Master 1 Informatique 2009/2010
TPE – Initiation à la recherche

Politiques de sélection dans les algorithmes génétiques
-
Application au problème du voyageur de commerce

par
Stéphane ESSAYIE

Rapport présenté et soutenu
le 25 juin 2010

Encadrant :
Moustafa NAKECHBANDI

A l'entame de ce rapport, je souhaite remercier M. Damien Olivier et M. Eric Sanlaville pour m'avoir ouvert les portes de cette filière malgré les difficultés qui m'attendaient, liées en outre à mon activité professionnelle.

Je souhaite également remercier M. Moustafa Nakechbandi pour son précieux encadrement. J'avais déjà eu l'occasion de croiser Moustafa en stage de DUT, à une époque où je me posais d'importantes questions sur une éventuelle poursuite d'études. Le retrouver six années plus tard dans le cadre d'un Master relève alors d'une amusante symbolique.

Et cette pensée en amenant une autre, je tiens à témoigner ma sympathie à mes anciens collègues de la société ExxonMobil qui à cette même époque m'ont tous encouragé à continuer mon cursus, avec un égard particulier à M. Cees-Jan De Wolf.

J'aimerais aussi signifier ma gratitude à deux personnes, M. Laurent Lechat et M. Denis Hébert, qui ont décidé il y a trois ans de m'embaucher dans leur entreprise alors qu'ils savaient parfaitement que je suivais des études et que cela engendrerait toutes les contraintes de fatigue et de stress que l'on sait. Merci à tous les deux pour la confiance et les encouragements que vous me témoignez chaque jour.

SOMMAIRE

Chapitre 1. Préambule

Le problème du voyageur commerce est un problème d'optimisation de la famille des NP-complets, reconnu comme un grand classique de la recherche scientifique. De très nombreuses applications logistiques concrètes dérivent de ce problème, d'où le franc succès qu'on lui connaît.

Si des algorithmes simples de résolution existent, ils n'en demeurent pas moins problématiques dès que la dimension du problème croît, la complexité de ces algorithmes étant telle que l'on obtient très rapidement des temps de calcul infinitésimaux.

De nombreux travaux ont donc été réalisés afin d'élaborer des algorithmes d'approximation permettant de retourner une solution proche de l'optimum dans des temps de calcul beaucoup plus raisonnables.

Parmi les diverses études menées jusqu'à aujourd'hui, on distingue les algorithmes génétiques dont le principe semble être en bonne adéquation avec le problème. Un projet a donc été réalisé en 2006 dans le cadre d'un Master à l'université du Havre[1] dans le but d'implémenter un algorithme génétique pour la résolution du problème. Mais à l'issue de ce projet, les performances de l'algorithme n'étaient pas celles qui étaient attendues.

Le projet ici proposé a donc pour objet une analyse de l'algorithme mis en place, accompagnée d'un état de l'art sur le problème du voyageur de commerce et les différentes méthodes de résolution élaborées à ce jour. A partir de cette analyse, une amélioration de cet algorithme est proposée et testée dans le but d'obtenir les performances attendues.

La principale question soulevée dans ce travail est celle de la stratégie globale à appliquer pour l'évolution de la population. La stratégie adoptée jusqu'à présent dans ce projet est celle d'une évaluation d'un individu par la seule distance totale du chemin représenté.

Nous aborderons dans ce papier des évaluations alternatives à cette méthode d'évaluation et nous procèderons à une étude comparative sur les différentes méthodes mises en œuvre.

[1] **Y.Liu**, *Application des algorithmes génétiques au problème du voyageur de commerce*, Rapport de mémoire de Master, Université du Havre (France), 2006

Chapitre 2. Analyse de l'existant

2.1. Le problème du voyageur de commerce

2.1.1. Généralités

Un voyageur de commerce doit visiter *n* villes données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Les distances entre les villes étant connues, la question que l'on se pose est de savoir quel chemin il faut choisir pour minimiser la distance totale parcourue.

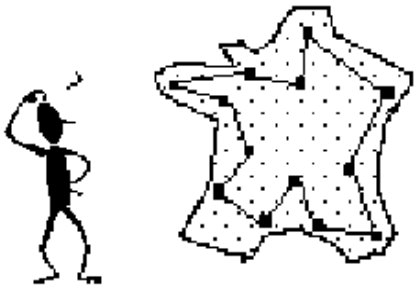


Figure 1 : Le problème du voyageur de commerce

De très nombreuses applications logistiques concrètes dérivent plus ou moins directement de ce problème. L'établissement de lignes de bus et métro d'une ville, le partage du champ d'action d'une société de livraison en plusieurs secteurs[2], l'établissement de missions pour les chariots cavaliers dans le cadre du rangement d'un terminal à conteneurs[3], etc.

Si ce problème paraît simple en apparence, la réalité mathématique est tout autre. En effet, le problème du voyageur de commerce est un problème dit NP-complet. Cela signifie qu'il est possible de **vérifier** une solution en un temps polynomial mais que le temps nécessaire pour **trouver** la solution est exponentiel.

[2] appelé « problème de tournées de véhicules ». Varie du PVC dans la mesure où il s'agit de fragmenter en plusieurs parcours partant tous du même point, l'ensemble de tous les parcours couvrant toutes les villes.
[3] Varie du PVC dans la mesure où une étape constitue le passage par deux points, et non un.

Dans notre cas, pour un nombre n de villes, il existe $n!$ parcours existants. Par exemple, pour un problème à 69 villes, le nombre de chemins possibles est un nombre à 100 chiffres. Pour comparaison, un nombre d'une longueur de 80 chiffres permettrait déjà de représenter le nombre d'atomes dans tout l'univers connu.

D'un point de vue plus formel, si on nomme n le nombre de villes du problème, alors on considère un graphe pondéré complet graphe $G = (V,A,\Omega)$ d'ordre n défini par :

- $V = \{x_0, x_1, ..., x_{n-1}\}$ est l'ensemble des nœuds.
- $A = \{x_i x_j\} \in V^2$ est l'ensemble des arcs.
- $\Omega = (\omega_{i,j})i, j \in [0, n-1]; \omega_{i,j} \in R$ est une matrice de coûts sur les arcs.

Ici, on ignore le cas où $i = j$[4]. Chaque élément $\omega_{i,j}$ de la matrice Ω représente le coût de l'arc $x_i x_j$. Pour représenter une solution quelconque au problème, on définit x_π un cycle hamiltonien quelconque dans G par :

- $x_\pi = (x_{\pi(0)}, x_{\pi(1)}, ..., x_{\pi(n-1)})$ où π est une permutation sur {0,1,…,n-1}

Le problème du voyageur de commerce consiste donc à chercher le cycle hamiltonien x_π le plus court possible dans G.

2.1.2. Etat de l'art

Plusieurs méthodes déterministes de résolution ont été développées. Les méthodes *« cutting plane »* et *« facet finding »* permettent entre autres de déterminer la solution optimale dans des temps beaucoup plus raisonnables que l'algorithme de parcours de base. Mais ces méthodes étant toujours d'une complexité de l'ordre du factoriel, les performances se dégradent toujours très rapidement lorsque l'on augmente la dimension du problème.

A titre d'exemple, en 1987, une méthode déterministe a été élaborée par Manfred Padberg et Giovanni Rinaldi avec pour performances l'établissement d'une solution optimale en 27 heures pour un problème à 2392 villes.

Afin de compenser ce problème, des algorithmes d'approximation par une approche heuristique ont été développés. Ceux-ci présentent une complexité beaucoup moins élevée et permettent de déterminer une solution dont le coût est proche de celui de la solution optimale. L'avantage est de réduire considérablement les temps de calcul, les applications concrètes du problème pouvant la plupart du temps se contenter d'une solution approchée de bonne qualité.

[4] Les chemins partant d'une ville et menant à cette même ville ne nous intéressent pas et ne sont même pas représentés.

Parmi les heuristiques les plus connues, on trouve la méthode des plus proches voisins, de l'élastique, du recuit simulé, l'algorithme de Lin-Kerninghan (généralisation adaptive de 2-Opt et 3-Opt), les réseaux de neurones auto adaptifs, l'optimisation par colonies de fourmis, etc. Les divers travaux effectués jusqu'alors révèlent que les algorithmes génétiques sont particulièrement adaptés à ce problème.

2.2. Algorithmes génétiques

L'algorithme génétique est une méta heuristique d'optimisation inventée par John Holland dans les années 1960 et reprise par David E. Goldberg dans les années 1970. Cette méthode est intégralement inspirée de la théorie sur la sélection naturelle et l'évolution des espèces de Charles Darwin. La technique des algorithmes génétiques est essentiellement utilisée dans la résolution de problèmes complexes nécessitant des temps de calcul élevés.

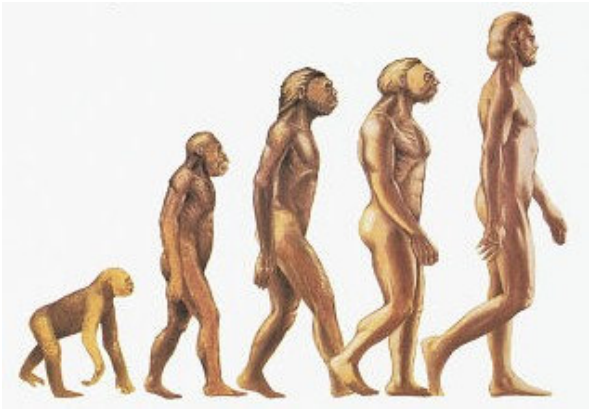


Figure 2 : L'évolution de l'Homme selon la théorie de Darwin

Le principe général de l'algorithme consiste à modéliser une solution quelconque au problème donné (même mauvaise) sous la forme d'une séquence spécifique de données que l'on appelle gène. Ce gène est la représentation exacte simplifiée d'un objet appelé individu, modélisant une solution.

On génère alors une population complète d'individus que l'on cherche à faire évoluer vers l'optimum global du problème selon un processus d'évolution défini. On évalue chaque individu à l'aide d'une fonction objectif[5], on sélectionne un certain nombre d'individus à l'aide de cette même fonction, on mélange leurs gènes via un opérateur de croisement afin de créer une nouvelle génération d'individus que l'on réinsère à la population en fonction de leur qualité (fonction objectif).

[5] C'est cette fonction dont on cherche l'optimum global

Afin de réduire le risque de tomber dans un optimum local et de voir la population stopper son évolution, on dispose également d'un opérateur de mutation qui consiste à modifier aléatoirement un individu quelconque selon une certaine probabilité. Cet opérateur est destiné à perturber le déroulement du processus afin de conserver non nulle une probabilité de ressortir d'un optimum local (à la manière d'un recuit simulé).

```
DEBUT

Création de la population g0
Evaluation des individus

TantQue condition d'arrêt=faux Faire
      Sélection des meilleurs individus
      Croisement
      Mutation
      Insertion dans la nouvelle génération gi
      Evaluation
FinTantQue

FIN
```

Figure 3 : Le processus général de l'algorithme génétique

Les principaux domaines d'application des algorithmes génétiques sont le traitement d'image (alignement de photos satellite, reconnaissance de suspects), optimisation d'emploi du temps, optimisation de design, apprentissage des réseaux de neurones.

2.3. Analyse de l'algorithme et de son paramétrage

Le projet développé en 2006 a été réalisé de manière à pouvoir saisir soi même le plus grand nombre possible de paramètres (choix de l'opérateur de croisement, de la probabilité de croisement, probabilité de mutation, etc.). Une étude statistique a donc pu être réalisée dans le même temps, afin de mettre en évidence de manière empirique le meilleur paramétrage possible de l'algorithme.

L'étude s'est réalisée sur le même *« échantillon »* de problème (même nombre de villes et même disposition). Les moyennes calculées portent sur un ensemble de 20 tests par type de paramétrage et sur un problème à 20 villes. Les deux critères de qualité de l'algorithme utilisés dans cette étude sont la distance moyenne trouvée que l'on cherche à faire diminuer et l'écart type des solutions trouvées que l'on cherche également à faire diminuer.

Nombre de villes	Opérateur de croisement	Probabilité de croisement	Type de sélection	Probabilité de mutation	Distance moyenne	Ecart type
20	CPA	**100%**	Roulette	0.1	2253	121.4
20	CPA	**60%**	Roulette	0.1	2136	99.5
20	CPA	100%	Roulette	**0.02**	2248	151.0
20	CPA	100%	Roulette	**0.2**	2244	102.2
20	**1X**	100%	Roulette	0.1	2159	101.9
20	**OX**	100%	Roulette	0.1	2068	36.7
20	CPA	100%	**Rang**	0.1	2289	133.0
20	CPA	100%	**Tournoi**	0.1	2322	193.2

Figure 4 : Résultats de l'étude de 2006 sur le paramétrage de l'algorithme

On constate que si on diminue la probabilité de croisement, la distance moyenne et l'écart type sont sensiblement améliorés. La modification de la probabilité de mutation n'impacte presque pas la distance moyenne mais par contre, plus elle augmente, plus l'écart type est bas. Les croisements 1X et 0X améliorent la distance moyenne et l'écart type. La sélection par rang et la sélection par tournoi dégradent la distance moyenne et l'écart type.

Cette étude nous a permis de conclure que le meilleur paramétrage connu est un croisement OX avec probabilité de croisement de 60%, une sélection par roulette, et une probabilité de mutation de 0.1. C'est ce paramétrage que nous utiliserons lors de nos tests.

Chapitre 3. Amélioration de l'algorithme

Après une première analyse, on peut constater que l'algorithme élaboré en 2006 possède des failles, principalement dans les choix de programmation pas forcément optimaux de ses différentes composantes telles que la mutation, la réinsertion des individus, ou encore la création de la population initiale.

3.1. Génération de la population initiale

Le problème de la création de la population initiale est une question fondamentale. C'est en effet ainsi que l'on détermine l'ensemble du matériel génétique qui sera utilisé pour faire évoluer notre population.

La question qui se pose est celle d'une éventuelle optimisation sur les individus de départ. Faut-il partir d'une population totalement aléatoire dont la probabilité de n'avoir que des mauvais[6] individus est forte, ou faut-il partir d'un ensemble d'individus déjà jugés intéressants ? Les avis divergent sur la question. Beaucoup appliquent un premier algorithme de type *« 2-opt »* ou bien *« plus proches voisins »* sur leurs individus initiaux afin de ne travailler que sur des individus intéressant.

Ici, nous partirons du principe qu'il n'est nullement besoin d'une quelconque optimisation sur nos individus de départ si par ailleurs l'algorithme génétique est correctement paramétré. La convergence est en effet suffisamment forte pour rapidement obtenir des individus intéressants.

De plus, l'un des concepts les plus importants dans un algorithme génétique est celui de la diversité des individus composant une population. En ce sens, appliquer un seul et même algorithme d'optimisation sur des individus (même générés aléatoirement dans un premier temps) paraît être une erreur. C'est pourquoi nous nous contenterons ici d'une génération totalement aléatoire.

3.2. Réinsertion des individus

Le processus de réinsertion des individus a également été revu. En effet, à chaque génération, on insère les k nouveaux individus créés, puis on supprime les k plus mauvais individus de la population jusqu'à redescendre à une taille de n. Cette approche semble incorrecte car induisant une notion d'élitisme quelle que soit la méthode de sélection choisie,

[6] Mauvais en termes d'évaluation par la fonction objectif

et augmentant donc la probabilité de tomber dans un optimum local. Alors, très rapidement, au bout de quelques générations, la méthode de sélection choisie n'a plus le moindre effet. C'est pourquoi on se propose de mettre en place une méthode plus *« légère »* qui consiste à déterminer, à l'insertion d'un individu, tous les individus plus mauvais que lui et à en supprimer un au hasard parmi ceux-ci.

3.3. Utilisation de l'opérateur de croisement KFP

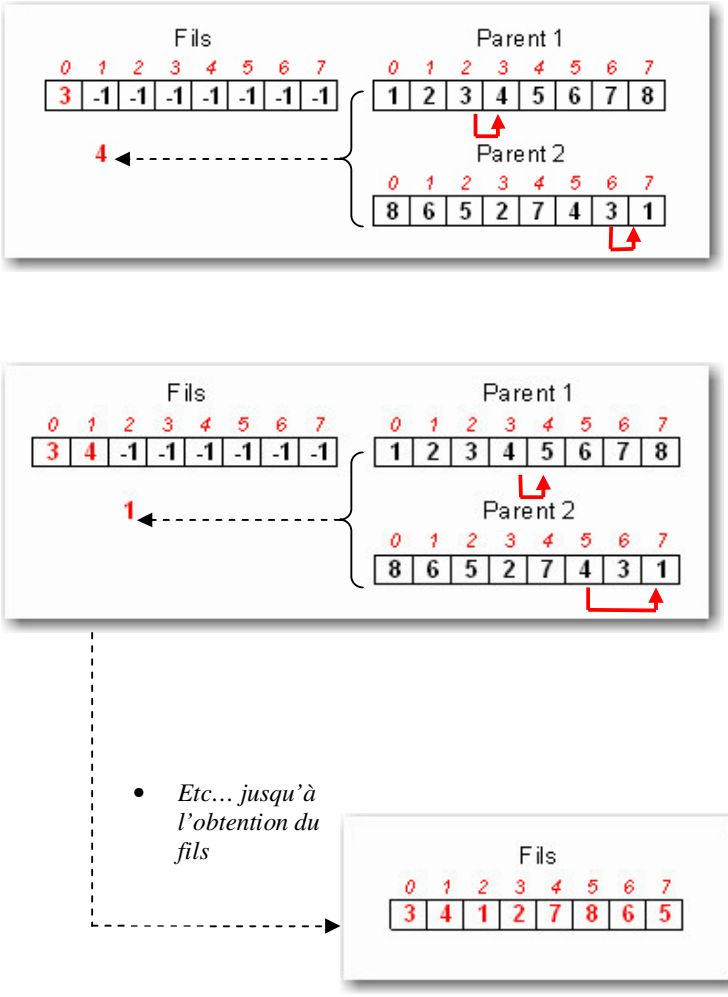
Le croisement KFP est un opérateur de croisement dans lequel on a introduit une heuristique basée sur un principe similaire à celui d'une sélection par roulette. C'est un opérateur de croisement spécifique au problème du voyageur de commerce.

Elaboré par Karoly F. Pal en 1993, cet opérateur crée un individu fils en commençant par une ville choisie au hasard, puis en incluant les villes suivantes selon une probabilité définie en fonction de la distance entre cette ville et la ville suivante dans chacun des deux parents.

```
c : Ville courante
a : Position de la ville suivante dans parent1
b : Position de la ville suivante dans parent2
pA : Probabilité
pB : Probabilité

DEBUT

fils[0] <- c

Pour i allant de 1 à n Faire

        a <- position(c,parent1)+1
        b <- position(c,parent2)+1
        TantQue parent1[a]∈ fils Faire a <- (a+1)%n
        TantQue parent2[b]∈ fils Faire b <- (b+1)%n
        pA <- ω c,parent1[a]
        pB <- ω c,parent2[b]

        Si pA > pB Alors fils[i] <- parent1[a]
        Sinon fils[i] <- parent2[b]

FinPour

FIN
```

Figure 5 : Processus du croisement par l'opérateur KFP

L'avantage de cette technique réside dans le fait qu'elle produit en moyenne des individus de bonne qualité assez rapidement, tout en introduisant un facteur de perturbation au sein même se l'opérateur de croisement, ce qui vient s'ajouter à l'effet de l'opérateur de mutation.



- r=3 choisi aléatoirement. On insère 3 au fils.
- Calcul p1 & p2 en fonction de $\omega_{3,4}$ et $\omega_{3,1}$
- 4 est *« préféré »* à 1. On insère 4 au fils
- *On réitère à partir de r=4.*
- *Attention, dans le parent 2, 3 existe déjà dans le fils, on prend donc la ville suivante.*
- *Etc… jusqu'à l'obtention du fils*

Figure 6 : Exemple de croisement KFP sur un gène du PVC

3.4. Amélioration de l'opérateur de mutation

Afin d'éviter de tomber dans des extrema locaux, un opérateur de mutation est utilisé dans les algorithmes génétiques. Celui-ci permet d'induire un critère de perturbation dans le processus. Or ici, il est utilisé dans une méthode où on ne conserve pas l'individu muté s'il est

plus faible que l'individu original, ce qui annihile en grande partie, si ce n'est totalement, son rôle originel de perturbateur. Il paraît donc évident et nécessaire de conserver l'individu muté quelle que soit sa qualité.

Par ailleurs, on procède ici à une mutation standard appelée mutation *« Twors »* qui consiste en une simple permutation de deux positions de gènes choisies aléatoirement. Or, les différentes études montrent que cet opérateur de mutation n'est pas le plus adapté au problème du voyageur de commerce.

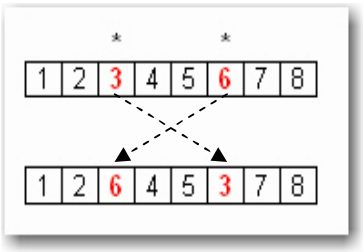


Figure 7 : L'opérateur de mutation "Twors"

On propose donc d'utiliser une mutation plus efficace, la mutation par renversement (opérateur *« Inverse »*). On détermine à partir de deux positions choisies aléatoirement une séquence complète de gènes, et on inverse complètement celle-ci.

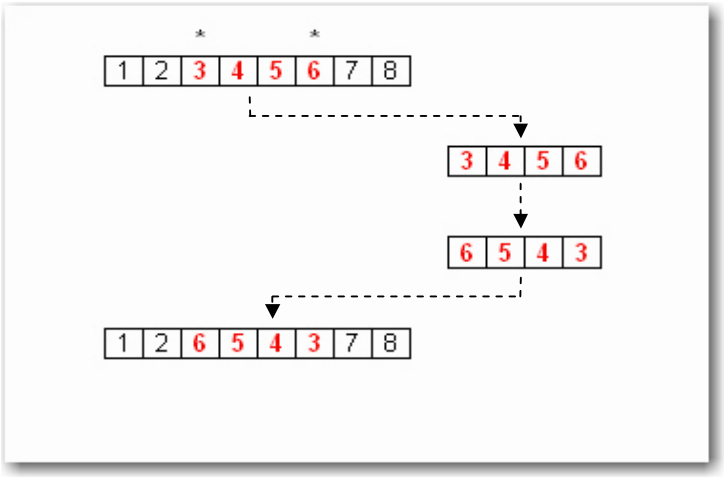
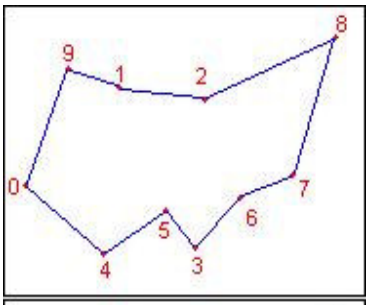


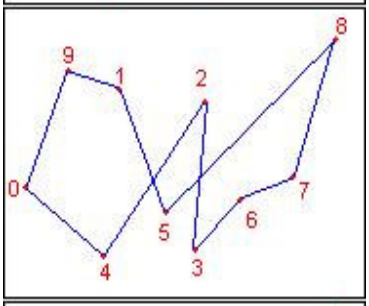
Figure 8 : L'opérateur de mutation "Inverse"

L'intérêt de cette mutation réside dans le fait que la perturbation engendrée est différente et permet à l'algorithme de plus facilement rebondir vers de nouveaux individus

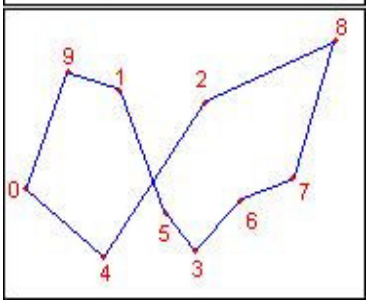
intéressants. Afin de mieux visualiser l'effet produit, voici ce que donnent les deux types de mutation sur un individu donné en exemple :



- Individu d'origine :
- 0453678219



- Individu muté avec l'opérateur *« Twors »*. On échange 5 et 2 :
- 04**2**3678**5**19
- On constate la perturbation sur les arcs 15, 58, 42, et 23



- Individu muté avec l'opérateur *« Inverse »*. On renverse toute la séquence allant de 5 à 2 :
- 04**28763**519
- On constate la perturbation seulement sur les arcs 15 et 42

Figure 9 : Exemple d'amélioration apportée par l'opérateur de mutation "Inverse"

Chapitre 4. Stratégies d'évaluation pour le processus d'évolution

4.1. Introduction

La principale problématique soulevée dans cette étude est celle des critères de choix des individus à croiser. Rappelons le contexte formel du problème du voyageur de commerce et l'objectif à atteindre pour sa résolution. Soit un graphe pondéré complet $G = (V,A,\Omega)$ d'ordre n (n étant le nombre de villes) défini par :

- $V = \{x_0, x_1, ..., x_{n-1}\}$ Ensemble des nœuds.
- $A = \{x_i x_j\} \in V^2$ Ensemble des arcs.
- $\Omega = (\omega_{i,j}) i, j \in [0, n-1]; \omega_{i,j} \in R$ Matrice de coût sur les arcs.

Si on considère un cycle hamiltonien quelconque dans notre graphe G par $x_\pi = (x_{\pi(0)}, x_{\pi(1)}, ..., x_{\pi(n-1)})$ où π est une permutation sur $\{0,1,\ldots,n\text{-}1\}$, alors le problème posé consiste à rechercher le cycle x_π le plus court possible.

La question que l'on se pose dans cette étude est de savoir ce qui va déterminer la qualité d'un cycle x_π. Rappelons que dans notre algorithme génétique, un individu est un cycle hamiltonien de G. Plusieurs stratégies d'évaluation pour la sélection ont donc été testées et comparées.

4.2. Evaluation par calcul de la somme des coûts

C'est la méthode la plus couramment utilisée pour la résolution du problème du voyageur de commerce. On estime que la qualité d'un individu x_π est directement déterminée par la somme des coûte de ses arcs. On définit alors δ la fonction d'évaluation associée à cette stratégie d'évaluation par :

$$\delta(x_\pi) = \left(\sum_{i=0}^{n-2} \omega_{\pi(i),\pi(i+1)} \right) + \omega_{\pi(n-1),\pi(0)}$$

Dans cette première stratégie, on utilise $\delta(x_\pi)$ à la fois à l'évaluation pour sélection et à la réinsertion des individus créés. Plus généralement, on utilisera systématiquement $\delta(x_\pi)$ pour la réinsertion, celle-ci restant la fonction objectif à minimiser.

L'inconvénient de cette méthode d'évaluation est qu'une simple somme a tendance à masquer certains mauvais arcs d'un individu si ses autres arcs sont bons.

- Le coût du mauvais arc est masqué par les coûts des bons arcs dans le calcul de la somme.
- Cet individu sera donc jugé comme assez bon.

Figure 10 : Exemple d'individu remettant en question l'évaluation sur la somme des coûts

4.3. Evaluation par « entropie »

La méthode de l'entropie croisée est une méthode générique d'optimisation. Elaborée en 1999 par Reuven Rubinstein, cette méthode peut être appliquée à des problèmes d'optimisation combinatoires telles que le problème du voyageur de commerce. Son principe général est le suivant :

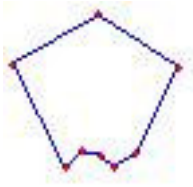
1. On choisit un vecteur initial $v^{(0)}$ et on pose $t=1$.
2. On génère un échantillon de variables aléatoires $X_1, X_2, ..., X_N$ selon la densité $f(.;v^{(t-1)})$.
3. On calcule $v^{(t)}$ où $v^{(t)} = \arg\max_v \frac{1}{N}\sum_{i=1}^{N} H(X_i)\frac{f(X_i;u)}{f(X_i;v^{(t-1)})}\log[f(X_i;u)]$ avec H qui est la fonction objectif et $f(x;u)$ qui est une densité de probabilité paramétrique.
4. Si l'algorithme a convergé, on arrête, sinon on incrémente *t* et on répète depuis l'étape 2.

Mais cette méthode n'est pas basée sur les algorithmes génétiques. Elle n'est donc pas prise en considération en tant que telle. On s'en inspire indirectement dans la méthode ici proposée en calculant un écart type de l'ensemble des coûts des arcs de notre parcours x_π. On définit donc la fonction φ par :

$$\varphi(x_\pi) = \frac{1}{n}\left[\left(\sum_{i=0}^{n-2}(\omega_{\pi(i),\pi(i+1)} - \overline{\omega})^2\right) + (\omega_{\pi(n-1),\pi(0)} - \overline{\omega})^2\right] \text{ avec } \overline{\omega} = \frac{1}{n}\delta(x_\pi)$$

Ici, la réinsertion des individus créés se fera donc par $\delta(x_\pi)$ et la sélection des individus se fera en fonction de $\varphi(x_\pi)$.

Cette méthode permet de mettre en évidence les mauvais arcs cachés par un simple calcul de la somme des coûts. Néanmoins, elle pose un problème dans la mesure où elle va naïvement juger un individu à l'écart type élevé comme mauvais même si celui-ci est l'optimum. Ce phénomène est atténué par la réinsertion sur $\delta(x_\pi)$ mais le principal problème est qu'un tel individu risque de ne pas être sélectionné pour évoluer bien qu'il soit conservé dans la population.



- Notre individu est l'optimum
- Mais il sera mal noté en raison de l'écart type trop élevé de ses arcs.

Figure 11 : Exemple d'individu remettant en question l'évaluation par "entropie"

4.4. Evaluation par écarts types partiels

Afin d'atténuer le phénomène de non sélection d'un bon individu en raison de son écart type élevé, on se propose d'élaborer une méthode similaire qui ne tient compte que d'une certaine partie des arcs dans notre calcul d'écart type. L'idée générale est de diviser les arcs de notre individu en catégories de longueur, puis de calculer l'écart type de chacune de ces catégories, et enfin de retourner une moyenne de ces écarts types.

Ici, on suggère de ne tenir compte que des arcs les plus courts et les arcs les plus longs. On reconsidère notre individu x_π cette fois comme une séquence d'arcs et non plus comme une séquence de nœuds. Si on considère π comme une permutation sur $\{0,1,\ldots,n\text{-}1\}$, alors $x_\pi = (x_{\pi(0)}, x_{\pi(1)}, \ldots, x_{\pi(n-1)})$ devient $x_\pi = (x_{\pi(0)}x_{\pi(1)}, x_{\pi(1)}x_{\pi(2)}, \ldots, x_{\pi(n-1)}x_{\pi(0)})$. On considère alors deux ensembles d'arcs A_{Min} et A_{Max} et une variable réelle k tels que :

- $k \in]0,0.5[$
- $A_{Min} \subset x_\pi$ est l'ensemble des $E(k \times n)$ arcs les plus courts de x_π
- $A_{Max} \subset x_\pi$ est l'ensemble des $E(k \times n)$ arcs les plus longs de x_π

Par précaution, on préfèrera renseigner un k strictement inférieur à 0.5 afin de s'assurer que $A_{Min} \cap A_{Max} = \varnothing$.

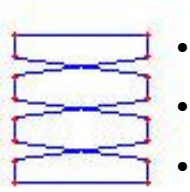
Par analogie avec la méthode d'évaluation par « entropie », on définit les fonctions δ_1, δ_2, φ_1 et φ_2 telles que :

- $\delta_1(x_\pi) = \sum_{x_i x_j \in A_{Min}} \omega_{i,j}$ et $\varphi_1(x_\pi) = \frac{1}{E(k \times n)} \sum_{x_i x_j \in A_{Min}} (\omega_{i,j} - \overline{\omega_1})^2$ avec $\overline{\omega} = \frac{1}{E(k \times n)} \delta_1(x_\pi)$

- $\delta_2(x_\pi) = \sum_{x_i x_j \in A_{Max}} \omega_{i,j}$ et $\varphi_2(x_\pi) = \frac{1}{E(k \times n)} \sum_{x_i x_j \in A_{Max}} (\omega_{i,j} - \overline{\omega_1})^2$ avec $\overline{\omega} = \frac{1}{E(k \times n)} \delta_2(x_\pi)$

On calcule alors notre fonction d'évaluation $\mu(x_\pi) = \frac{\varphi_1(x_\pi) + \varphi_2(x_\pi)}{2}$.

L'inconvénient de cette méthode est qu'on ne sait pas distinguer le cas où les arcs d'une même catégorie de longueur se suivent du cas où ils sont dispersés dans notre individu. Ainsi, un individu qui alterne systématiquement arc court et arc long sera jugé bon alors qu'en réalité on n'en sait rien.



- Les arcs le plus courts ont un faible écart type.
- Les arcs les plus longs ont un faible écart type.
- Mais l'individu n'est pas intéressant

Figure 12 : Exemple d'individu remettant en question l'évaluation par écarts types partiels

4.5. Evaluation par partition de la population

Toutes les méthodes que nous avons mises en œuvre jusqu'ici sont basées sur un seul et même principe. Quelle que soit l'évaluation choisie, on détermine des critères de qualité pour un individu donné, puis on sélectionne les individus jugés de bonne qualité pour procéder à un croisement. Mais cela soulève une question essentielle : est-ce que deux individus jugés chacun de bonne qualité sont forcément intéressants à croiser ?

Intuitivement, on peut penser que non. En effet, si on considère un individu *A* de bonne qualité mais pas optimale, et un individu *B* différent de *A* mais également de bonne qualité, rien ne nous garantit que *B* va posséder le gène manquant à *A* pour atteindre l'optimalité, et vice versa.

Pire encore, au croisement, le gène de l'un peut dégrader la partie intéressante du gène de l'autre, auquel cas le croisement effectué s'avère totalement inutile. D'ailleurs, lors de l'élaboration de la méthode par écarts types partiels, il a d'abord été testé une évaluation sur

les $E(k \times n)$ arcs les plus longs, puis sur les $E(k \times n)$ arcs les plus courts avant d'envisager une moyenne des deux. Il a alors été constaté qu'une évaluation sur l'écart type des arcs les plus courts avait tendance à défavoriser l'écart type des arcs les plus longs. Et il en est de même pour une évaluation sur l'écart type des arcs les plus longs qui a tendance à défavoriser l'écart type des arcs les plus courts.

En revanche, il semble intéressant de croiser un individu à faible écart type sur ses arcs les plus longs avec un individu à faible écart type sur ses arcs les plus courts. C'est ici la stratégie qu'on se propose d'adopter.

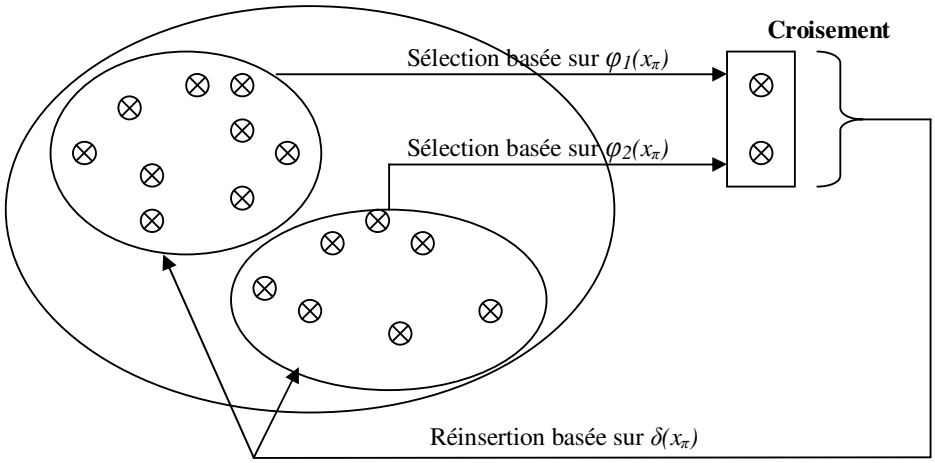


Figure 13 : Schéma du processus global d'évolution sur une population partitionnée

Chapitre 5. Analyse statistique

5.1. Echantillon de test

Afin de donner un sens statistique réel à notre étude, il convient évidemment de procéder à tous nos tests avec le même échantillon. Redisposer aléatoirement les villes à parcourir à chaque exécution, n'aurait en effet eu aucun sens.

Ainsi, nous travaillerons sur un problème à 50 villes dont les coordonnées générées une première fois aléatoirement ont ensuite été enregistrées et réutilisées. Cet échantillon a été testé un grand nombre de fois avec plusieurs algorithmes différents. Sans pour autant avoir la preuve qu'il s'agisse bien de l'optimum global, nous comparerons nos résultats avec le meilleur chemin connu à ce jour pour ce parcours dont la longueur est de 2985.

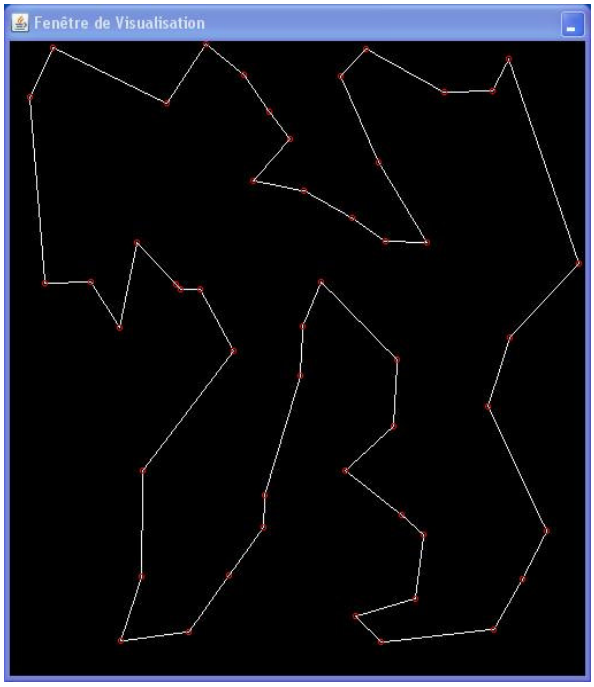


Figure 14 : L'échantillon de test et le meilleur chemin connu sur celui-ci

De cet échantillon, nous savons également qu'il existe au moins 23 parcours différents dont la longueur est de moins de 2% plus élevée que le meilleur connu. Cela signifie donc que nous avons ici un grand nombre d'optima locaux, ce qui rend notre échantillon intéressant et significatif.

Par ailleurs, notre étude ne porte que sur trois paramètres à faire varier : le choix de l'opérateur de croisement[7], le choix de l'opérateur de mutation[8] et le choix de la méthode d'évaluation. En ce qui concerne l'ensemble des autres paramètres de l'algorithme, ils resteront fixes sur l'ensemble de l'étude avec les valeurs suivantes (valeurs optimales déterminées en 2006) :

- Une probabilité de croisement à 60%
- Une probabilité de mutation de 0.2
- Une sélection par roulette
- Une population de 100 individus

5.2. Indicateurs utilisés

Chaque cas de test fera l'objet de 20 exécutions successives de l'algorithme sur lesquelles, en fonction du résultat obtenu, nous calculerons les indicateurs suivants :

5.2.1. Distance moyenne

On calcule tout simplement une moyenne des 20 résultats retournés. Cet indicateur nous informe directement sur l'ordre de grandeur de la qualité de la solution retournée. Plus la moyenne est basse, plus l'algorithme retourne des solutions de bonne qualité.

Parallèlement à cela, on met aussi en évidence le meilleur résultat retourné sur les 20 tests, ainsi que le plus mauvais résultat retourné sur ces mêmes 20 tests.

5.2.2. Ecart type

L'un des inconvénients des méta heuristiques d'exploration réside dans le degré d'imprévisibilité de la solution retournée. Un même algorithme exécuté deux fois de suite dans les mêmes conditions initiales peut donner deux solutions d'un degré d'optimalité différent, ou encore deux solutions quasi optimales mais complètement différentes dans leur forme.

[7] On compare l'opérateur OX avec l'opérateur KFP
[8] On compare l'opérateur *« Twors »* avec l'opérateur *« Inverse »*

Afin de mesurer ce phénomène dans une étude statistique, il convient donc de calculer l'écart type des solutions retournées. Plus celui-ci sera bas, plus cela voudra dire que l'algorithme sera fiable en termes de reproductibilité de la solution trouvée[9].

5.2.3. Ecart au meilleur résultat connu

Basé sur la moyenne des 20 tests et sur le meilleur résultat connu, on détermine la *« distance »* moyenne restant pour atteindre l'optimum global[10] par :

$$d_{Opt} = \frac{Moyenne - Optimum}{Optimum}$$

5.3. Etude de l'amélioration apportée par KFP et IM

Algorithme	Opérateur de croisement	Opérateur de mutation	Meilleur parcours	Plus mauvais parcours	Moyenne	Ecart type	Ecart au meilleur connu
2006	OX	Twors	3850	5172	4251.45	372.14	42.43%
2010	OX	**Twors**	3194	4176	3675.65	242.28	23.14%
2010	**OX**	**Inverse**	3054	3356	3201.80	82.71	7.26%
2010	**KFP**	Inverse	2985	3098	3043.15	34.00	1.95%

Figure 15 : Résultats de l'étude de l'amélioration de l'algorithme

[9] Uniquement en termes de distance totale. L'écart type ne pourra pas quantifier la différence entre deux solutions de même distance totale mais de parcours différents.
[10] On rappelle qu'on n'a pas la preuve qu'il s'agisse bien de l'optimum global.

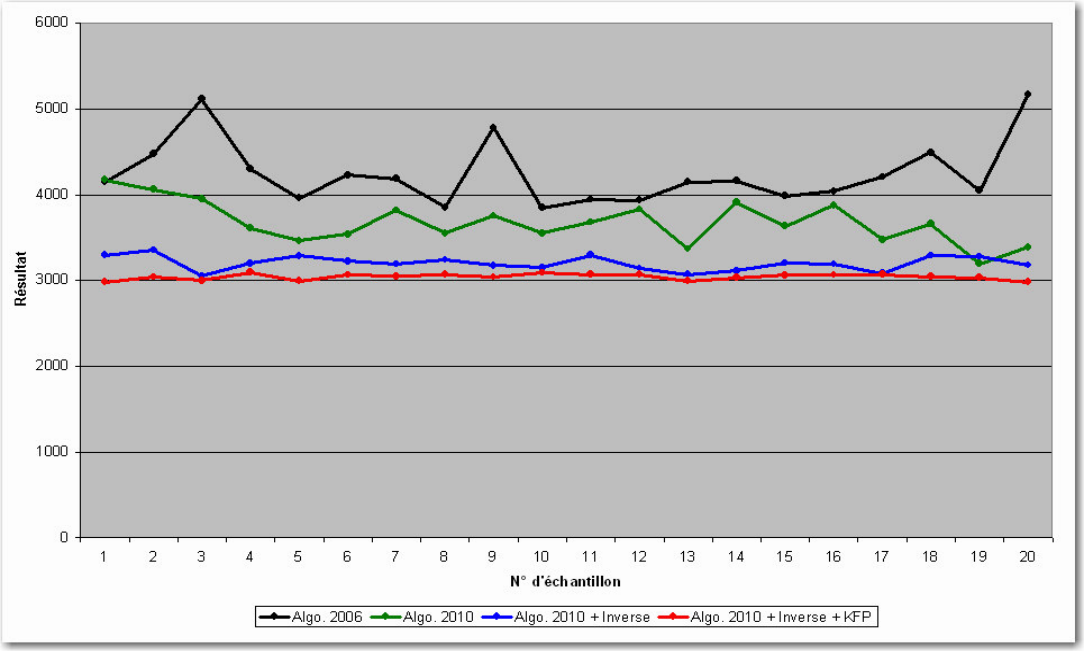


Figure 16 : Détail des tests d'amélioration de l'algorithme

5.4. Etude de la mise en œuvre des différentes stratégies d'évaluation

Algorithme	Méthode d'évaluation	Meilleur parcours	Plus mauvais parcours	Distance moyenne	Ecart type	Ecart au meilleur connu
2006[11]	Somme des coûts	3850	5172	4251.45	372.14	42.43%
2010	**Somme des coûts**	2985	3098	3043.15	34.00	1.95%
2010	**« Entropie »**	2985	3088	3043.55	30.82	1.96%
2010	**Ecarts types partiels**	2995	3182	3055.90	42.50	2.38%
2010	**Partition population**	2995	3201	3046.35	49.32	2.06%

Figure 17 : Résultats de l'étude sur les stratégies d'évaluation

[11] Avec un croisement OX et une mutation *« Twors »*

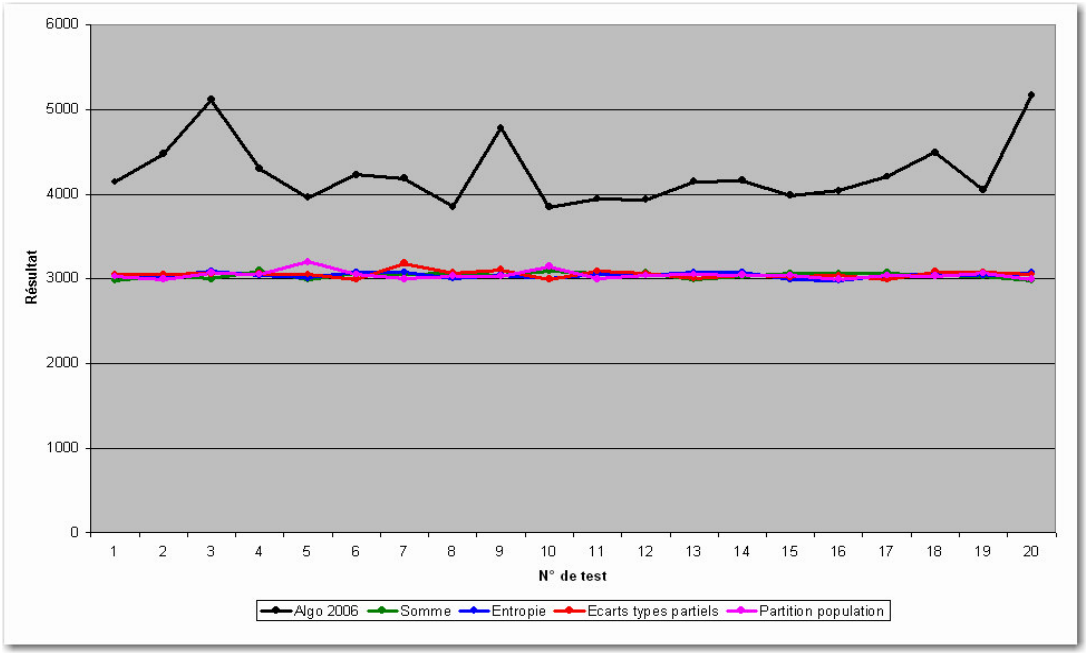


Figure 18 : Détail des tests sur les stratégies d'évaluation

Chapitre 6. Bilan & Perspectives

6.1. Bilan sur les améliorations de l'algorithme

Les tests réalisés sur l'étude d'amélioration de l'algorithme sont extrêmement concluants. L'apport de nouvelles méthodes de génération de la population initiale et de réinsertion des individus, ainsi que l'apport de l'opérateur de croisement KFP et de l'opérateur de mutation *« Inverse »* permettent en effet d'améliorer la longueur moyenne du parcours de 28% et l'écart type de 91%.

Néanmoins, l'algorithme reste assez largement perfectible puisque l'on est en moyenne à 1.95% de ce que l'on considère être l'optimum global, et on rappelle qu'on connaît au moins 23 parcours différents à moins de 2% de l'optimum global. D'ailleurs si on augmente la dimension du problème, on voit apparaître des optima locaux plus évidents, malgré que les résultats obtenus soient en moyenne de bonne qualité.

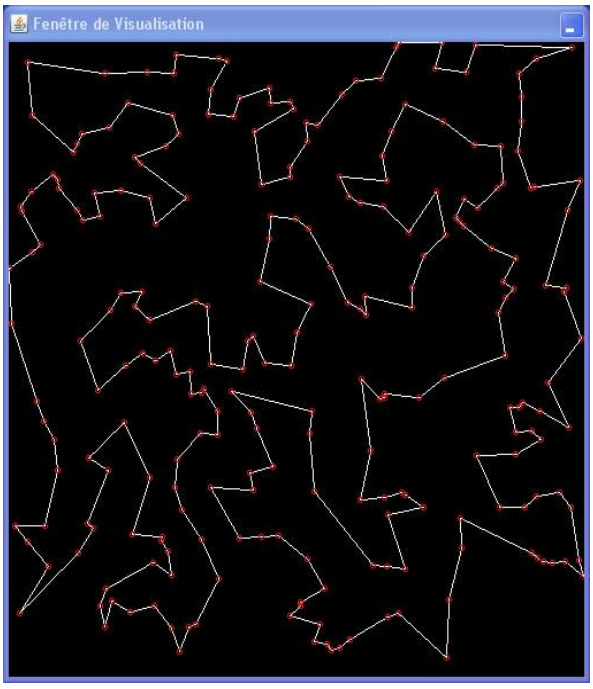


Figure 19 : Résultat d'exécution pour un problème à 250 villes

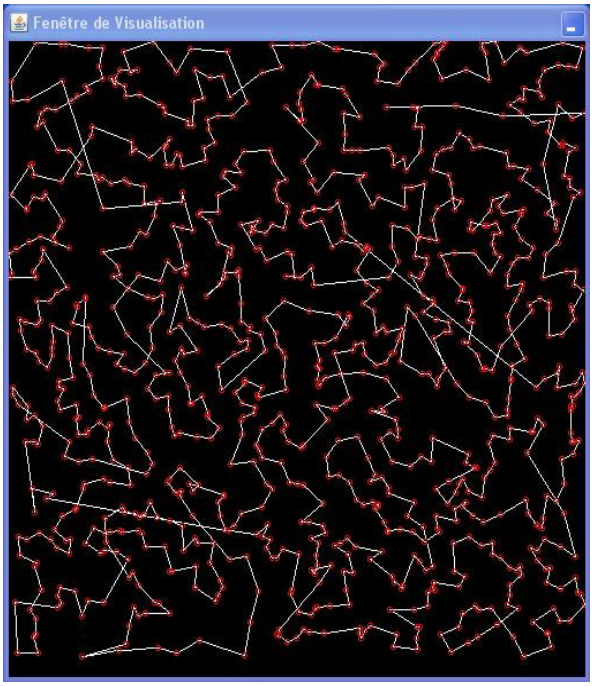


Figure 20 : Résultat d'exécution pour un problème à 1000 villes

6.2. Discussion sur les politiques de sélection mises en œuvre

L'étude des différentes méthodes d'évaluation mises en œuvre est beaucoup moins significative. Les comportements sont différents sur la forme mais les résultats sont tous semblables en termes de distance totale.

On constate notamment qu'on tombe systématiquement dans des optima locaux, ce qui est plus visible sur des problèmes à 250 et 1000 villes où on voit bien que les évolutions se font jusqu'à la fin de l'algorithme mais systématiquement vers une solution quasi identique à la précédente. On ne voit pratiquement pas la structure globale de la solution changer.

Deux hypothèses sont envisagées afin d'expliquer ce phénomène. La première d'entre elles concerne le degré de perturbation de l'algorithme. L'opérateur de mutation et le croisement KFP ne sont peut être pas suffisants pour permettre une continuité du potentiel évolutif de l'algorithme.

La seconde hypothèse concerne l'ensemble de la population. Les individus finissent peut être par tous être identiques ou très semblables. En d'autres termes, la population atteindrait son plateau d'évolution.

Il en ressort alors un autre critère fondamental pour l'exécution d'un algorithme génétique. S'il est important de bien savoir déterminer la qualité d'un individu, il est également primordial de savoir déterminer la qualité d'une population complète. La question que l'on se pose alors est de savoir quels critères déterminent la qualité d'une population. Il nous faut parvenir à mesurer sa capacité à faire évoluer ses individus. Si tous les individus se ressemblent, alors ils n'évolueront que très peu, ce qui signifie que la population est mauvaise (même si les individus sont bons).

Il nous faut donc réfléchir à une méthode permettant de maintenir une diversité de population élevée au fil des générations afin de ne pas voir diminuer le potentiel évolutif de l'algorithme au cours de son exécution. Par exemple, on pourrait imaginer ne conserver qu'un individu parmi ceux qui se ressemblent trop et remplacer les autres par de nouveaux individus générés aléatoirement.

Mais cette idée soulève deux problèmes. Comment modéliser notre population afin d'y retrouver facilement des groupes d'individus se ressemblant ? Car il est clair qu'un simple tableau ne suffit plus. Et surtout, comment quantifier la ressemblance entre deux individus ? Nous avons d'ores et déjà commencé à réfléchir sur une première approche qui consisterait à coder un individu différemment afin de pouvoir calculer une distance de Hamming entre deux individus :

$$x_\pi = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{ avec } x_{\pi_{i,j}} = \begin{cases} 1 \; si \; (ij) \in x_\pi \\ 0 \; si \; (ij) \notin x_\pi \end{cases}$$

Figure 21 : Exemple d'implémentation matricielle d'un individu

Mais cette première approche, bien que non implémentée ni testée, semble avoir quelques failles. Il semblerait en effet dans certains cas que deux individus proches en termes de distance de Hamming puissent être éloignés dans l'espace de recherche, et inversement.

TABLE DES ILLUSTRATIONS

REFERENCES BIBLIOGRAPHIQUES

- **Y.Liu**, *Application des algorithmes génétiques au problème du voyageur de commerce*, Rapport de mémoire de Master, Université du Havre (France), 2006

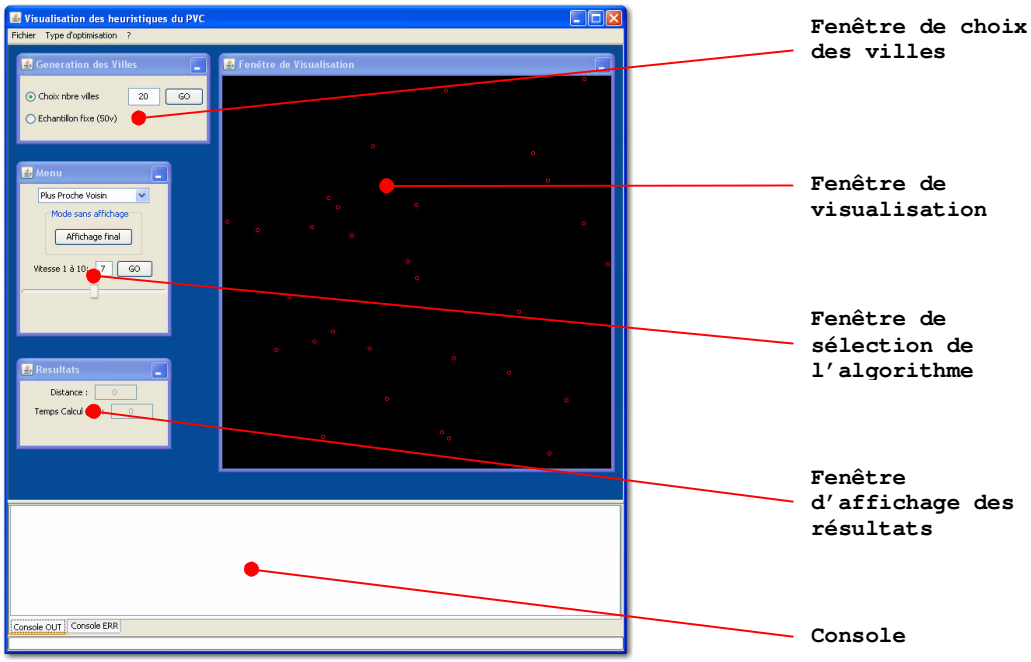
- **S.Bourazza**, *Variantes d'algorithmes génétiques appliquées aux problèmes d'ordonnancement*, Thèse de doctorat, Université du Havre (France), 2006

- **L.Margolin**, *On the convergence of the Cross-Entropy method, Faculty of Industrial Engineering and Management*, Technion, Haifa (Israel), 2004

- **K.F. Pál**, *Genetic algorithms for the travelling salesman problem based on a heuristic crossover operation*, Institute of Nuclear Research of the Hungarian Academy of Sciences, Debrecen (Hungary), 1993

ANNEXE : Manuel d'utilisation

L'objet de cette annexe est de présenter un mode opératoire pour l'utilisation de l'interface graphique du projet :



Pour démarrer un algorithme génétique :

1. Sélectionner les villes à l'aide de la **fenêtre de choix des villes**. Il est possible de disposer aléatoirement un nombre saisissable de villes ou bien de travailler avec l'échantillon fixe de test comportant 50 villes.

2. Sélectionner *« Algorithme génétique »* dans le menu déroulant de la **fenêtre de sélection de l'algorithme**.

3. Une nouvelle fenêtre apparaît. Y saisir les paramètres choisis pour notre algorithme génétique, ainsi que la politique de sélection testée. Cliquer sur *« OK »*.

4. Cliquer sur *« Go »* dans la **fenêtre de sélection de l'algorithme**.

5. L'algorithme se lance. On pourra constater les résultats dans la **fenêtre de visualisation**, la **fenêtre d'affichage des résultats** et la **console**.