Travail d'Étude
Algorithmes Génétiques

Cédric Dalmasso Didier Dalmasso

Juin 2004

Sous la direction de Philippe Audebaud

Remerciements

Nous tenons à remercier
Monsieur Philippe Audebaud, notre tuteur, pour sa critique constructive et sa patience.
Marjorie, notre sœur, pour sa relecture attentive.

Résumé

De nombreux problèmes n'ont pas de méthodes efficaces et exhaustives pour les résoudre. De nombreuses techniques ont été mises au point dans le but d'apporter une réponse adaptée à cette problématique, les algorithmes génétiques en font partie. Dans ce travail d'étude, nous nous intéresserons aux algorithmes génétiques. Nous verrons dans quel contexte les algorithmes génétiques ont été développés, depuis le domaine biologique jusqu'au milieu informatique. Nous développerons plus particulièrement les principes de fonctionnement des algorithmes génétiques. Un exemple concret d'application de l'industrie sera également étudié, le placement de formes. À partir de ces éléments nous discuterons de l'intérêt et de l'utilisation des algorithmes génétiques.

Mots clefs : algorithmes génétiques, placement de formes.

Table des matières

1 Introduction

Trouver une des meilleures solutions si ce n'est la meilleure à un problème ! C'est un objectif que l'on aimerait atteindre dans de nombreux domaines. Mais cela s'avère bien souvent difficile y compris en informatique, où de nombreuses méthodes ont été développées pour pallier à ces difficultés. Nous allons dans ce document nous intéresser à l'une d'entre elles en particulier : *les algorithmes génétiques*.

Les algorithmes génétiques (AG) font partie de la famille des algorithmes évolutionnaires (AE). Il s'agit de méthodes adaptatives qui ont pour fondement le processus d'adaptation génétique observé en biologie sur des générations successives d'une population. Ces algorithmes sont utilisés afin d'améliorer un ensemble de solutions d'un problème par sélections et modifications successives. En particulier, ces algorithmes fonctionnent particulièrement bien sur les problèmes d'optimisation combinatoire.

Nous commencerons par donner une approche générale des AG en discutant de différents types de problèmes qui ont amené entre autre à développer les algorithmes génétiques. Puis, nous nous intéresserons aux principes biologiques dont sont inspirés les AG en expliquant au préalable les idées qui permettent de passer de la biologie à l'informatique. Les algorithmes génétiques, comme n'importe quelle méthode, ne sont pas une solution universelle et ne s'appliquent efficacement que dans certaines conditions que nous aborderons avant de terminer par un bref rappel historique des étapes dans le domaine des AG.

1.1 Des problèmes difficiles !

Des problèmes apparemment aussi simples qu'organiser un emploi du temps, trouver le trajet en train le moins cher entre deux villes, ou comment un postier peut passer par toutes les rues d'un quartier en parcourant le moins de distance possible deviennent rapidement incalculables pour des données trop importantes. Ces problèmes d'optimisation combinatoire appartiennent à la classe des problèmes NP-Complets. Il n'existe pas d'algorithme polynomial connu pour de tels problèmes si bien que leur complexité est exponentielle.

Un informaticien face à un tel problème, pour des données de grandes tailles, est dépourvu quelle que soit la puissance des machines à sa disposition. Il est impossible de parcourir de manière naïve toutes les solutions pour trouver la meilleure. Il est donc nécessaire d'utiliser d'autres méthodes pour s'en approcher en un temps raisonnable.

À cet effet de nombreuses techniques ont été mises au point :

- la programmation dynamique, on se souvient et on réutilise ce que l'on a déjà calculé ;
- les heuristiques ;
- les algorithmes gloutons ;
- l'alpha-beta, on élimine des portions de l'espace de recherche dont on sait qu'elles ne pourront fournir de meilleures solutions ;
- colonie de fourmis : chaque fourmi marque son passage et plus un chemin est marqué, plus il est intéressant [DMC96] ;
- liste tabou : méthode de recherche qui garde en mémoire les actions modifiant une solution et interdit celle-ci pendant un certain temps [Glo86] ;
- le recuit simulé, basé sur des analogies physiques(voir section 3.4) ;
- les algorithmes génétiques.

Toutes ces méthodes ont pour but de trouver de manière efficace de bonnes solutions à des problèmes difficilement calculables. Chacune est plus ou moins adaptée à une classe de

problèmes en particulier mais ces différentes méthodes ont aussi en commun qu'aucune ne garantie de trouver la meilleure solution.

Nous nous intéresserons plus particulièrement aux algorithmes génétiques, partant des observations biologiques qui ont donné l'idée de départ, jusqu'à l'adaptation qui en a été faite en informatique. Puis, nous discuterons de quelques points permettant de discerner l'applicabilité des AG à un problème donné. Mais commençons par voir plus en détails les fondements biologiques.

1.2 Origine

L'origine des espèces est expliquée de diverses manières : la théorie de l'évolution de Darwin, et ses nombreuses déclinaisons, ou la création par un être intelligent. Nous nous contentons dans cette partie de décrire les principaux aspects de la théorie darwinienne originelle qui ont donné les idées de départ des algorithmes génétiques

Les algorithmes génétiques reprennent, en la modifiant et l'adaptant, la théorie développée par Charles Darwin au XIXème siècle [Dar59]. La théorie darwinienne de l'évolution des espèces soutient que les individus les plus adaptés survivent et peuvent donc se reproduire et transmettre leurs caractères à leurs descendants.

Nous allons décrire ci-après les principales étapes de la génétique, science de l'hérédité, en parlant dans un premier temps des lois que l'on peut observer puis, dans un second temps, en nous intéressant au support matériel des caractères héréditaires.

1.2.1 Observation macroscopique

De part l'observation de la nature, l'homme a conscience de notions telles que l'hérédité, les espèces et les races qui les composent, et la faiblesse de certains individus par rapport à d'autres dans le milieu naturel.

Hérédité : ensemble des caractères physiques ou moraux transmis des parents aux enfants.

Même sans comprendre les mécanismes sous-jacents, l'homme a depuis longtemps observé et utilisé l'hérédité. Les éleveurs, par exemple, savent bien que les meilleurs animaux sont choisis comme reproducteurs. De plus, les effets possibles d'un manque de diversité dans un cheptel sont connus ; comme une plus grande fréquence de maladie ou la dominance de certaines caractéristiques physiques. Ce phénomène est également observé chez les humains dans les familles où les mariages consanguins sont fréquents.

Les mécanismes complexes de l'hérédité (dominance, récessivité, ...) ont été mis en évidence par Mendel. Il a poursuivi ses travaux sur l'hérédité des caractères chez le pois de 1856 à 1864. Croisant des pieds différents par un seul caractère (graines lisses ou ridées, fleurs blanches ou colorées, ...), il va suivre, au fur et à mesure des générations et des nouveaux croisements, les lois de la réapparition de ce caractère, et donc de sa transmission. Il publie ses travaux en 1865 qui donneront les lois fondamentales de la génétique, appelées lois de Mendel.

Hypothèse sur la sélection naturelle. Selon Darwin, les individus les plus adaptés survivent ; mais cela ne suffit pas à expliquer la diversité des espèces animales sur terre, ainsi que les nombreuses races au sein des espèces et aussi des individus au sein des races. Certains avancent que seuls les plus faibles meurent précocement. Également, la majorité des mutations ne donnerait pas d'avantage ou de désavantage mais aurait seulement pour effet de maintenir

la diversité génétique et d'avoir une meilleure capacité d'adaptation aux variations de son environnement.

1.2.2 Observation microscopique

Malgré les progrès dans la compréhension des mécanismes naturels, décrit ci-dessus, il a fallu attendre des avancées dans la biochimie et la microbiologie pour mettre en évidence le fonctionnement physique des lois de Mendel.

C'est au XXème siècle que le fonctionnement des lois de l'hérédité a pu être expliqué. Au début des années 1930, quand les chromosomes ont été découverts par Thomas Hunt Morgan, on connaissait le support physique des caractères héréditaires. Puis, en 1953, Watson et Crick ont découvert l'acide désoxyribonucléique (ADN) qui contient le code génétique ce qui a permis de comprendre le mécanisme exacte de l'hérédité.

Le code génétique. L'intégralité du code génétique est contenue dans l'ADN qui se compose, en simplifiant, d'une succession de bases azotées : adénine, thymine, cytosine et guanine. Une suite de trois bases azotées permet de coder un acide aminé (il y en a une vingtaine) qui lui même entrera dans la composition d'une protéine.

FIG. 1 – ADN

Ce code génétique est capable de se reproduire au moyen d'un autre acide nucléique, l'ARN, et de muter soit par échange de brin d'ADN soit par une erreur dans le processus de recopie.

1.3 De la biologie à l'informatique

Nous avons très partiellement décrit ci-dessus le modèle biologique qui sert de base aux algorithmes génétiques. Nous allons essayer de montrer les principales idées qui permettent de passer de la biologie aux AG. D'une manière générale, les algorithmes génétiques ne retiennent que l'idée principale de chaque notion pour en simplifier la mise en œuvre et en améliorer l'efficacité.

1.3.1 Hérédité

Comme nous l'avons vu précédemment l'hérédité est un mécanisme très complexe dans la nature. Les algorithmes génétiques en général le simplifient. Tout d'abord la valeur de chaque gène est utilisée dans l'aspect du résultat, le phénotype. Il n'y a donc pas de caractères qui peuvent disparaître pendant une ou plusieurs générations pour ressurgir ensuite ; il est

cependant possible de gérer le mécanisme de récessivité en modifiant les opérateurs génétiques et le codage.

1.3.2 Sélection

Les meilleurs survivent ou les plus faibles sont éliminés, la réponse n'est pas encore trouvée. Pour ce qui est des algorithmes génétiques, en général on sélectionne les meilleurs individus. Ce choix provoque des difficultés telles que le risque d'uniformité de la population et la convergence vers un optima local ; ces problèmes et les solutions que l'on peut y apporter seront abordées en détails dans la section 2.2.4.

1.3.3 Codage

Dans la nature, c'est l'ADN qui contient les informations génétiques d'un individu. Ce codage est bien entendu simplifié dans les algorithmes génétiques où en général un tableau de bit suffit mais on peut avoir recours à des structures plus complexes.

1.3.4 Croisement

Plusieurs types de croisements sont observables dans la nature : en premier lieu, la reproduction sexuée où chacun des parents donne la moitié de son patrimoine génétique, deuxièmement, la méiose autrement dit deux chromosomes qui s'échangent une partie de leurs gènes, et enfin,la modification d'ADN au sein d'un même chromosome.

Les algorithmes génétiques utilisent principalement la deuxième méthode (crossover) bien qu'elle soit extrêmement rare dans la nature.

1.3.5 Mutation

C'est un phénomène rare dans la nature, qui de plus n'engendre pas forcément une modification effective de l'aspect de l'individu. Dans les algorithmes génétiques, la mutation est réalisée par l'altération du codage d'un ou plusieurs gènes. Elle est gérée de manière aléatoire.

1.4 Pour quoi ?

Nous avons déjà expliqué que les algorithmes génétiques fournissent une alternative à des méthodes plus classiques lorsqu'il est impossible de résoudre un problème difficile en un temps raisonnable.

Le point de départ est une population qui représente déjà un ensemble de candidats à la solution au problème traité. Le but est de trouver à partir de cette population, en lui appliquant divers opérateurs génétiques, de nouvelles solutions (qui peuvent être meilleures ou moins bonnes), d'en sélectionner et ainsi approcher de la meilleure solution pour un problème donné. Ces algorithmes peuvent donc être utilisés avec de nombreux types de problèmes. De plus, il n'est pas nécessaire d'avoir une connaissance poussée du problème traité, mais il suffit juste de savoir reconnaître une solution comme valide et l'évaluer.

Pour résumer, on peut s'intéresser aux algorithmes génétiques comme méthodes de résolution si :

- on ne connaît pas d'algorithme pour résoudre un problème ;
- on ne connaît pas d'algorithme polynomial et on a des données de grande taille.

– on ne cherche pas la meilleure solution d'un problème ;

1.5 Éléments Historiques

S'inspirer des mécanismes de la création, et plus particulièrement du vivant, pour en tirer des applications techniques n'est pas nouveau. Pour ce qui est des algorithmes génétiques, les premiers travaux significatifs se basant sur les mécanismes biologiques d'adaptation ont été proposés par John Holland et son équipe en 1962 [Hol62]. La nouveauté apportée par leurs recherches a été la prise en compte de l'opérateur de croisement en plus de la mutation. Voici quelques considérations d'ordre historique qui indiquent les principaux jalons et les étapes les plus significatives dans le développement des algorithmes génétiques.

1859 Parution du livre «The Origin of Species by means of Natural Selection» de Darwin, la théorie qui y est développée sera le point de départ pour l'idée de Holland.

1865 lois de Mendel : description des lois fondamentales de la génétique qui président à la transmission héréditaires des particularités d'un individu.

20ème siècle Découvertes scientifiques (ADN, chromosomes, gènes, mise en évidence des mutations génétiques, ...).

1973 Stratégie d'Évolution I. Rechenberg [Rec73] : Les algorithmes SE diffèrent des AG sur plusieurs points. La représentation d'une solution potentielle se fait avec des variables réelles et normalement distribuées, d'espérance mathématique nulle. L'algorithme d'origine conçu par Rechenberg est un simple algorithme de sélection-mutation sur un seul individu représentant une solution potentielle.

1975 J. H. Holland de l'Université du Michigan donne le premier modèle formel des algorithmes génétiques [Hol75] (voir section 2.4).

1989 Goldberg [Gol89] : introduction de deux nouvelles idées :
– liaison individu/environnement grâce à l'ADN,
– la liaison d'une solution au problème est donnée par son indice de qualité.

1.6 But et contenu de ce document

Après avoir discuté du contexte des AG la deuxième partie approfondira les principes de fonctionnement des algorithmes génétiques ainsi que leurs principales caractéristiques ; de plus, un aperçu des fondements théoriques sur lesquels se basent les AG sera donné ainsi que quelques variations possibles de l'algorithme canonique donné par Holland.

Nous poursuivrons par un exemple d'application au problème du sac à dos (knapsack), avec en particulier le problème de placement de formes qui consiste à remplir au maximum une surface donnée avec un ensemble de formes complexes. Par ailleurs, nous aborderons brièvement le recuit simulé pour permettre une comparaison avec les algorithmes génétiques.

Nous nous efforcerons d'amener les éléments nécessaires pour permettre, en conclusion, une critique objective des algorithmes génétiques tout en offrant des informations pratiques tout au long de ce document pour qui souhaiterait évaluer l'utilisation et l'applicabilité des AG.

2 Les algorithmes génétiques

Les AG sont une méthode de recherche qui fait progresser, par recombinaison, une population de solutions potentielles auxquelles sont attribuées des valeurs. Une solution potentielle est représentée par un génome qui caractérise le problème étudié. Le but est de réussir à générer des solutions potentielles ayant la meilleure évaluation possible, ce qui correspond à chercher les maxima d'une fonction f comme l'illustre la figure 2.

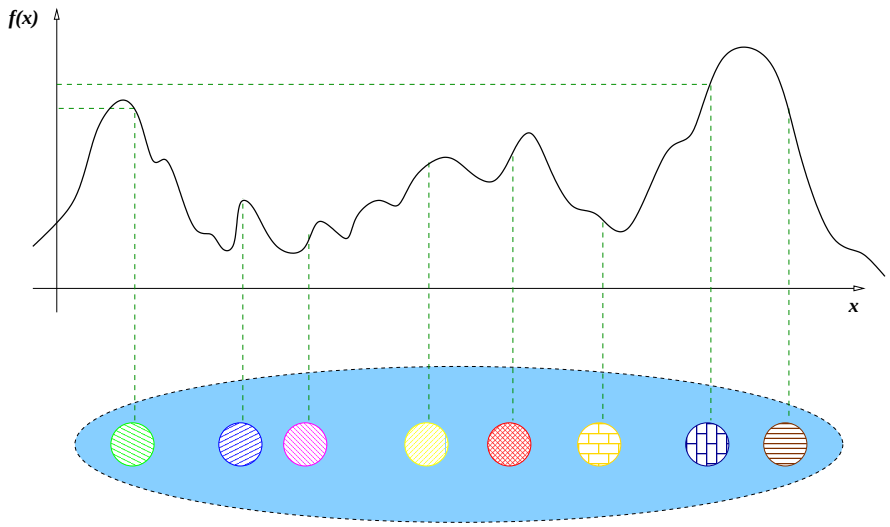

FIG. 2 – Explorer l'espace de recherche en gérant une population de solutions potentielles

2.1 Terminologie

Les AG suivent une approche inspirée de l'observation de ce qui se passe dans notre environnement naturel. La terminologie utilisée s'inspire également du vocabulaire utilisé en biologie. Nous allons donc définir le vocabulaire utilisé d'un point de vue informatique afin d'éviter un discours trop flou engendrant la confusion entre le milieu biologique et informatique.

individu Un individu est une solution potentielle. Il est représenté par un ou plusieurs chromosomes.

gène Un gène est la suite de symboles dans un chromosome représentant une caractéristique d'un individu.

chromosome Un chromosome est la représentation d'une partie d'un individu, ou de tout l'individu s'il n'y a qu'un seul chromosome (cas le plus courant).

2.2 Principes

Un AG, dans sa forme la plus simple, manipule une population d'individus soumis aux opérateurs génétiques de recombinaison (appelés croisement ou *cross-over*) et de mutation (voir les figures 5 et 6). Ces opérateurs génétiques sont appliqués sur des individus sélectionnés

en fonction de leur adaptation au problème. Ce processus de sélection et d'application des opérateurs génétiques est répété plusieurs fois afin d'améliorer le potentiel des individus (la figure 3 illustre ces propos). Dans la suite de cette étude, nous allons nous attacher à expliquer comment ces différents concepts sont utilisés et appliqués dans les AG.

Les algorithmes génétiques font partis de la classe des algorithmes probabilistes (stochastiques). Une partie aléatoire intervient donc dans le fonctionnement de ces algorithmes. Ce hasard est «dirigé» par une fonction d'évaluation qui déterminise le travail des opérateurs génétiques. Un algorithme génétique peut être divisé en trois parties :

- population initiale (sous forme de chromosomes ou chaînes),
- évaluation des individus (fitness),
- génération d'une nouvelle population issue de l'application des opérateurs génétiques (appariement, mutation, ...).

Ces trois étapes sont répétées jusqu'à ce que la condition d'arrêt soit satisfaite (un nombre fixé d'itérations, une évaluation satisfaisante des solutions, ...).

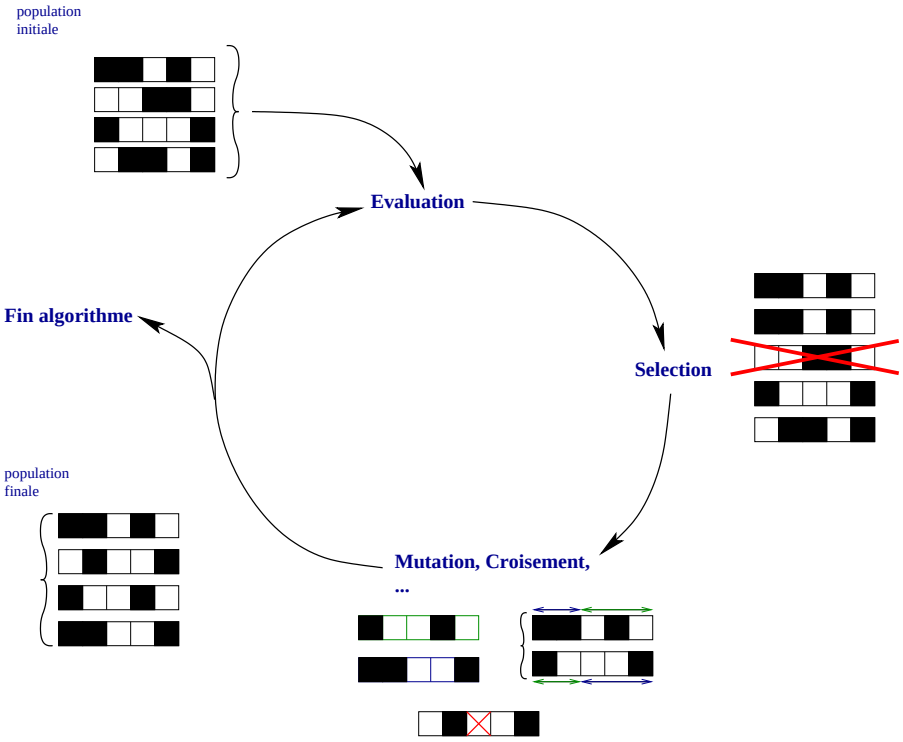

FIG. 3 – Idée générale du fonctionnement d'un AG

2.2.1 Initialisation

La population initiale peut être initialisée de deux façons, soit de manière aléatoire, soit à l'aide de solutions déjà connues ou obtenues à partir d'autres algorithmes de recherche. L'initialisation aléatoire présente l'avantage de ne demander aucune connaissance préalable du problème.

Une initialisation à partir de solutions connues ou partielles peut être utilisée pour privilégier une direction de recherche, mais elle peut faire converger trop rapidement la recherche

vers un optima local.

2.2.2 Itération

Pour pouvoir obtenir une optimisation, il est nécessaire de répéter plusieurs fois les opérations de sélection, croisement et mutation. L'algorithme 1 définit formellement ce processus. Nous avons déjà détaillé l'initialisation, aussi nous poursuivrons la présentation de chaque étape de l'algorithme dans la suite de ce document. La condition d'arrêt de l'algorithme dépend de plusieurs critères, comme nous l'étudierons à la fin de cette section.

Algorithme 1 Algorithme génétique

```
Initialisation de la population
Évaluation des individus de la population
tant que Critère d'arrêt non satisfait faire
  Sélection
  Application des opérateurs génétiques
  Évaluation
fin tant que
```

2.2.3 Évaluation et *fitness*

La fonction de *fitness*, très importante dans le fonctionnement d'un AG, permet de dire à quel point une solution donnée satisfait au problème. Son résultat étant utilisé lors de la sélection, son comportement influence la convergence de l'algorithme vers des solutions optimales pour le problème étudié. L'opérateur d'évaluation est le plus dépendant du problème traité. Bien souvent l'opérateur travaille sur une représentation différente du génome d'un individu ce qui induit un temps de calcul supplémentaire pour le décodage d'un individu.

La fonction d'évaluation et la fonction de *fitness* sont à différencier dans la pratique en fonction de la méthode de sélection. La fonction d'évaluation donne une estimation pour un individu tandis que la fonction de *fitness* fournit le coût d'un individu par rapport à l'ensemble de la population. Une méthode de sélection proportionnelle utilisant les résultats de la fonction d'évaluation (ne tenant compte que de l'individu, non pas du groupe) amènera très rapidement une uniformité dans la population diminuant ainsi l'espace de recherche étudié, d'où la nécessité d'une fonction de *fitness*. Dans le cas d'une sélection par tournoi (voir la section 2.2.4) dans la population, la fonction de *fitness* peut se réduire à la fonction d'évaluation.

Dans la conception d'une fonction de *fitness* on tiendra compte, en plus de chercher à faciliter le travail de la sélection, des critères – à maximiser ou minimiser – propres au problème. Par exemple, dans le cas d'une application des AG sur des problèmes de satisfaction de contraintes (l'optimisation correspondant à la recherche d'une solution violant le moins de contraintes), la fonction de *fitness* doit pouvoir attribuer une valeur proportionnelle au non respect des contraintes. Il est également intéressant de noter que la valeur de *fitness* peut être attribuée par un utilisateur (ex. valeur esthétique voir les fractals).

2.2.4 Sélection

L'opérateur de sélection va choisir les individus dans la population qui seront conservés dans la génération suivante, et ceux sur lesquels les opérateurs génétiques seront appliqués.

Cet opérateur va guider la population vers les valeurs élevées de la fonction à optimiser. De manière générale, la sélection étant basée sur le *fitness* des individus, les bons individus ont plus de chances d'apparaître dans la génération suivante ou d'être appliqués aux opérateurs génétiques (une ou plusieurs fois suivant le nombre d'occurrences déterminé par la sélection). On peut choisir également d'appliquer l'opérateur de sélection uniquement sur une partie de la population. Ceci permet le croisement entre des individus inter-générations et limite le risque de perte de bons individus. Plusieurs techniques existent pour sélectionner les individus : roulette, rang, tournoi, proportionnelle avec reste stochastique, steady-state, élitisme, ... Les méthodes les plus courantes, à savoir la sélection à la roulette, par tournoi, élitiste et steady-state, sont traitées ci-après.

Roulette La méthode la plus couramment choisie est la sélection proportionnelle. Les individus sont choisis proportionnellement à leur *fitness*. Pour reprendre la métaphore utilisée, dans une population de n individus, on place sur une roue n cases dont la taille correspond au *fitness* de chaque individu. Ensuite, la roue est lancée n fois. À chaque fois qu'une case est sélectionnée, l'individu correspondant est ajouté dans la nouvelle population en construction. De manière plus formelle, la probabilité qu'un individu soit choisi est la suivante :

$$\forall i \in [0..n], p(x_i) = \frac{fitness(x_i)}{\sum_{j=0}^{n} fitness(x_j)}$$

On remarque, comme vu dans la partie évaluation/fitness (voir section 2.2.3), que l'existence d'un individu ayant un *fitness* bien plus élevé que le reste de la population provoque, à terme, une homogénéité (le but des AG étant de fournir un ensemble de bonnes solutions). Pour pallier à ce problème, on peut également utiliser une méthode de probabilité cumulée [Mic92] qui permet d'éviter qu'un individu ne soit trop souvent sélectionné.

Tournoi La sélection par tournoi consiste à prendre au hasard k individus parmi la population et leur faire un tournoi, en l'occurrence sélectionner l'individu qui possède le *fitness* le plus élevé. Elle possède l'avantage de régler les problèmes de la sélection proportionnelle vu plus haut. k est généralement assez faible (2 à 5). Si $k = 1$ cela correspond à une sélection aléatoire où le *fitness* n'a plus aucun rôle. Plus k grandit (se rapproche de la taille de la population) plus les individus ayant un *fitness* élevé seront avantagés (le degré de compétition entre les individus augmente).

Steady-state La stratégie steady-state consiste en la sélection d'une paire d'individus auxquels sont appliqués les opérateurs génétiques. Cela permet des croisements inter-générations. En pratique, cette stratégie est souvent utilisée dans les algorithmes hybrides utilisant la recherche locale (voir section 2.5.2).

Élitisme Cette méthode de sélection vient en complément de celles vues précédemment. La sélection élitiste impose la présence des meilleurs individus de la population initiale dans la population finale. Cette méthode permet d'éviter de perdre de bons individus par l'application des opérateurs génétiques qui en modifieraient (mutation) ou en disperseraient (croisement) les caractéristiques. En pratique, la probabilité p qu'un individu soit conservé dans une population de taille n est souvent $p = 1/n$ (loi uniforme), ce qui signifie que le meilleur individu est conservé d'office dans la nouvelle population générée.

2.2.5 Opérateurs génétiques

Une fois la sélection effectuée, les opérateurs génétiques sont appliqués avec une certaine probabilité sur la population. Les deux opérateurs génétiques des AG standards sont le croisement et la mutation.

Croisement L'opérateur de croisement – également appelé opérateur d'exploitation car en combinant les individus il permet de découvrir de meilleures solutions – est un opérateur binaire. Il consiste en l'échange de gènes à partir des chromosomes de chaque parents. L'application de l'opérateur produit deux enfants. En fonction de la probabilité de croisement, entre 40% et 100%, les parents seront soit remplacés par les deux enfants (reproduction), soit maintenus dans la population (recopie). Nous développerons deux techniques de croisement.

Croisement multi-points Il s'agit du croisement le plus général. k points de sections sont choisis de manière aléatoire. Le chromosome est découpé en $k+1$ sections qui sont réparties alternativement pour produire les deux individus fils (voir figure 4). La répartition des sections peut également être effectuée aléatoirement. Ce croisement plus technique que le croisement simple, détaillé ci-après, permet d'hériter d'un gène de manière plus diversifiée et provoque un effet de cassure plus important. Pour une investigation plus poussée consultez [Gol89].

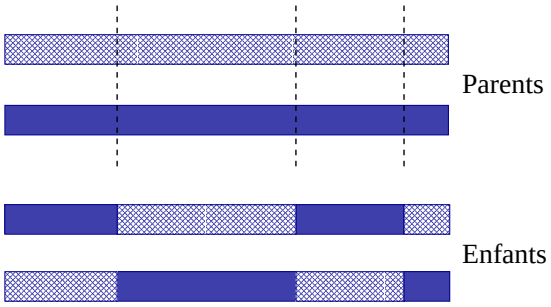

FIG. 4 – Croisement en trois points

Croisement simple Le croisement simple correspond à un cas particulier du croisement multi-points. C'est ce type de croisement qui est utilisé dans les algorithmes génétiques canoniques. Un point de section aléatoire est choisi. À partir des deux sections de chaque parents les enfants sont produits comme cela est présenté à la figure 5.

Mutation L'opérateur de mutation, illustré par la figure 6, est un opérateur unaire qui va modifier un ou plusieurs gènes d'un individu. La probabilité de mutation est généralement assez faible (entre 0% et 5%). La mutation introduit de l'aléatoire dans la recherche de solutions, qui permet d'éviter les optima locaux, mais s'il est mal réglé (probabilité trop importante) on s'orientera vers une recherche essentiellement aléatoire. Cet opérateur est également appelé opérateur d'exploration car il permet de sortir de la zone de l'espace de recherche définie par un individu.

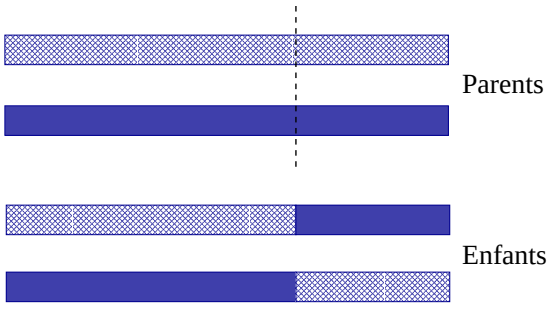

FIG. 5 – Croisement simple

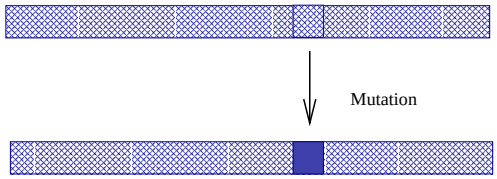

FIG. 6 – Mutation

2.2.6 Représentation d'un individu

Dans l'algorithme canonique de Holland [Hol75], les individus sont représentés par un seul chromosome. Ce chromosome est une chaîne de bits codant un point de l'espace de recherche. L'avantage de cette représentation est la simplicité des opérateurs de croisement et de mutation qui sont entièrement indépendants du problème sur lequel l'AG est appliqué. Toutefois, si un gène correspond à plus d'un bit du chromosome, l'utilisation des opérateurs classiques risque de produire des individus incohérents à partir d'individus sains. Pour pallier à ce problème, on peut soit utiliser un code de Gray (on travaille alors sur des valeurs entières et non plus binaires), soit utiliser des opérateurs génétiques spécialisés (voir section 2.5) qui sont dépendants de la représentation d'un individu. Une autre solution est d'utiliser un nouvel opérateur appelé opérateur de réparation qui corrigera les erreurs introduites par l'application des opérateurs génétiques classiques. La représentation d'un individu n'est souvent pas exploitable pour des opérateurs dépendants du problème tels que l'opérateur d'évaluation et l'opérateur de recherche locale, dans le cas d'un algorithme hybride (voir section 2.5.2), ce qui introduit un calcul supplémentaire lors de l'application de ces opérateurs.

2.2.7 Condition d'arrêt

Différents critères d'arrêt peuvent être mis en évidence, certains arbitraires, d'autres tenant compte de la convergence de l'algorithme. Le choix du critère d'arrêt dépend également du type de problème à résoudre. Dans le cas d'un problème d'optimisation, ne connaissant pas d'optima, on ne sait pas quand il est atteint ou peu éloigné de la solution courante. Bien souvent, on se décidera pour un nombre d'itérations fixées. S'il y a convergence prématurée (uniformité dans la population) l'algorithme pourra être stoppé, la solution obtenue étant en général non optimale.

2.3 Contrôle d'un algorithme génétique

Nous allons détailler au cours de cette section comment faire fonctionner un AG de manière à obtenir un résultat intéressant. Pour une description plus avancée sur le contrôle d'un AG, le lecteur pourra consulter le travail de Michèle Sebag et Marc Schœnauer [SS96]. Un algorithme génétique canonique est contrôlé par quatre paramètres :

1. La taille de la population détermine la capacité de l'AG à explorer correctement l'ensemble de l'espace de recherche. Une taille insuffisante pénalisera l'AG dans la recherche notamment s'il y a des maxima locaux dans la fonction f à maximiser (voir figure 2).
2. La fonction de fitness qui détermine la valeur des individus et influence directement sur la sélection des individus qui seront croisés et mutés.
3. La probabilité de croisement. Une probabilité de croisement élevée peut créer une uniformité dans la population.
4. La probabilité de mutation d'un gène. La mutation permet une meilleure exploration de l'espace de recherche, mais si le taux de probabilité est trop élevé il risque de détruire des gènes intéressants avant qu'ils n'aient pu être diffusés par le croisement.

D'autres éléments peuvent être pris en compte comme la stratégie de reproduction (steady-state, élitisme) qui protège une partie de la population de l'application des opérateurs de croisement et de mutation. Ces différents paramètres peuvent être fixés lors de l'exécution de l'AG ou au contraire évoluer dynamiquement : la mutation pourra être plus élevée au début (afin de privilégier la recherche de solutions), un opérateur spécialisé de recherche locale pourra être utilisé sur la fin pour améliorer des individus, ... Le contrôle d'un AG dépend surtout de techniques d'ingénierie et d'observations empiriques. Chaque problème ou classe de problèmes a vu des réglages et des adaptations spécifiques lui être appliqué. Une recherche dirigée suivant le problème permettra de prendre connaissance des réglages et techniques qui lui sont appropriées.

2.4 Un peu de théorie

John Holland a proposé en 1975 un formalisme des AG. Cette analyse théorique des algorithmes génétiques est basée sur le concept de «schéma» [Hol75, Gol89]. Un schéma décrit un ensemble de chromosomes ayant des caractéristiques en commun. Un schéma H peut être défini dans l'alphabet $\{0, 1, \#\}$ où $\#$ est un joker. Le schéma $1\#\#0$ décrit en fait les chromosomes $1000, 1010, 1100, 1110$. Nous allons nous efforcer par la suite de mieux comprendre le fonctionnement des AG au travers de la théorie des schémas.

2.4.1 Théorie des schémas

L'ensemble de tous les schémas couvre l'espace de recherche avec une forte redondance. Un chromosome de longueur l est le représentant de 2^l schémas différents. Dans la suite de cette section, nous appellerons $m(H, t)$ le nombre d'occurrences d'un schéma H à l'instant t. $f(H)$ est la valeur moyenne des chromosomes qui sont décrits par le même schéma H. $\bar{f}$ et f représentent respectivement la valeur moyenne des individus de la population de l'AG et la valeur d'un individu.

Pour un AG doté d'une population de n individus avec renouvellement total de la population, un individu i est sélectionné avec la probabilité $f_i / \sum f_i$ (voir par exemple la section

2.2.4). Sachant que $n/\sum f_i = 1/\bar{f}$, on en déduit le nombre d'occurrences d'un schéma H à l'instant $t+1$:

$$m(H, t+1) = m(H, t)\frac{f(H)}{\bar{f}}$$

En posant la valeur moyenne d'un schéma H :

$$f(H) = \bar{f} + c.\bar{f} \text{ avec } c > 0 \text{ et fixé}$$

on peut désormais écrire :

$$m(H, t+1) = m(H, t)\frac{\bar{f} + c\bar{f}}{\bar{f}} = m(H, 0).(1+c)^t$$

Le développement d'un schéma dans la population dépend de sa valeur comparée à la valeur moyenne de la population. La formule ci-dessus indique qu'un schéma dont la valeur est au dessus de la moyenne augmentera exponentiellement avec le temps. Mais un schéma ne survit pas forcément à l'application des opérateurs génétiques. Pour l'évaluation de la fragilité d'un schéma consultez la synthèse de A. Gaspard [Gas00, p.50–2].

Le parallélisme implicite L'évaluation d'un individu correspond à l'évaluation de tout les schémas le décrivant. Cela implique qu'un algorithme génétique traitant une population de n individus évalue un ensemble de schémas bien plus grand. Un algorithme génétique ne parcourt pas l'espace de recherche seulement au travers d'individus mais il évalue également une partition de cet espace.

2.5 Spécialisation des algorithmes génétiques

Les algorithmes génétiques canoniques présentent l'avantage d'être relativement simple à mettre en œuvre, mais bien souvent pour obtenir de meilleurs résultats il est nécessaire de tenir compte du problème que l'on cherche à résoudre. Certains points vu précédemment apporte des améliorations générales, par exemple de nouvelles méthodes de sélection et de stratégies de reproduction (tournoi, élitisme, steady-state), le croisement multi-points plus adapté pour des chromosomes longs. D'autres améliorations ont été mises au point comme la spécialisation des opérateurs et l'ajout d'un opérateur de recherche locale.

2.5.1 Opérateurs spécialisés

Lorsque la représentation du problème n'est pas standard ou que l'ordre des gènes a de l'importance, l'application des opérateurs standards au mieux est inefficace, au pire construit des solutions incohérentes. Cette spécialisation présente l'avantage de pouvoir guider la recherche stochastique pour une meilleure performance. Par exemple, M. C. Riff Rojas donne un ensemble d'opérateurs spécialisés pour les problèmes de satisfaction de contraintes [Roj97, p.50–4] qui permettent d'obtenir une recherche plus performante.

2.5.2 Algorithmes hybrides

L'alliance d'un AG avec une autre méthode de recherche donne souvent de bons résultats. Bien souvent, cela se traduira par l'ajout d'un opérateur de recherche locale (LS pour *local search*). Cet opérateur est appliqué après le croisement et la mutation pour améliorer le nouvel individu.

3 Placement de formes, une application pratique aux AG

Nous allons aborder un problème concret sur lequel on peut appliquer un algorithme génétique : le problème de placement de formes. Ce problème connaît de nombreuses applications dans l'industrie plus particulièrement dans la découpe (tôle, tissu, bois, pierre, ...) où l'objectif est de réduire au maximum les chutes et ainsi optimiser les coûts, tout cela en un temps relativement court.

Nous commencerons par présenter le problème de placement de formes. Puis, nous décrirons la classe théorique à laquelle il se rattache, le problème du *sac à dos*, et enfin nous aborderons la résolution du problème de placement de formes proprement dit à l'aide des algorithmes génétiques puis la qualité des résultats en nous basant sur les travaux développés dans [Bou98]. Pour finir, nous comparerons ces résultats avec ceux obtenus par l'algorithme du recuit simulé [Bou98] (voir section 3.4, page 21) et enfin critiquerons ces résultats.

3.1 Le problème de placement

On a au départ un ensemble de formes et un support à deux dimensions sur lequel on veut positionner ces formes. Cela doit être effectué en respectant certaines contraintes (décrites ci-après pour la confection) et en optimisant l'arrangement pour diminuer les chutes. Dans l'industrie, la matière première est sous forme d'une bande de tissu. Cela donne un problème très complexe.

Le placement de formes est le cœur du problème de la découpe de tissus et celui-ci est NP-Complet. Mais d'autres problèmes se rajoutent au placement proprement dit dans l'industrie de la confection :

- problème du carnet de commande,
- contraintes de placements :
  - contraintes d'outils de coupe / optimisation de trajectoire (gain de temps et d'énergie),
  - contraintes impératives (orientations, ajustements de carreaux, ...),
  - contraintes de préférences ou de qualité (alignement de motif, texture et/ou aspect différent suivant l'orientation, ...),
  - contraintes temporelles pour trouver une solution.

Pour une description en détail et les solutions possibles consultez [Bou98].

3.2 Problème du sac à dos

Le problème de placement de formes se ramène à celui du sac à dos dans sa forme discrète. Le but est de maximiser le poids au plus près de la limite que l'on peut porter en choisissant des objets (on ne peut ni les démonter ou les séparer d'où la discrétude du problème) dans un ensemble.

Ce problème est NP-Complet. Par conséquent, il n'existe pas d'algorithmes en temps polynomial.

3.2.1 Modélisation

Un objet à mettre dans le sac est représenté par un triplet $< i, p, v >$ qui représente sa référence, son poids et sa valeur :

- ensemble des objets, $O = \{< i, p, v > | i \in$ ensemble des références d'objets, $p, v \in \Re^+\}$
- sac à dos, $S \in \Re^+$

3.2.2 Formalisation

On cherche à trouver le sous-ensemble de plus grande valeur d'objet inférieur au poids maximum que l'on peut porter.

Données : O, S
Sortie : $\{\bar{O}, \bar{O} \subseteq O | \sum_{<i,p,v> \in \bar{O}} p \leq S\}$
Fonction : $\sum_{<i,p,v>} v$
Optimisation : max

3.2.3 Le problème du sac à dos au placement de formes

Comme nous l'avons dit précédemment le problème de placement de formes se ramène au problème du sac à dos. Les objets sont des pièces de tissus, avec une référence, une valeur proportionnelle à la surface et leur surface comme poids. Le sac à dos peut avoir plusieurs formes dans la découpe de tissus suivant qu'il s'agisse d'une découpe en bande (on fait avancer la bande au fur et à mesure) qui est utilisée ci-après ou en matelas (on superpose plusieurs pièces de tissus que l'on découpe, ce qui a pour inconvénient de nécessiter des tolérances de découpe plus grande).

Pour une résolution du problème du sac à dos en particulier de nombreux algorithmes génétiques hybrides ont été proposé, notamment VEGA, NSGA, SPEA, M-PAES, MOGLS. Dernièrement [BH02] ont proposé un nouvel algorithme génétique hybride GTS^{MOKP} (pour Genetic Tabou Search pour le MOKP) qui améliore grandement les performances sur la résolution à l'aide des AG du problème du sac à dos.

3.3 Résolutions à l'aide des AG

Nous allons présenter dans cette section la solution que propose [Bou98] au problème de placements de forme au moyen des algorithmes génétiques.

3.3.1 Représentations des individus

Codage d'une forme. C. K. Bounsaythip [Bou98] utilise le codage par peigne de contour qu'elle décrit comme suit : «Une méthode originale de codage de contour, appelé "codage par peigne de contour", a été développée par G. Roussel [Rou94]. Ce codage est basé sur le principe d'échantillonnage du contour visible de chaque côté du rectangle. Ces échantillons forment un ensemble de dents qui sont des segments exprimant la distance entre un point du contour et le bord du rectangle. Les dents sont générées le long du contour par une distribution de Dirac de période T dans la référence liée à chaque côté. Un peigne représente donc des échantillons du contour circonscrit (voir Figure I. 8).»

Deux approches sont ensuite utilisées pour coder des individus :

- on conserve uniquement des formes codées par peigne de contour, ce procédé complique le croisement puisqu'il s'agit d'un opérateur arithmétique ;
- on assemble ces formes dans une structure arborescente à l'aide de deux opérateurs de placement (connexité, numéro du côté adjacent à la pièce à placer ; rotation, orientation d'une forme).

La deuxième solution est préférable car elle facilite le travail pour le croisement.

3.3.2 Initialisation

Un individu initial est un arbre contenant une feuille indiquant le placement de la première forme dans une bande.

Cette première forme est positionnée en haut à gauche avec une certaine rotation. Dans la seconde feuille, on associe à cette première forme la matière où elle est placée, qui fournit une référence aux placements car fixe.

3.3.3 Le croisement.

Comme cela a été dit plus haut, le croisement revient à assembler deux arbres. Chaque forme n'est nécessaire qu'un nombre fixé de fois ; il faut donc vérifier la redondance des formes à chaque croisement. Les points de croisements sont choisis au hasard mais le croisement n'est effectué qu'après la vérification de redondance.

La redondance des formes est une contrainte dont on pourrait se passer en ajoutant une pénalité dans la fonction de fitness.

3.3.4 Mutation

Trois types de mutation peuvent être envisagées :
- changer un opérateur dans un arbre choisi aléatoirement, c'est à dire modifier l'orientation ou la connexité ;
- permuter au hasard deux formes dans l'arbre ;
- supprimer une forme de l'arbre, cela est utile pour pouvoir faire de la place pour une forme plus grande, ou permettre un croisement qui était impossible à peu de chose près.

La mutation n'est effectuée que si l'arbre satisfait les contraintes, sinon il reste inchangé.

3.3.5 Sélection

Pour une population de i individus la sélection s'effectue sur $i + l$ individus, où les l individus sont ceux nouvellement générés par les opérateurs génétiques. La priorité de sélection est donnée à ces nouveaux individus. Les l individus de plus faible fitness sont éliminés lorsque l'on atteint m individus.

Les parents et les enfants peuvent donc se côtoyer sur plusieurs générations, ce qui est vrai dans la nature pour de nombreuses espèces.

3.4 Résolutions à l'aide du recuit simulé

Nous allons tout d'abord brièvement présenter le recuit simulé et une rapide comparaison avec les algorithmes génétiques. Puis, nous donnerons les idées nécessaires pour utilisé cette méthode dans la résolution du problèmes de placement de formes.

3.4.1 Origine

Le recuit est un processus physique thermodynamique qui tire parti de la variation du degré de liberté des atomes en fonction de la température. C'est ce qui permet d'avoir du métal à l'état liquide. Mais la façon dont la température varie influe sur le produit fini : une baisse brutale de la température (trempe) produit un verre, une baisse progressive produit un cristal. Si la baisse progressive est trop rapide, il y a des défauts au niveau du cristal, le

recuit va permettre de redonner la liberté suffisante aux atomes pour corriger le défaut. En informatique, le recuit simulé est basée sur la méthode de Metropolis utilisé pour modéliser les processus physiques.

3.4.2 Algorithme

Algorithme 2 Algorithme de recuit simulé

choisir une solution potentielle s de manière aléatoire
fixer une température T
tant que critère d'arrêt faux **faire**
  choisir aléatoirement un voisin $\bar{s}$ de s
  calculer la différence de valuation des deux solutions : $\Delta = E(s) - E(\bar{s})$
  $s \leftarrow \bar{s}$ avec une probabilité $e^{\frac{\Delta}{T}}$ si $\Delta < 0$, 1 sinon.
  diminuer T
fin tant que

Le critère d'arrêt peut dépendre de la qualité de la solution, du nombre de variations depuis la solution s initiale, de la température, ... Plus la température de départ sera élevée, plus la solution pourra évoluer pendant le refroidissement. On remarque que si la solution $\bar{s}$ améliore la solution courante alors s est systématiquement remplacé sinon la solution $\bar{s}$ remplace s avec une probabilité qui décroît en même temps que la température T. Le recuit peut sortir d'un optima local pour autant que la température T soit suffisamment élevée. La principale difficulté d'application de cet algorithme est le choix d'une température et le réglage de la fonction de décroissance de cette température. Pour une description détaillée et les applications de cette méthode consultez [LA88].

3.4.3 Comparaisons avec les AG

Les algorithmes génétiques et le recuit simulé sont très proches. Les deux sont des algorithmes stochastiques. La détermination aléatoire d'une nouvelle solution voisine est remplacé par les deux opérateurs génétiques stochastiques. La fonction de sélection dépendant de la température est remplacé par une fonction de sélection générale. La principale différence est la manipulation d'une population par les AG alors que le recuit manipule un individu.

3.4.4 Éléments de résolution du problème de placement de formes

[Bou98] utilise un algorithme de recuit simulé hybride arborescent. Le placement est représenté sous formes d'arbre. La température de départ est égale à la surface totale de la bande, ainsi la température diminue à chaque ajout d'une forme dans l'arbre d'une quantité équivalente à sa surface. Le recuit consiste à enlever la dernière formes ajouté et faire un nouveau tirage pour placer une nouvelle forme. Deux critères d'arrêts sont retenus, premièrement un certain niveau de rendement à atteindre, deuxièmement un nombre maximal de recuit à ne pas dépasser.

3.5 Résultats des deux méthodes

Les graphiques (Figure 7 et 8) rassemblent les résultats sur trois tests des méthodes de résolutions utilisées par [Bou98]. Les deux qui nous intéressent sont les algorithmes génétiques et le recuit simulé.

On remarque que les algorithmes génétiques se distinguent en obtenant le meilleur rendement toutes méthodes confondues et se maintiennent dans la moyenne en général.

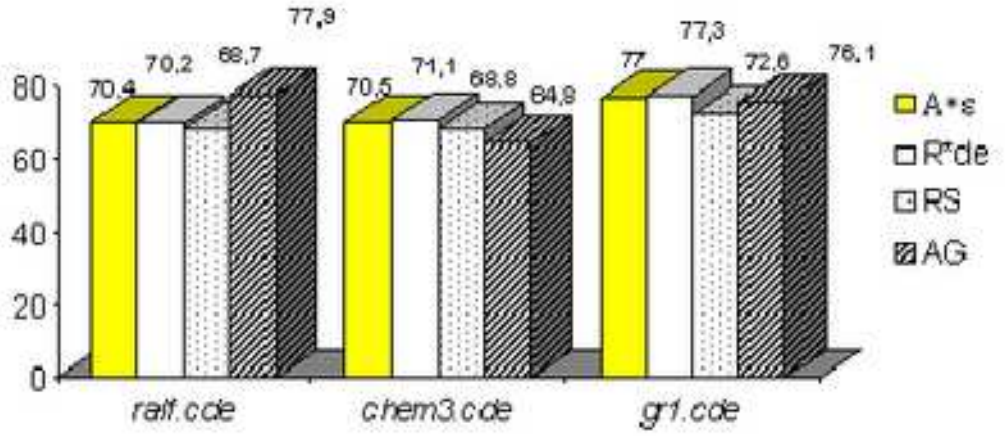

FIG. 7 – Exemples de rendement en %. Les AG se distinguent en ayant le meilleur rendement obtenu.

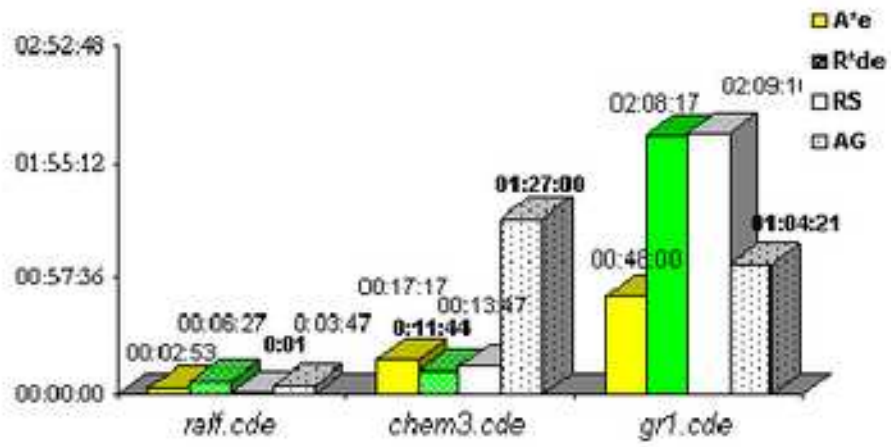

FIG. 8 – Exemples de durée de recherche pour les rendements de la Figure précédente.

Dans l'exemple chem3.cde (Figure 8, l'algorithme génétique est le plus coûteux en temps de calcul. Cela est explicable par la très grande taille des données ajouté au fait qu'il faut une taille de population assez importante pour avoir une bonne diversité. De plus ce temps de calcul pourrait être amélioré en évitant les réévaluations des individus qui sont conservés

d'une génération à l'autre.

3.6 Conclusion

La résolution du problème de placement de formes à l'aide des algorithmes génétiques donne de bons résultats. C'est une méthode performante qui peut être utilisée de manière complémentaire avec d'autre mécanismes non exhaustifs ou stochastiques tel que le recuit simulé ou les recherches en arbres décrites dans [Bou98].

4 Conclusion

Les AG apportent une méthode de résolution intéressante et déjà utilisée dans l'industrie [AY96] : simulation de comportement d'automobilistes pour la sécurité routière, simulation d'animaux artificiels, optimisation de profils dans l'aéronautique, ..., et comme nous l'avons vu précédemment, dans la confection pour la résolution du problème de placement de formes. Mais, comme toutes techniques, les AG présentent des qualités et des défauts.

4.1 Qualités

Les AG, et plus particulièrement les AG canoniques, sont rapides à mettre en œuvre et ne demandent qu'une connaissance limitée du problème. Ils permettent d'obtenir rapidement des résultats valides voir même intéressants. Les AG peuvent également être améliorés au travers d'une spécialisation pour un problème donné et/ou en l'alliant à d'autres stratégies de recherches. Ces améliorations donne souvent de bons résultats.

4.2 Défauts

Malgré un certain cadre théorique, aucune garantie de convergence vers les meilleures solutions n'est donné. Il existe également des problèmes dits AG-difficile, appelés également problèmes trompeurs, sur lesquels les AG ne peuvent fournir de bonnes optimisations [Gol89]. Le codage présente, pour beaucoup de problèmes, une difficulté importante du fait de la représentation binaire canonique. De plus, le réglage d'un AG, pour obtenir de meilleurs résultats, s'avère souvent délicat et difficile, ce qui est plus particulièrement vérifiable dans le cas des AG spécialisés qui donnent les meilleurs résultats.

4.3 Pour aller plus loin

Une nouvelle analyse des AG a vu le jour il y a quelques années ; il s'agit de MOSES (Mutation Or Selection Evolution Strategy). L'algorithme MOSES utilise seulement les opérateurs de sélection et de mutation. À la différence des algorithmes génétiques classiques, MOSES possède une formulation qui permet de déduire des conditions théoriques de convergence qui ne dépendent pas de la fonction à optimiser (voir [CF01]).

Table des figures

Liste des Algorithmes

Références

[AY96] Jean-Louis Amat and Gérard Yahiaoui. *Techniques avancées pour le traitement de l'information.* Cépaduès-éditions, 1996.

[BH02] Vincent Barichard and Jin-Kao Hao. Un algorithme hybride pour le problème de sac à dos multi-objectifs. JNPC, 2002.

[Bou98] Catherine Khamphang Bounsaythip. *Algorithmes Heuristiques et Evolutionnistes : Application à la Résolution du Problème de Placement de Formes Irrégulières.* PhD thesis, Université des Sciences et Technologies de Lille, 1998.

[CF01] A. Cercueil and O. François. Monte carlo simulation and population-based optimization. Congress on Evolutionnary Computation (CEC01), IEEE Press, 2001.

[Dar59] Charles Darwin. *The Origin of Species by means of Natural Selection; or, the Preservation of Favoured Races in the Struggle for Life.* 1859.

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B : Cybernetics*, pages 26(1) :29–41, 1996.

[Gas00] Alessio Gaspar. *Étude de l'Adaptativité de Systèmes Évolutionnaires en Environnement à Fitness Dynamique.* PhD thesis, I3S CNRS / UPRES-A, 2000.

[Glo86] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operation Research*, 13 :533–549, 1986.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* 1989.

[Hol62] John Holland. Outline for adaptative systems with programs roving cellullar computers, 1962.

[Hol75] John Holland. *Adaptation in Natural and Artificial System.* 1975.

[LA88] P. Van Laarhoven and E. Aarts. *Simulated Annealing : Theory and Applications.* Kluwer, 1988.

[Mad02] Blaise Madeline. *Algorithmes évolutionnaires et résolution deproblèmes de satisfaction de contraintes en domaines finis.* PhD thesis, UNSA - UFR Sciences Ecole Doctorale STIC, 2002.

[Mic92] Z. Michalewicz. Genetic algorithms + data structures = evolution programs. *Artificial Intelligence*, 1992.

[Rec73] I. Rechenberg. Evolutions strategie : Optimierung technischer systeme nach prinzipien der biologischen evolution, 1973.

[Roj97] M. C. Riff Rojas. *Résolution de problèmes de satisfaction de contraintes avec des algorithmes évolutionistes.* PhD thesis, Ecole Nationale des Ponts et Chaussées, 1997.

[Rou94] Gilles Roussel. *Optimisation du Placement de Formes Irrégulières sur Matières Planes. Application à l'industrie de la Confection.* PhD thesis, Université des Sciences et Technologies de Lille, 1994.

[SS96] M. Sebag and M. Schoenauer. Contrôle d'un algorithme génétique. *Intelligence artificielle*, 1996.

Autres références

Introduction to Genetic Algorithms. Matthew Wall, http ://lancet.mit.edu/ mbwall/, Massachusetts Institute of Technology. Introduction brève et complète aux algorithmes génétiques.

Algorithmes génétiques pour résoudre le problème du commis voyageur. Yves Coueque, Julien Ohler, Sabrina Tollari, TER, http ://sis.univ-tln.fr/ tollari/TER/.

Wikipédia. http ://en.wikipedia.org/, http ://fr.wikipedia.org/. Information générale.

Dictionnaire en 4 volumes Larousse. Librairie Larousse, 1988. Définitions précises et informations générales.