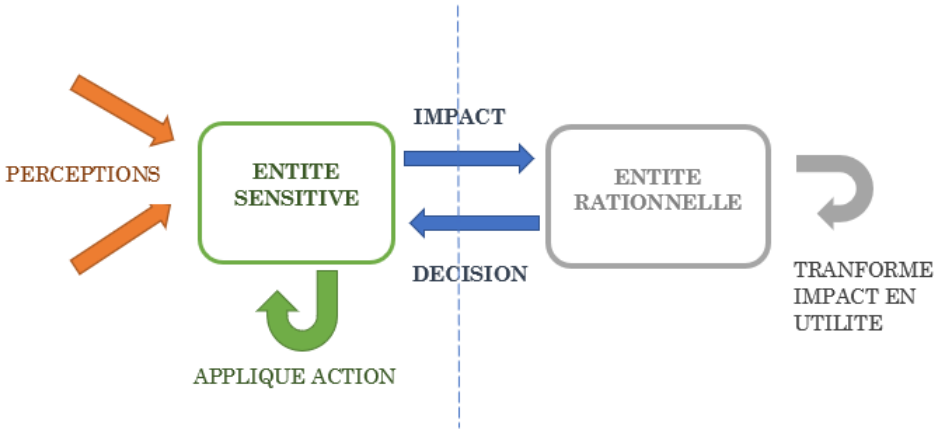

Figure 1. Le système de récompense - l'intéraction entre les entités génère une décision

La première partie présente le modèle des algorithmes évolutionnaire qui constituera l'entité apprenante. Puis nous introduirons une couche de contrôle adaptative afin de mettre en place l'apprentissage. Enfin des expérimentations seront conduites premièrement à des fins de validation de la méthode (via le *One-max*) et enfin à des fins d'illustration et de résolution (via le problème du voyageur de commerce).

2 `EA` : algorithme évolutionnaire

Le modèle des algorithmes évolutionnaires (EA) est librement influencé par la théorie de l'évolution. Eiben et Smith dans [1] l'explique comme suit : une population d'individus lutte pour la survie et la reproduction. L'aptitude (*fitness*) de ces individus est déterminé par l'environnement qui se régule par le mécanisme de sélection naturelle.

Dans notre contexte, cette analogie s'exprime sur un problème d'optimisation où une population d'individus (ensemble de configurations possibles d'un problème) converge - au sens d'une fonction objective - à l'aide d'opérateurs stochastiques (opérateurs de mutation et de recombinaison).

2.1 Formalisme

Data : f fonction objective, pop population, op_m opérateur de mutation, pm probabilité de mutation, op_c opérateur de croisement, pc probabilité de croisement, `CS` critère de sélection, `CR` critère de remplacement
begin
  `init(`pop`)` ▷ Initialiser la population avec une heuristique (random, greedy, recherche locale...) ;
  while ¬ *condition de fin* **do**
    `evaluer(`pop`,`f`)` ;
    $parents \leftarrow$ `selectionner(`pop`,CS,`f`)`;
    $enfants \leftarrow$ `croiser(`$parents, pc, op_c$`)` ;
    foreach $enfant \in enfants$ **do**
      `muter(`$enfant, pm, op_m$`)` ;
    end
    `remplacer(`pop`,CR,`f`)` ;
    `inserer(`$pop, enfants$`)` ;
  end
end

Algorithme 1 : Pseudo-code steady-state

En résumé, un EA est un gestionnaire qui applique des processus stochatisques définis (mutation, croisement) à un ensemble de données structurées (population). De plus, la capacité d'évaluer les individus − au moyen d'une fonction objective f − permet à l'EA de guider les solutions vers l'optimalité, au sens de f.

Généralités
La méthode d'évaluation décrite par `evaluer` consiste à trouver parmi la population, l'individu ayant la meilleure *fitness* ou mérite selon f.

Une mutation op_m est une fonction probabiliste (s'exécute avec pm chances) qui modifie la structure de la solution. Cette opération retourne une solution valide ou non.

Une recombinaison ou croisement op_c est un processus probabiliste (s'exécute avec pc chances) qui construit à partir de k individus k nouveaux individus. Cette opération retourne une solution faisable ou non.

Le critère de sélection `CS` favorise les individus ayant les caractéristiques voulues pour la recombinaison (meilleure *fitness*). Le critère de remplacement `CR` est utilisé pour choisir les individus à enlever de la population (moins bonne *fitness*).

3 AOS pour l'apprentissage

AOS (Adaptive Operator Selection) s'articule autour de deux axes : l'apprentissage et la sélection. En particulier, ici l'AOS s'applique sur un EA.

Quatre politiques stochastiques de Multi-Armed Bandits (MAB) sont implémentées : la Fixed Roulette Wheel, l'Adaptive Roulette Wheel, l'Adaptive Pursuit et l'Upper Confidence Bound. Le Dynamic Island Model (DIM) est également présenté dans le cadre AOS.

3.1 Formalisme

Data : une population *pop*, une politique π, un ensemble d'opérateurs Ω
begin
  `init_politique(`π`)` ;
  while *(¬ condition de fin)* **do**
    $o_i \leftarrow$ `selectionner_operateur(`π, Ω`)` ;
    $f^{(t)} \leftarrow$ `evaluer(`*pop*`,`f`)` ;
    `steady_state(`f, *pop*, o_i`)` ▷ Appliquer une itération du steady state ;
    $f^{(t+1)} \leftarrow$ `evaluer(`*pop*, f`)` ;
    $\Delta f \leftarrow$ `max(`$f^{(t+1)} - f^{(t)}$`,0)` ;
    `maj_gains(`$\Delta f, o_i$`)` ;
    `maj_utilite(`o_i`)` ;
    `appliquer_politique(`π, *pop*`)` ;
  end
end

Algorithme 2 : Pseudo-code AOS

3.2 Renforcement

AOS propose un système de récompense renforcé par l'utilité. A l'itération t est évalué l'impact de l'opérateur courant via le calcul de Δf, les autres opérateurs ont un gain nul puisqu'ils n'ont pas de contribution à cette itération.

La formule pour le calcul de l'utilité effectué par `maj_utilite` pour l'opérateur o_i est

$$u_i^{(t)} = (1-\alpha)u_i^{(t-1)} + \alpha g(o_i, s^{(0)}, \ldots, s^{(t-1)})$$

où o_i est l'opérateur courant, $s^{(0)}, \ldots, s^{(t-1)}$ les anciens états, g la fonction de calcul du gain, α est le taux d'apprentissage.

En prenant des valeurs extrêmes pour α, il est plus facile de se rendre compte de son incidence sur le renforcement : lorsque celui-ci est nul, il n'y a pas de prise en compte du gain de l'opérateur mais une totale et (aveugle ?) confiance en les observations passées. Lorsque $\alpha = 1$ c'est le contraire : on ne considère que ce qui est survenu à l'instant t et on oublie l'impact de la connaissance passée : il n'y a simplement pas d'apprentissage.

Quelques constats
Des valeurs élevées pour α entraine un apprentissage lent. Toutefois, une valeur trop faible compromet l'apprentissage : il est nécessaire d'expérimenter avec un ensemble de valeurs raisonnables afin de déterminer ce facteur au mieux.

3.3 Politique de sélection

Le résultat d'une politique de sélection est l'aboutissement de l'apprentissage, il est conditionné par celui-ci. Dans cette section nous aborderons quelques unes de ces méthodes de sélection d'opérateurs dans le cadre du MAB.

Roulette selection wheel
Cette méthode est très simple à comprendre et à implémenter. A partir d'un vecteur de probabilités à un instant t, $\theta^{(t)} = (\sigma_1^{(t)}, \ldots, \sigma_n^{(t)})$, on procède au tirage aléatoire pondéré d'un opérateur.

ϵ-greedy
Etant donné un vecteur de poids $\theta^{(t)}$, on sélectionne

$$\begin{cases} \textbf{argmax } \theta^{(t)} \text{avec une probabilité de } (1-\epsilon) \\ \text{les autres opérateurs avec une probabilité de } \epsilon \end{cases}$$

Bien entendu, plus ϵ est faible plus la probabilité de sélectionner le meilleur opérateur est grande. Lorsque $\epsilon = 1$ on parle de politique de sélection *greedy*.

Softmax
La fonction exponentielle normalisée ou *softmax* est définie pour tout opérateur i par :

$$\sigma_i^{(t)} = \frac{e^{\sigma_i^{(t)}}}{\sum_{j=1}^{n} e^{\sigma_j^{(t)}}}$$

Cette fonction de transformation peut être utilisée pour les réseaux de neurones d'architecture perceptron multicouches comme fonction d'activation des neurones.

3.4 Fonction et politique de contrôle

Les politiques de contrôle possèdent des particularités qui agissent sur l'**intensification** (examen d'une zone particulière de l'espace de recherche) et la **diversification** (oriente la recherche vers de nouvelles zones de l'espace de recherche).

Dans les fonctions de contrôle, la diversification peut être apportée par le calcul d'un indice (par exemple dans UCB1 à travers un ratio d'occurrence de sélection qui favorisera les opérateurs les moins utilisés). L'intensification est représentée soit par une plus grande utilité, soit par une probabilité de sélection plus grande ou bien par une combinaison des deux facteurs.

Le DIM agit sur l'intensification et la diversification avec le processus stochastique de migration : une île prometteuse pour f est ainsi plus attractive.

Fixed Roulette Wheel
La Fixed Roulette Wheel est une politique seulement à base de probabilité de sélection. Elle garde fixe le poids d'un opérateur i lors d'une exécution.

$$\sigma_i^{(t)} = \sigma_i^{(t+1)}$$

Cette politique est pertinente (dans le sens de l'intensification de la recherche) si les probabilités de sélection sont déterminées par un processus de *tuning*.

Soit n, le nombre d'opérateurs

$$\sigma_i^{(0)} = \frac{1}{n}, \forall i \in \{1, \ldots, n\}$$

dans ce cas, la probabilité de sélection est la même pour tous les opérateurs et celle-ci n'évoluera pas au cours de l'exécution : c'est la politique la plus équitable en terme de diversification mais elle ne tient pas compte de la qualité des opérateurs en fonction des itérations.

Pour le *One-max* le 5-flip est avantageux au début puisqu'il y a beaucoup de bits *off* dans les solutions, toutefois il dégrade beaucoup lorsqu'on se rapproche de f^* (beaucoup de bits *on*).

Cette politique est intéressante si l'on apprécie tous les aspects du problème à traiter, dès lors les opérateurs importants à appliquer à un instant donné sont connus.

Adaptive Roulette Wheel

L'Adaptive Roulette Wheel est également une politique à base de probabilité de sélection mais elle tient compte de l'utilité des opérateurs. Celle-ci est définie pour l'itération suivante par

$$\sigma_i^{(t+1)} = p_{min} + (1 - np_{min}) \frac{u_i^{(t+1)}}{\sum_{k=1}^{n} u_k^{(t+1)}}$$

où n est le nombre d'opérateurs et $p_{min} \in \left[0, \frac{1}{n}\right[$ la probabilité minimale fixée par l'utilisateur.

Elle discerne les opérateurs ayant une moins bonne utilité par le calcul du ratio entre utilité de l'opérateur et la somme des utilités de tous les opérateurs jusqu'à présent. Plus ce ratio est petit, plus la probabilité de sélection en $(t+1)$ tendra vers p_{min} et par conséquent pénalisera cet opérateur.

Elle se comporte comme une Fixed Roulette Wheel équitable si $p_{min} = \frac{1}{n}$, en effet $\sigma_i^{(t+1)} = p_{min}$.

Adaptive Pursuit

Cette politique de contrôle propose une stratégie du *vainqueur rafle tout*.

En effet, AP distingue le meilleur opérateur par son utilité et lui affecte une probabilité supérieure aux autres opérateurs. On peut ajuster l'influence de cette heuristique avec un paramètre β. La fonction de contrôle est définie comme suit

$$\sigma_{i^*}^{(t+1)} = \sigma_{i^*}^{(t)} + \beta(p_{max} - \sigma_{i^*}^{(t)})$$

où i^* est l'opérateur ayant la meilleure utilité et $p_{max} = 1 - (n-1)p_{min}$. Pour les autres opérateurs, on a une probabilité de sélection définie par

$$\sigma_i^{(t+1)} = \sigma_i^{(t)} + \beta(p_{min} - \sigma_i^{(t)})$$

Notons que cette formule prend en compte la probabilité de sélection précédente, cela en fait un mécanisme de mémoire associative.

L'AP se comporte comme une Fixed Roulette Wheel si $\beta = 0$. En effet, on a alors $\sigma_i^{(t+1)} = \sigma_i^{(t)}$ pour tous les opérateurs.

Si $\beta = 1$, on obtient la politique la plus sévère :

$$\begin{cases} \sigma_{i^*}^{(t+1)} = p_{max} = 1 - (n-1)p_{min} \\ \sigma_i^{(t+1)} = p_{min}. \end{cases}$$

Lorque $p_{min} = 0$ alors le meilleur opérateur obtient une probabilité de sélection de 1, les autres opérateurs sont sanctionnés par une probabilité de sélection nulle.

Upper Confidence Bound

UCB est une politique de contrôle qui sélectionne systématiquement l'opérateur répondant au mieux à un critère particulier (ici $UCB1$). Pour un instant (t), chaque opérateur i est associé à un coefficient

$$UCB1(o_i, t) = u_i^{(t)} + \sqrt{\frac{2 \log \sum_{1 \leq k \leq n} nb_k^{(t)}}{nb_i^{(t)}}}$$

où $nb_i^{(t)}$ est le nombre de fois que l'opérateur i a été utilisé aux itérations $(0), \ldots, (t-1)$.

Ce coefficient est composé de deux parties modélisant deux objectifs différents :
— la partie gauche favorise les opérateurs ayant la meilleure utilité (intensification)
— la partie droite favorise les opérateurs les moins utilisés (exploration)

Pour reprendre justement [2], cette politique tente un compromis entre exploitation et exploration.

Toutefois, le fait de choisir systématiquement le meilleur opérateur au sens de $UCB1$ n'est pas tout à fait performant. De plus, les algorithmes de type UCB obtiennent de bons résultats lorsqu'il y a un petit nombre d'opérateurs et une grande variance sur les récompenses, ce qui n'est pas le cas de *One-max* lorsqu'on approche f^*. UCB est pertinent quand il est utilisé conjointement avec une politique de sélection ϵ-greedy ou avec *softmax*. [3]

Dynamic Island Model

DIM propose une démarche sensiblement différente et complexe puisqu'elle considère non pas la population dans son ensemble mais des sous-ensembles d'individus et leur dynamique comme indicateur. Chaque individu est associé à une île ou opérateur.

A chaque itération les individus migrent. Une matrice de transition T est définie pour chaque paire d'opérateurs i et j par

$$T_{ij} = (1-\beta)(\alpha T_i j + (1-\alpha)D_i) + \beta N_i$$

où β est un facteur de bruit avec le vecteur *noisy* N_i , α le taux d'apprentissage, D_i le gain de l'ile i.

Les observations faites sur le *One-max* dans [4] révèlent que les individus convergent vers les opérateurs les plus adaptés à la situation : le 5-flip est prometteur au départ, puis le 3-flip et enfin le 1-flip. Cette sélection d'opérateur adaptative est formalisée dans [4].

Notons que DIM est une métaheuristique similaire à VNS (Variable Neighborhood Search) puisqu'avec le mécanisme de migration, les solutions sont modifiées par différents opérateurs de mutation ce qui change la notion d'optimum.

Remarques

Dans l'implémentation, la matrice de transition n'a d'abord pas été utilisée. C'est plutôt l'architecture parallèle qui a été retenue. La politique de sélection s'effectue au moyen d'un vecteur de probabilités. Lors d'une migration celui-ci indique l'île suivante à l'individu. Ainsi, on peut appliquer AP, FRW et UCB comme politique de contrôle et *roulette*, *softmax* ou *ϵ-greedy* comme politique de sélection. Toutefois, après discussion, l'intérêt d'une matrice de transition se justifie avec des problèmes d'optimisations plus fins que *One-max*.

3.5 En résumé

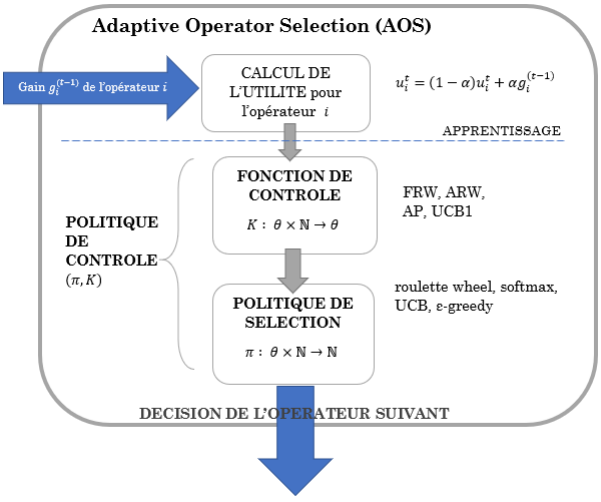

Figure 2. Workflow de l'AOS en reprenant les termes de [2]

4 Expérimentations et validation

4.1 Cadre expérimental : le *One-max*

On étudie le problème du *One-max* de taille $N = 1000$. On se place dans un problème de maximisation. On définit la *fitness* d'une solution $x = (x_1, \ldots, x_N)$ par la somme de ses bits x_i, pour tout $i \in \{1, \ldots, N\}$.

$$f(x) = \sum_{i=1}^{N} x_i$$

La valeur optimale est alors $f^* = N$.

Les opérateurs de mutation de Ω sont
— le bitflip $\frac{1}{N}$ qui inverse des bits avec une probabilité de $\frac{1}{N}$
— le 1-flip qui inverse 1 bit au hasard
— le 3-flip qui inverse 3 bits au hasard
— le 5-flip qui inverse 5 bits au hasard

4.2 Première étude : Paramétrage de l'EA

Objectif 1 Détermination empirique du meilleur paramétrage pour l'EA appliqué au problème du *One-max*. Pour une telle évaluation on utilise les critères suivants :
— atteint f^* rapidement.
— robuste (faible dispersion du nombre d'itérations pour atteindre f^*)
.
Les paramètres étudiés sont de deux natures : 1) *numériques* : probabilité de mutation et de croisement, taille de la population 2) *symboliques* : opérateurs, critères de remplacement et de sélection.

Conditions expérimentales 1 Cette étude a été réalisée avec une implémentation `C++` de l'algorithme évolutionnaire *steady-state* présenté dans la première section. Initialement, la population est consitiué d'individus ayant tous leurs bits à 0.

Protocole 1 Les algorithmes évolutionnaires étant stochastiques nous ne pouvons pas garantir la reproductibilité de l'expérience. Celle-ci est répétée plusieurs pour être statistiquement fiable. Pour chaque paramétrage :
— Lancer au moins 10 exécutions du programme
— Calculer les statistiques de bases des échantillons : médiane, écart-type, moyenne et valeurs extrêmes (avec un script `R`)

4.3 Deuxième étude : Validation de l'apprentissage

Objectif 2 Etude du problème du *One-max* pour valider l'apprentissage et hiérarchisation des politiques de contrôle selon les mêmes critères énoncés précédemment.

Conditions expérimentales 2 La validation de l'apprentissage a également été réalisée avec une implémentation `C++` de l'algorithme proposé en deuxième section. Comme dans la première expérimentation, au départ, les individus ont tous leurs bits *off*.

Paramétrage de l'EA*
— Critère de sélection : 2 individus ayant la meilleure $fitness$.
— Critère de remplacement : 2 individus ayant la moins bonne $fitness$.
— Opérateur de croisement : *monopoint*.
— Probabilité de croisement : $p_c = 1.0$.
— Probabilité de mutation : $p_m = 1.0$.
— Taille de la population : 400.
Paramétrage de l'AOS
— Probabilité minimale de sélection : $p_{min} = 0.1$ ($< \frac{1}{n}$)
— Taux d'apprentissage $\alpha = 0.20$ (DIM $\alpha = 0.85$)
— $\beta = 0.01$

Protocole 2 Pour les mêmes raisons citées ci-dessus, l'expérience est répétée.

Pour chaque politique de contrôle :
— Lancer 50 exécutions du programme
— Calculer les statistiques de bases des échantillons : médiane, écart-type, moyenne et valeurs extrêmes.

4.4 Résultats expérimentaux

4.5 Etude de l'EA : influence des paramètres

Opérateurs Les opérateurs ont d'abord été considérés avec un paramétrage plus ou moins neutre : remplacement des individus par leur fitness, sélection par tournoi (5/2), taille de la population fixée à 80, croisement *monopoint* avec une probabilité de 0.1 et une probabilité de mutation à 1.0.

Opérateurs	Médiane	Moyenne	Ecart-type	Min	Max
bitflip	678841	676329.9	16045.76	651797	696380
1-flip	970134	965766.8	39769.05	884958	1023438
3-flip	–	–	–	–	–
5-flip	–	–	–	–	–

Table 1. Comparaison des opérateurs en fonction du nombre de cycles pour atteindre f^* (échantillon de taille 10)

Lorsqu'ils sont utilisés seuls, les opérateurs *bitflip* et *1-flip* obtiennent des résultats médiocres. On retient que l'opérateur *bitflip* est meilleur que le *1-flip* tant en terme de résultat médians qu'en terme de robustesse (valeur minimale, dispersion faible). Les deux autres opérateurs ont de très mauvais résultats (temps d'exécution élevé).

Taille de la population

Pour étudier l'impact de la taille de la population sur l'EA, l'opérateur bitflip a été utilisé. Les tailles de population qui ont été analysé sont : 10, 25, 50, 100. On reprend les paramètres proposés précédemment.

Taille	Médiane	Moyenne	Ecart-type	Min	Max
10	142387	142210.1	16061.58	124423	175329
25	255067	254818.1	22075.31	225295	299697
50	394041	393533.1	15453.8	367351	417046
100	690957	693399.7	30947.03	661001	764104

Table 2. Comparaison du nombre de cycles pour atteindre f^* avec tailles variables (échantillon de taille 10)

La taille de la population joue sur la performance de l'algorithme : plus elle est grande, plus le nombre de cycles nécessaire pour atteindre l'optimum est élevé.

Probabilité de croisement En reprenant le paramétrage précédent, avec une taille de population à 10 et différentes probabilités de croisement $pc \in \{0.10, 0.25, 0.5, 0.75, 1.0\}$ on obtient les résultats suivant.

pc	Médiane	Moyenne	Ecart-type	Min	Max
0.10	158066	154110.2	18694.23	127648	176138
0.25	62152	62225.1	10127.05	49419	82971
0.50	29054	30890.6	4141.15	27110	40706
0.75	19287	19340	1924.89	16332	22061
1.0	15053	15235.2	1505.74	12614	17468

Table 3. Comparaison du nombre de cycles pour atteindre f^* avec probabilités de croisement pc variables (suivant un échantillon de taille 10)

La probabilité de croisement contribue beaucoup à la qualité de l'algorithme : plus elle est élevée, meilleurs seront les résultats.

Remplacement Avec une population de taille 10, des probabilités $pm = pc = 1.0$ et une sélection par tournoi (3/5), le remplacement par âge n'arrive pas à atteindre la valeur optimale (stagne autour de $f = 500$ en moyenne). C'est en effet le remplacement des moins bons individus qui est le meilleur critère.

Toutefois, avec la sélection des meilleurs individus à la place d'une sélection par tournoi, on obtient les statistiques suivantes :
- médiane = 46222.5
- moyenne = 56977.3
- écart-type : 40628.66
- étendue : min= 14546 / max = 127166

Sélection Avec le paramétrage précédent (avec le remplacement des pires individus au sens de la *fitness*), la sélection des deux meilleurs individus (élite) pour la recombinaison génère une performance supérieure :
- médiane = 10432
- moyenne = 10425.9
- écart-type : 2265.13
- étendue : min= 6894/ max = 14807

Pour résumé Le paramétrage optimal pour *One-max* est le suivant : $pc = pm = 1.0$, taille de la population à 10, sélection élitiste, remplacement des moins bons individus, opérateur *bitflip*.

4.6 Etude de l'AOS

Politique de contrôle π	Médiane	Moyenne	Ecart-type	Min	Max	Etendue
Fixed roulette wheel	8979	9169.22	1741.19	5888	13215	7327
Adaptive roulette wheel	6865	6986.62	1655.24	4739	13811	9072
Adaptive pursuit	6779	6925.48	1285.26	4874	11469	6595
Dynamic Island Model /w AP	**2701**	**2806.86**	**392.88**	**2269**	**3824**	**1555**
Dynamic Island Model /w UCB	**2877**	**3013.14**	**519.05**	**2301**	**4724**	**2423**
Dynamic Island Model	**2818**	**2860.28**	**360.9**	**2302**	**4126**	**1823**

Table 4. Comparaison des politiques de contrôle en fonction du nombre d'itérations pour atteindre f^*

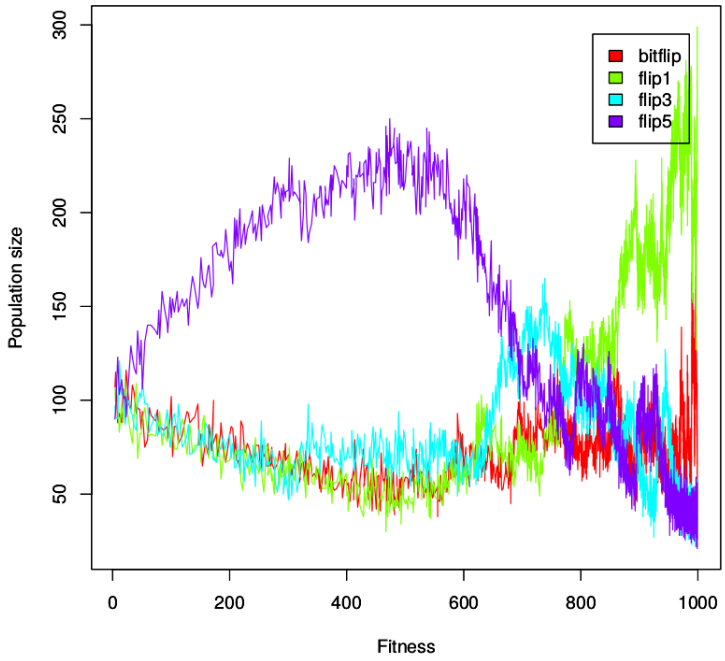

Figure 3. DIM/AP avec une *roulette selection wheel* - Evolution de la taille des populations en fonction de la moyenne de fitness des individus avec $\alpha = 0.20$, $\beta = 0.01$, $p_{min} = 0.10$.

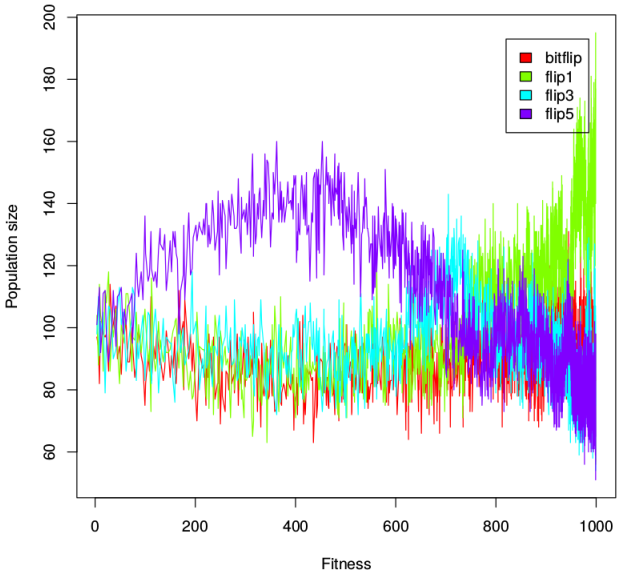

Figure 4. DIM/AP avec *softmax* - Evolution de la taille des populations en fonction de la moyenne de fitness des individus avec $\alpha = 0.20$, $\beta = 0.01$, $p_{min} = 0.10$.

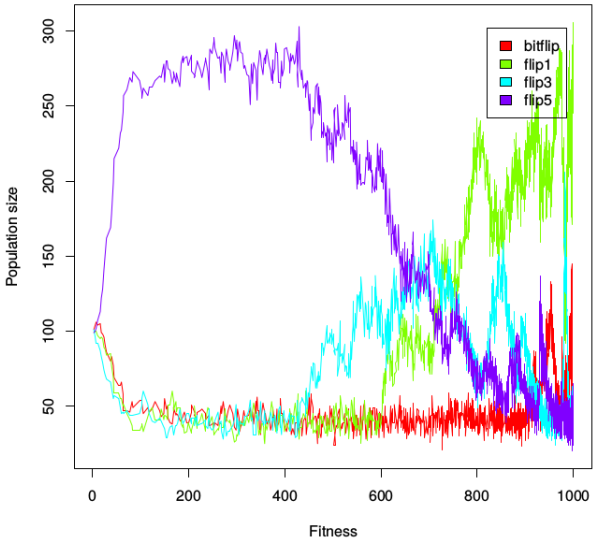

Figure 5. Progression de DIM avec la matrice de transition $\alpha = 0.85$, $\beta = 0.01$,

Les figures 3 et 4 laissent apparaitre que l'apprentissage est effectif : le 5-flip est très attractif jusqu'à la moyenne de *fitness* de 500, puis le 3-flip prend le relais sur un cours intervalle entre 600 et 800, enfin le 1-flip est utilisé pour atteindre f^*. L'opérateur bitflip est toujours moins bon que les autres.

Le tableau 4.6 expose les statistiques établies sur un échantillon de 50 exécutions et ce pour toutes les politiques de contrôle.

DIM se démarque très fortement tant par ses bons résultats que par sa robustesse. Le fait est que l'architecture parallèle de DIM lui procure un grand avantage.

Parmi les autres politiques, AP se distingue le plus en terme de résultats médians et d'étendue. Il vient ensuite ARW ayant des résultats proches et même dans ce cas une meilleure borne inférieure minimale . Enfin, FRW est la moins bonne politique, ce qui était prévisible.

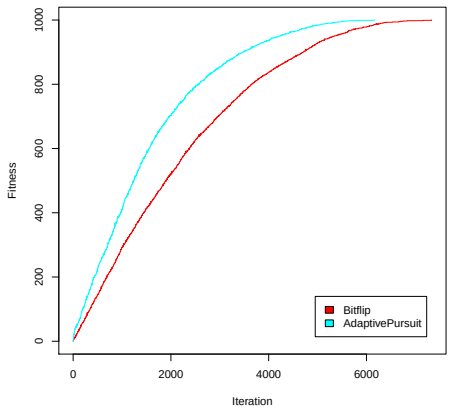

Figure 6. Progression typique d'un EA - Comparaison de l'évolution de la *fitness* en fonction du nombre d'itérations avec les échantillons médians de l'AP et de l'opérateur bitflip seul.

5 Application à un problème $\mathcal{NP}$

Ayant des résultats encourageants pour le problème canonique : l'apprentissage est garanti, nous pouvons maintenant nous pencher sur les problèmes de la classe $\mathcal{NP}$.

5.1 Benchmarking

Un benchmark est un ensemble d'instances d'un problème qui possèdent des caractéristiques communes. Par exemple our le *graph coloring problem*, il existe une ressource à cette adresse. L'idée serait de trouver une séquence d'opérateurs intéressants pour chaque benchmark avec la méthode d'apprentissage par renforcement.

5.2 TSP

Le *Traveling Salesman Problem* est un problème qui consiste à déterminer un ordre total de coût minimal. Lorsqu'il y a n éléments, l'espace de recherche du TSP est $n!$, c'est l'arrangement de n éléments distincts parmi n éléments distincts noté A_n^n. Cela justifie l'utilisation d'une méthode approchée pour résoudre ce problème.

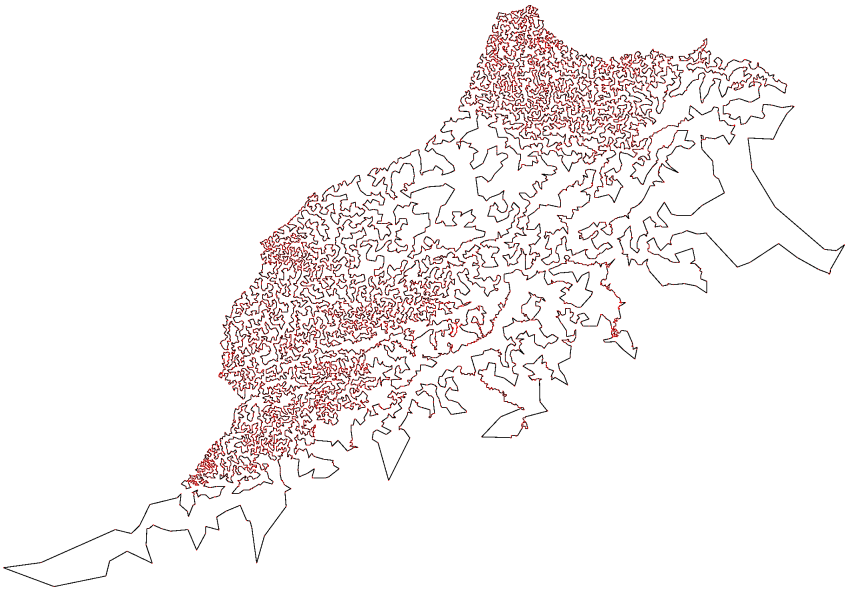

Figure 7. Illustration du TSP avec le tour optimal du Maroc trouvé par Keld Helsgaun (14185 zones habitées, 427377 km)

5.3 Résultats expérimentaux

Les expérimentations ont été réalisées sur deux types d'instances de TSP [5] : les instances aléatoires, les instances de type *drilling*. Il a été utilisé le DIM paramétré avec $\alpha = 0.20$ et $\beta = 0.01$. Les opérateurs de mutations sont le *swap*, l'inversion, l'insertion et le *scramble*.

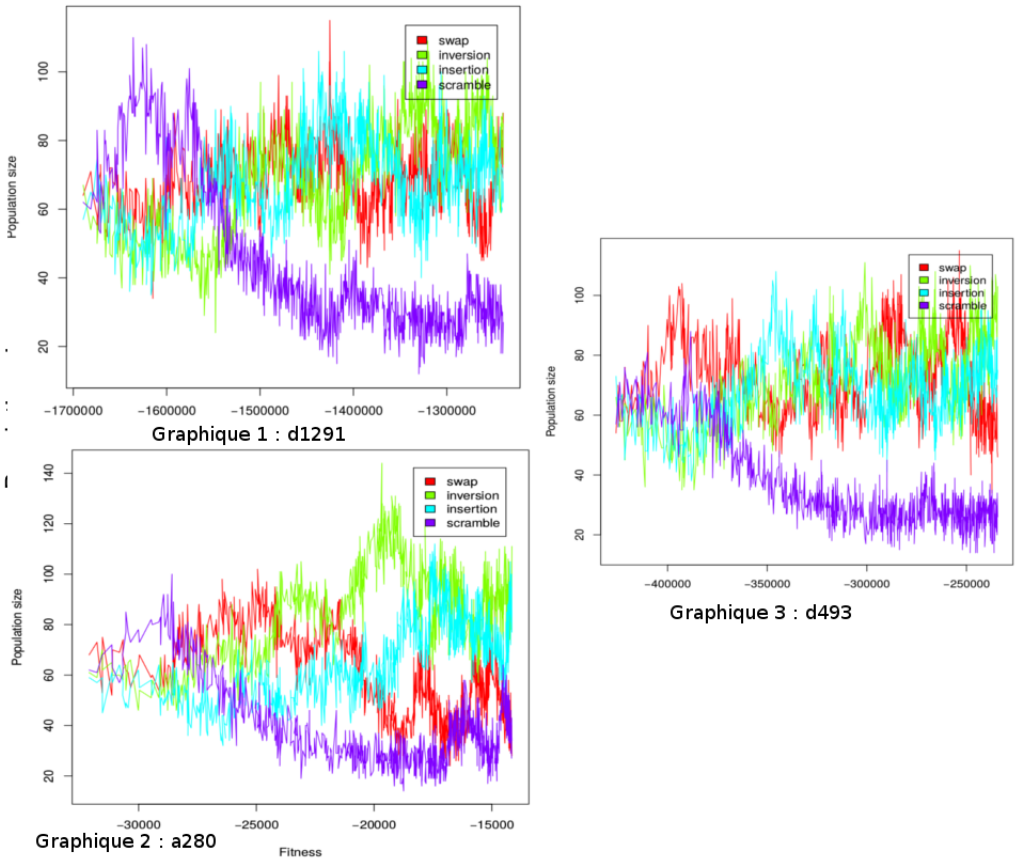

Figure 8. Trois instances de type *drilling*

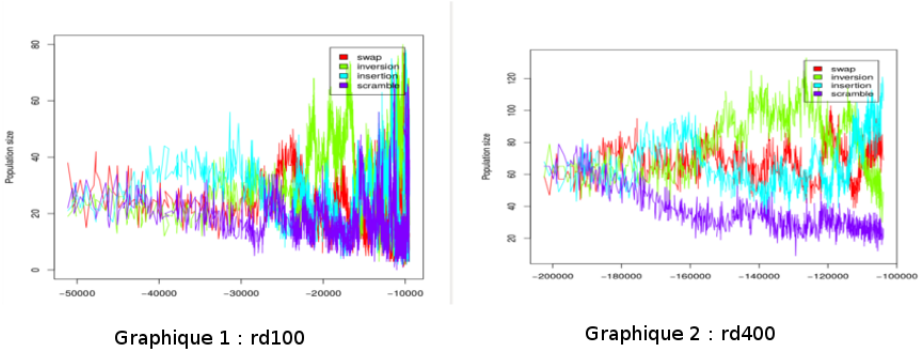

Figure 9. Deux instances aléatoires

Références

1. Eiben A.E., Smith J.E., **Introduction to Evolutionary Computing**, Second Edition, Springer, (2015).
2. Goëffon, A., Lardeux, F., Saubion, F. : **Simulating non-stationary operators in search algorithms**. Applied Soft Computing 38, 257-268, Elsevier, (2016)
3. Kuleshov V., Precup D., **Algorithms for the multi-armed bandit problem**, Journal of Machine Learning Research 1, 1-48 (2000).
4. Candan C., Goëffon, A., Lardeux, F., Saubion, F. : **A Dynamic Island Model for Adaptive Operator Selection**. Proceedings of the 14th annual conference on Genetic and evolutionary computation, Pages 1253-1260, (2012).
5. Reinelt G., **TSPLIB : A Traveling Salesman Problem Library** , ORSA Journal on Computing 3 (1991), pp. 376-384. (http ://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html)