REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE
Ministère de l'Enseignement Supérieur
& de la Recherche Scientifique

Université USTO MB
Faculté des Sciences
Département d'Informatique

Spécialité : Informatique
Option : Reconnaissance des Formes- Intelligence Artificielle (RF-IA)
Laboratoire SIMPA

EXPOSE DE OA

Titre :
«La Méthode de Recherche à Voisinage Variable (RVV)

Sous la direction de :

Pr. BENYETTOU

Réaliser par :

M.ROUMANE
BENCHERKI

Année universitaire : 2011 – 2012

I- INTRODUCTION

Les métaheuristiques forment un ensemble de méthodes utilisées en recherche opérationnelle pour résoudre des problèmes d'optimisation réputés difficiles. Résoudre un problème d'optimisation combinatoire, c'est trouver l'optimum d'une fonction, parmi un nombre fini de choix, souvent très grand. Les applications concrètes sont nombreuses, que ce soit dans le domaine de la production industrielle, des transports ou de l'économie – partout où se fait sentir le besoin de minimiser des fonctions numériques, dans des systèmes où interviennent simultanément un grand nombre de paramètres.
A ces problèmes de minimisation, les métaheuristiques permettent, dans des temps de calcul raisonnables, de trouver des solutions, peut-être pas toujours optimales, en tout cas très proches de l'optimum ; elles se distinguent en cela des méthodes dites *exactes*, qui garantissent certes la résolution d'un problème, mais au prix de temps de calcul prohibitifs pour nombres d'applications industrielles.
Nous nous proposons de fournir un panorama de toutes ces techniques, parfois redoutablement efficaces, qui se développent depuis une vingtaine d'années.

Après avoir donné quelques définitions préalables, et rappelé les énoncés de quelques problèmes traditionnels d'optimisation combinatoire, nous replacerons les métaheuristiques dans leur contexte : nous dégagerons leurs principales propriétés, et nous établirons une première classification.
Puis nous passerons en revue les métaheuristiques les plus usuelles, en analysant leur mode de fonctionnement, en évoquant les évolutions auxquelles elles ont donné lieu, et en soulignant leurs avantages et leurs inconvénients respectifs. Nous verrons à cette occasion toute la richesse et la variété des procédés mis en jeu dans les métaheuristiques.
Enfin, dans un troisième temps, nous nous pencherons sur la question de la stratégie à suivre pour mettre en place une métaheuristique, et nous verrons que, malgré leur dissemblances, les métaheuristiques sont toutes structurées par un petit nombre de concepts, dont la compréhension aide au paramétrage de leurs algorithmes de base. L'examen des techniques d'hybridation et l'association de métaheuristiques avec d'autres méthodes d'optimisation clôtureront notre étude.

I-1/LES METAHEURISTIQUES, UNE REPONSE AUX PROBLEMES D'OPTIMISATION DIFFICILE

Le cadre de l'optimisation combinatoire

En mathématiques, *l'optimisation* recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction, avec ou sans contraintes.
En théorie, un problème d'optimisation combinatoire se définit par l'ensemble de ses instances, souvent infiniment nombreuses. Dans la pratique, le problème se ramène à résoudre numériquement l'une de ces instances, par un procédé algorithmique.

A chaque instance du problème est associé un ensemble discret de solutions *S*, un sous-ensemble *X* de *S* représentant les solutions admissibles (réalisables), ainsi qu'une fonction de coût *f* (appelée aussi *fonction objectif*).

f assigne à chaque solution $s \in X$ le nombre *f(s).*

Résoudre un problème d'optimisation combinatoire consiste alors à trouver une solution $s \in X$ optimisant la valeur de la fonction de coût *f*.
Formellement, on cherche donc $s^* \in X$ tel que $f(s^*) \leq f(s)$ pour tout $s \in X$.

Une telle solution *s** s'appelle une *solution optimale*, ou un *optimum global*.

Remarque : on peut tout aussi bien résoudre des problèmes de maximisation, les principes de résolution restant naturellement les mêmes.

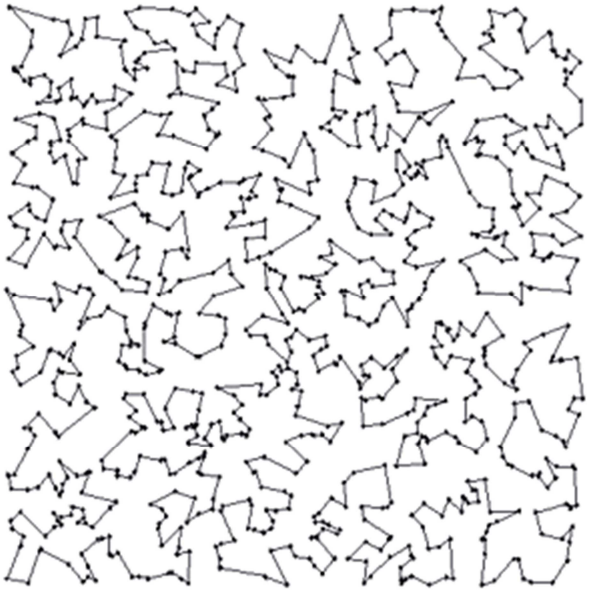


Schéma 1 : représentation du chemin le plus court passant par une distribution aléatoire de 1000 points.

Dans le schéma ci-dessus, le problème était de trouver le chemin minimal passant une seule fois par un certain nombre de points (problème du voyageur de commerce, voir paragraphe suivant). Ce graphique correspond à une certaine *instance* du problème. Avec une distribution différente de points, nous aurions eu une autre instance du problème.

Il existe une seconde formalisation du problème d'optimisation combinatoire, plus générale, qui inclue la notion de *contraintes* et d'*affectation* de valeurs à des variables. En voici la définition. Soit :

- un ensemble de variables $(x_1, x_2, \ldots, x_n)$
- un ensemble de domaines de définitions $D_1, D_2, \ldots, D_n$
- un ensemble de contraintes *C* sur les variables (par exemple, des inéquations, ou bien l'obligation que toutes les variables prennent des valeurs différentes)
- une fonction objectif *f* que l'on cherche à minimiser : $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \Re$

L'ensemble S des solutions possibles peut alors se définir comme :
S = { $s = \{(x_1, v_1), \ldots, (x_n, v_n)\}$ tels que $v_i \in D_i$, et tels que *s* satisfasse toutes les contraintes *C*}

Les problèmes d'affectation sous contraintes possèdent de nombreuses applications pratiques, comme l'affectation de ressources, le groupement, la classification, la planification, l'emploi du temps et l'ordonnancement, dans des domaines très variés…

I-2/LES METAHEURISTIQUES, UNE REPONSE AUX PROBLEMES D'OPTIMISATION DIFFICILE

Le cadre de l'optimisation combinatoire

En mathématiques, *l'optimisation* recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction, avec ou sans contraintes.
En théorie, un problème d'optimisation combinatoire se définit par l'ensemble de ses instances, souvent infiniment nombreuses. Dans la pratique, le problème se ramène à résoudre numériquement l'une de ces instances, par un procédé algorithmique.
A chaque instance du problème est associé un ensemble discret de solutions *S*, un sous-ensemble *X* de *S* représentant les solutions admissibles (réalisables), ainsi qu'une fonction de coût *f* (appelée aussi *fonction objectif*).
f assigne à chaque solution $s \in X$ le nombre *f(s)*.
Résoudre un problème d'optimisation combinatoire consiste alors à trouver une solution $s \in X$ optimisant la valeur de la fonction de coût *f*.
Formellement, on cherche donc $s^* \in X$ tel que $f(s^*) \leq f(s)$ pour tout $s \in X$.
Une telle solution *s** s'appelle une *solution optimale*, ou un *optimum global*.
Remarque : on peut tout aussi bien résoudre des problèmes de maximisation, les principes de résolution restant naturellement les mêmes.

Schéma 1 : représentation du chemin le plus court passant par une distribution aléatoire de 1000 points.

Dans le schéma ci-dessus, le problème était de trouver le chemin minimal passant une seule fois par un certain nombre de points (problème du voyageur de commerce, voir paragraphe

suivant). Ce graphique correspond à une certaine *instance* du problème. Avec une distribution différente de points, nous aurions eu une autre instance du problème.

Il existe une seconde formalisation du problème d'optimisation combinatoire, plus générale, qui inclue la notion de *contraintes* et d'*affectation* de valeurs à des variables. En voici la définition. Soit :

- un ensemble de variables $(x_1, x_2, \ldots, x_n)$
- un ensemble de domaines de définitions $D_1, D_2, \ldots, D_n$
- un ensemble de contraintes *C* sur les variables (par exemple, des inéquations, ou bien l'obligation que toutes les variables prennent des valeurs différentes)
- une fonction objectif *f* que l'on cherche à minimiser : $f : D_1 \times D_2 \times \ldots \times D_n \rightarrow \Re$

L'ensemble S des solutions possibles peut alors se définir comme :
S = { $s = \{(x_1, v_1), \ldots, (x_n, v_n)\}$ tels que $v_i \in D_i$, et tels que *s* satisfasse toutes les contraintes *C*}

Les problèmes d'affectation sous contraintes possèdent de nombreuses applications pratiques, comme l'affectation de ressources, le groupement, la classification, la planification, l'emploi du temps et l'ordonnancement, dans des domaines très variés…

I-2.1/L'optimisation difficile

L' « optimisation combinatoire » consiste à trouver la meilleure solution entre un nombre fini de choix. Autrement dit, à minimiser une fonction, avec ou sans contraintes, sur un ensemble fini de possibilités. Quand le nombre de combinaisons possibles devient exponentiel par rapport à la taille du problème, le temps de calcul devient rapidement critique.

Ce temps de calcul devient si problématique que pour certains problèmes, on ne connaît pas d'algorithme exact *polynomial*, c'est-à-dire dont le temps de calcul soit proportionnel à N^n, où *N* désigne le nombre de paramètres inconnus du problème, et où *n* est une constante entière. Lorsqu'on conjecture qu'il n'existe pas une telle constante *n* telle qu'un polynôme de degré *n* puisse borner le temps de calcul d'un algorithme, on parle alors d'optimisation difficile, ou de problèmes *NP-difficiles* (c'est tout l'objet de la théorie de la NP-complétude).

Exemples de problèmes

L'un des exemples les plus caractéristiques du champ de l'optimisation combinatoire est le problème du **voyageur de commerce**, dans lequel un représentant doit visiter un certain nombre de villes, avant de retourner à son point de départ : la question est de déterminer le trajet le plus court traversant chaque ville une seule fois, c'est-à-dire la succession de villes qui minimise la tournée du représentant.
De nombreux problèmes d'ingénierie peuvent se ramener au problème du voyageur de commerce (PVC), d'où son intérêt. On le retrouve dans le domaine des réseaux informatiques par exemple, avec les algorithmes de routage.
Par ailleurs, le PVC est un cas particulier d'un problème plus général, celui des **tournées de véhicules**, qui consiste à calculer les meilleurs parcours pour une flotte de véhicules devant

livrer un ensemble de produits dans plusieurs destinations à partir d'un entrepôt. Chaque véhicule a une capacité limitée. Le problème des tournées de véhicules trouve ses applications dans le domaine de la distribution bien sûr, mais aussi dans les télécoms.

Un autre problème classique est celui de **l'affectation quadratique**, qui, par rapport au PVC, introduit la notion de *flot* circulant entre les villes, et qui peut s'énoncer ainsi : il s'agit de déterminer comment placer *n* objets dans *n* emplacements de manière à minimiser la somme des produits *flots* par *distances*. Mathématiquement, cela revient à minimiser :

$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} \cdot d_{ij}$ où f_{ij} représente le flot entre l'objet *i* et l'objet *j*, et d_{ij} la distance entre l'objet *i* et l'objet *j*.

La répartition de bâtiments dans un espace donné en fonction du nombre de personnes amenées à circuler entre ces bâtiments, ou la façon de répartir des modules électroniques sur une carte en fonction du nombre de connexions les liant les uns aux autres, reviennent à résoudre le problème de l'affectation quadratique.

Enfin, troisième problème combinatoire caractéristique, **l'ordonnancement de tâches**, dont l'énoncé classique est le suivant : soit un ensemble *m* de machines, un ensemble *t* de tâches à réaliser, chaque tâche étant composée d'une certaine suite d'opérations. Une machine ne peut réaliser qu'une seule opération à la fois. Comment ordonner l'exécution des opérations sur les différentes machines de telle façon que le temps mis à faire tout le travail soit fait soit minimal ?

Naturellement, dans la vie courante, les situations sont encore plus compliquées, et il faut rajouter à ces situations-types des contraintes, des particularités, certaines « règles du jeux » qui viennent encore compliquer l'énoncé du problème à résoudre.

I-2.2/Méthode exacte et méthode approchée

Une recherche exhaustive par énumération explicite de toutes les solutions est impensable pour résoudre un problème d'optimisation difficile en raison du temps de calcul induit. Dans le cas du problème du voyageur de commerce, par exemple, l'espace de recherche croit en (*n*-1)!, où *n* est le nombre de villes à visiter, ce qui dépasse rapidement les capacités de calcul de n'importe quel ordinateur.
Avec seulement 50 villes, il faudra évaluer 49! trajets, soit $6{,}08.10^{62}$ trajets.
C'est l'explosion combinatoire.

Néanmoins, la résolution d'un tel problème d'optimisation peut se faire de manière exacte, en modélisant soigneusement le problème, puis en appliquant un algorithme ad-hoc, qui écarte d'emblée l'examen de certaines configurations, dont on sait d'ores et déjà qu'elles ne peuvent pas être optimales.
Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de *séparation et évaluation* (branch-and-bound), ou les *algorithmes avec retour arrière* (backtracking). Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Mais malgré les progrès réalisés (notamment en matière de programmation linéaire en nombres

entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés avec les applications de taille importante.
Notons qu'il existe ainsi des applications informatiques génériques (AMPL, CPLEX, LINDO, MPL, OMP, XPRESS...) qui permettent à l'ingénieur de résoudre facilement les problèmes pouvant s'écrire sous une forme algébrique en variables binaires ou entières.
Si les méthodes de résolution exactes permettent d'obtenir une ou plusieurs solutions dont l'optimalité est garantie, dans certaines situations, on peut cependant se contenter de solutions de bonne qualité, sans garantie d'optimalité, mais au profit d'un temps de calcul réduit. On utilise pour cela une méthode **heuristique**, adaptée au problème considéré, avec cependant l'inconvénient de ne disposer en retour d'aucune information sur la qualité des solutions obtenues.

I-3/Heuristique et métaheuristique

Afin d'améliorer le comportement d'un algorithme dans son exploration de l'espace des solutions d'un problème donné, le recours à une méthode *heuristique* (du verbe grec heuriskein, qui signifie « trouver ») permet de guider le processus dans sa recherche des solutions optimales.
Feignebaum et Feldman (1963) définissent une **heuristique** comme une règle d'estimation, une stratégie, une astuce, une simplification, ou tout autre sorte de système qui limite drastiquement la recherche des solutions dans l'espace des configurations possibles. Newell, Shaw et Simon (1957) précisent qu'un processus heuristique peut résoudre un problème donné, mais n'offre pas la garantie de le faire.
Dans la pratique, certaines heuristiques sont connues et ciblées sur un problème particulier.

La **métaheuristique**, elle, se place à un niveau plus général encore, et intervient dans toutes les situations où l'ingénieur ne connaît pas d'heuristique efficace pour résoudre un problème donné, ou lorsqu'il estime qu'il ne dispose pas du temps nécessaire pour en déterminer une.
En 1996, I.H. Osman et G. Laporte définissaient la métaheuristique comme « un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales ».

En 2006, le réseau Metaheuristics (metaheuristics.org) définit les métaheuristiques comme « un ensemble de concepts utilisés pour définir des méthodes heuristiques, pouvant être appliqués à une grande variété de problèmes. On peut voir la métaheuristique comme une « boîte à outils » algorithmique, utilisable pour résoudre différents problèmes d'optimisation, et ne nécessitant que peu de modifications pour qu'elle puisse s'adapter à un problème particulier ».
Elle a donc pour objectif de pouvoir être programmée et testée rapidement sur un problème.
Comme l'heuristique, la métaheuristique n'offre généralement pas de garantie d'optimalité, bien qu'on ait pu démontrer la convergence de certaines d'entre elles. Non déterministe, elle incorpore souvent un principe stochastique pour surmonter l'explosion combinatoire. Elle fait parfois usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche.

I-4/Classification des métaheuristiques

Les métaheuristiques n'étant pas, a priori, spécifiques à la résolution de tel ou tel type de problème, leur classification reste assez arbitraire. On peut cependant distinguer :

a) les approches « **trajectoire** »
Ces algorithmes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans cette catégorie, se rangent :
- la méthode de descente
- le recuit simulé
- la méthode Tabou
- la recherche par voisinage variable

Le terme de *recherche locale* est de plus en plus utilisé pour qualifier ces méthodes.

b) les approches « **population** » (ou évolutionnaires)
Elles consistent à travailler avec un ensemble de solutions simultanément, que l'on fait évoluer graduellement. L'utilisation de plusieurs solutions simultanément permet naturellement d'améliorer l'exploration de l'espace des configurations. Dans cette seconde catégorie, on recense :
- les algorithmes génétiques
- les algorithmes par colonies de fourmi
- l'optimisation par essaim particulaire
- les algorithmes à estimation de distribution
- le path relinking (ou chemin de liaison)

Notons d'ores et déjà que ces métaheuristiques évolutionnaires seront probablement plus gourmandes en calculs, mais on peut supposer aussi qu'elles se prêteront bien à leur parallélisation.
Certains algorithmes peuvent se ranger dans les deux catégories à la fois, comme la méthode GRASP, qui construit un ensemble de solutions, qu'elle améliore ensuite avec une recherche locale. La plupart des méthodes développées ces dernières années sont d'ailleurs souvent à cheval sur ces deux approches.

Une autre manière, plus intuitive, de classifier les métaheuristiques consiste à séparer celles qui sont inspirées d'un **phénomène naturel**, de celles qui ne le sont pas.
Les algorithmes génétiques ou les algorithmes par colonies de fourmi entrent clairement dans la première catégorie, tandis que la méthode de descente, ou la recherche Tabou, vont dans la seconde.

On peut également raisonner par rapport à l'usage que font les métaheuristiques de la **fonction objectif**. Certaines la laissent « telle quelle » d'un bout à l'autre du processus de calcul, tandis que d'autres la modifient en fonction des informations collectées au cours de l'exploration – l'idée étant toujours de « s'échapper » d'un minimum local, pour avoir davantage de chance de trouver l'optimal. La *recherche locale guidée* est un exemple de métaheuristique qui modifie la fonction objectif.

Enfin, il faut distinguer les métaheuristiques qui ont la faculté de **mémoriser** des informations à mesure que leur recherche avance, de celles qui fonctionnent sans mémoire, en aveugle, et qui peuvent revenir sur des solutions qu'elles ont déjà examinées. On distingue la mémoire à

court terme (celles des derniers mouvements effectués), et la mémoire à long terme (qui concerne des paramètres synthétiques plus généraux). Le meilleur représentant des métaheuristiques avec mémoire reste la recherche Tabou. Dans les « sans mémoire », on trouve le recuit simulé par exemple.
On s'intéresse dans notre exposé à l'une de ces méthodes qui est la méthode de Recherche à Voisinage Variable (RVV).

II -HISTORIQUE :

La Recherche à Voisinages Variables (RVV) a été proposé par Mladenovic et Hansen en 1997. une RVV utilise méthodiquement plusieurs types de voisinages.
Soit L = (N(1),…, N(T)) une liste finie de voisinages, où N(t)(s) est l'ensemble des solutions dans le te voisinage de s.Dans la plupart des méthodes de recherche locale, on a T=1. Ces voisinages interviennent de la manière suivante
Dans le processus de recherche d'une RVV. Étant donnée une solution initiale s, on génère une solution voisine s' dans N(1)(s) et on lui applique une procédure de recherche locale afin d'obtenir une solution s''. Si f(s'') < f(s), alors on pose s = s'' et on génère une nouvelle solution voisine dans N(1)(s). Sinon, la solution courante reste s et on change de voisinage en générant une solution s' dans N(2)(s).
Plus généralement, on change de voisinage à chaque fois que l'un d'entre eux n'est pas parvenu, après application de la procédure de recherche locale, à améliorer la solution courante s. Par contre, dès qu'un voisinage permet d'améliorer s, alors on recommence le processus avec le premier voisinage de la liste L.

III -RECHERCHE A VOISINAGE VARIABLE (RVV) :

III-1/Structure de voisinage et minimum local

Soit S un ensemble de solutions à un problème d'optimisation, et soit *f* la fonction objectif.
Une structure de voisinage (ou tout simplement un voisinage) est une fonction N qui associe un sous-ensemble de S à toute solution $s \in$ S. Une solution $s' \in$ N(s) est dite *voisine* de s.
Une solution $s \in$ S est un *minimum local* relativement à la structure de voisinage N si $f(s) \leq f(s') \ \forall\ s' \in$ N(s).
Une solution $s \in$ S est un *minimum global* si $f(s) \leq f(s') \ \forall\ s' \in$ S.

Certaines méthodes d'optimisation, qui partent d'une solution initiale et qui l'améliorent en explorant son voisinage immédiat, présentent l'inconvénient de s'arrêter au premier minimum local trouvé.
Comme nous le verrons plus loin, les métaheuristiques contiennent donc souvent une technique ou une astuce permettant d'éviter de se retrouver piégé dans ces minima locaux, en explorant davantage tout l'espace des solutions, de façon à augmenter la probabilité de rencontrer le minimum optimal, c'est-à-dire le minimum global.

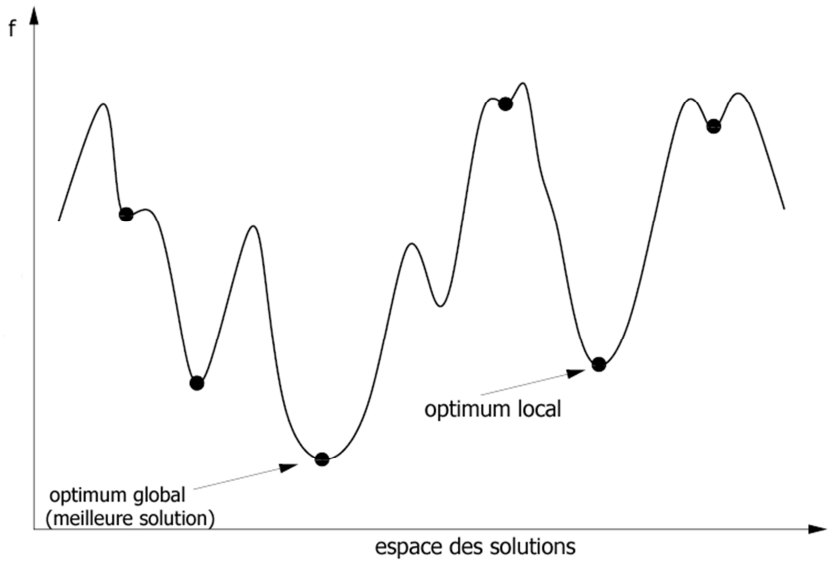


Schéma 2 : analogie entre une fonction numérique à une variable,
et la fonction de coût d'un problème combinatoire

Dans le cadre de l'optimisation combinatoire, en pratique, on aura tout intérêt à définir le voisinage en considérant l'ensemble des modifications élémentaires que l'on peut appliquer à une solution *s* donnée, par exemple l'ensemble des permutations (si les solutions peuvent s'écrire sous la forme d'une séquence finie d'éléments, comme le cas se présente fréquemment en optimisation combinatoire)
Si cet ensemble est trop grand, on pourra toujours le réduire à un sous-ensemble, aléatoirement, ou en fonction d'un critère précis.

Ainsi, dans le cas du problème du voyageur de commerce, et partant d'un trajet donné A – B – C – D – E, un voisinage pourra se définir comme l'ensemble des inversions de 2 villes sur le parcours (dans notre exemple, une inversion possible serait : A – B – D – C – E), ou bien comme l'ensemble des déplacements unitaires (dans notre exemple, un déplacement serait : A – B – E – C – D)

III-2/Definition :
La Recherche à voisinage variable (RVV) est une métaheuristique récente pour la résolution de problèmes d'optimisation combinatoire et globale, dont l'idée de base est le changement systématique de voisinage au sein d'une recherche locale. Dans ce chapitre, nous présentons les règles de base de RVV et de certaines de ses extensions. De plus, des applications sont brièvement résumées. Elles comprennent la résolution approchée d'un ensemble de problèmes d'optimisation, des manières d'accélerer les algorithmes exacts et d'analyser le processus de résolution des heuristiques ainsi qu'un système automatique de découverte de conjectures en théorie des graphes.
L'heuristique RVV utilise les constats suivants :

- ✓ Un minimum local par rapport à un voisinage n'en est pas nécessairement un par rapport à un autre
- ✓ Un minimum global est un minimum local par rapport à tous les voisinages possibles
- ✓ Pour de nombreux problèmes, les minimaux locaux par rapport à un ou à plusieurs voisinages sont relativement proches les uns des autres.

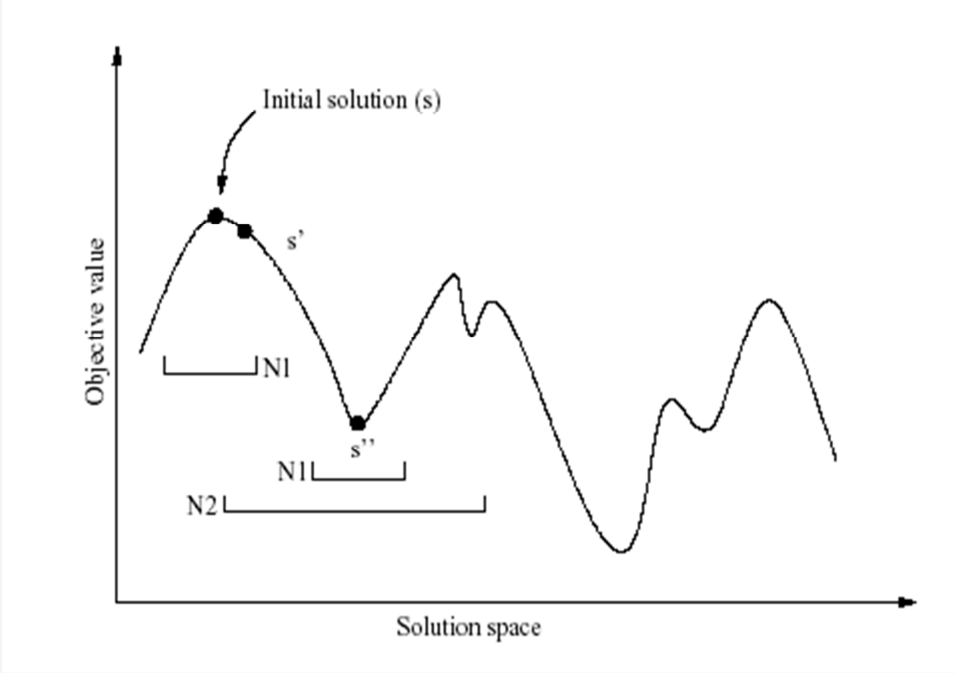


Figure : Recherche à voisinage variable

III -3 /L'Algorithme

Principe : changement systématique de voisinage pour s'éxtraire des maxima locaux.
On peut combiner une recherche locale avec un changement de voisinage variable, ce qui nous donne notre algorithme **VNS**

Initialisation : On choisit un ensemble de structures de voisinages N_k ,avec k =1…;k_{max} que l'on utilisera lors de la phase de secousse, On choisit une solution initiale x et une condition d'arrêt.

Répeter la séquence suivante jusqu'à ce que l'on atteigne la condition d'arrêt :

k⟵ 1

Répeter les étapes suivantes jusqu'à ce que k = k_{max} .

1- ***Secousse :***on génère un point x' au hasard dans le k-ième voisinage de x (x' □ N_k(x).

2-*Recherche locale :*

-Exploration du voisinage :On cherche le meilleur voisin x'' de x' (On applique une méthode de recherche locale avec x' la solution initiale)

3-*Se déplacer ou non :* Si l'ptimum local x'' est meilleur que le point d'origine x (si f(x'') <f(x')), alors on s'y déplace (x'⟵ x''), et on reprend la recherche dans N_1 (k⟵1) ; sinon on incrémente k (k⟵ k+1).

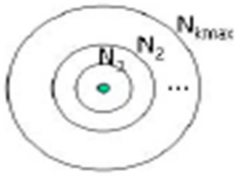
Une des étapes de la recherche VNS, le choix initial, se fait habituellement de manière aléatoire, ce qui peut dans certains cas donner une convergence rapide, mais peut également entraîner une perte de temps.

Le critère d'arrêt de la recherche locale peut être une limite en durée. Une autre possibilité est de s'arrêter quand la meilleure solution trouvée par l'algorithme n'a pas été améliorée depuis un nombre donné d'itérations. Les algorithmes de recherche locale sont des algorithmes sous-optimaux puisque la recherche peut s'arrêter alors que la meilleure solution trouvée par l'algorithme n'est pas la meilleure de l'espace de recherche.
Cette situation peut se produire même si l'arrêt est provoqué par l'impossibilité d'améliorer la solution courante car la solution optimale peut se trouver loin du voisinage des solutions parcourues par l'algorithme
Pour qu'une Recherche à Voisinages Variables soit efficace, il est recommandé d'utiliser des structures de voisinage qui soient complémentaires en ce sens qu'un minimum local pour un voisinage ne l'est pas forcément pour un autre.

Initialisation :

-Sélectionner l'ensemble de structures de voisinage N_k

-Trouver la solution initiale x.

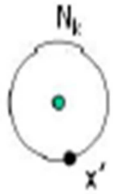


Répeter la séquence suivante jusqu'à ce que l'on atteigne la condition d'arrêt :

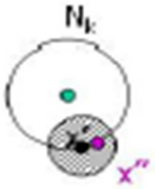
- k ← 1

- ***Répeter*** les étapes suivantes jusqu'à ce que k = k_{max}

1-*Secousse :* on génère un point x' au hasard dans $N_k(x)$.

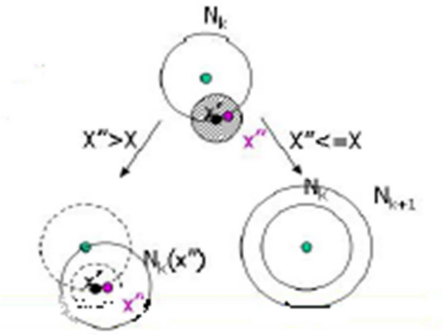


2- Recherche local : x'' est l'optimum obtenue.



3- Se déplacer ou non :

- Si x'' est meilleur que x alors x = x'' et k = 1.
- Sinon k= k+1.



III -4 /Exemples :

Exemple : Coloriage d'un graphe [MET]

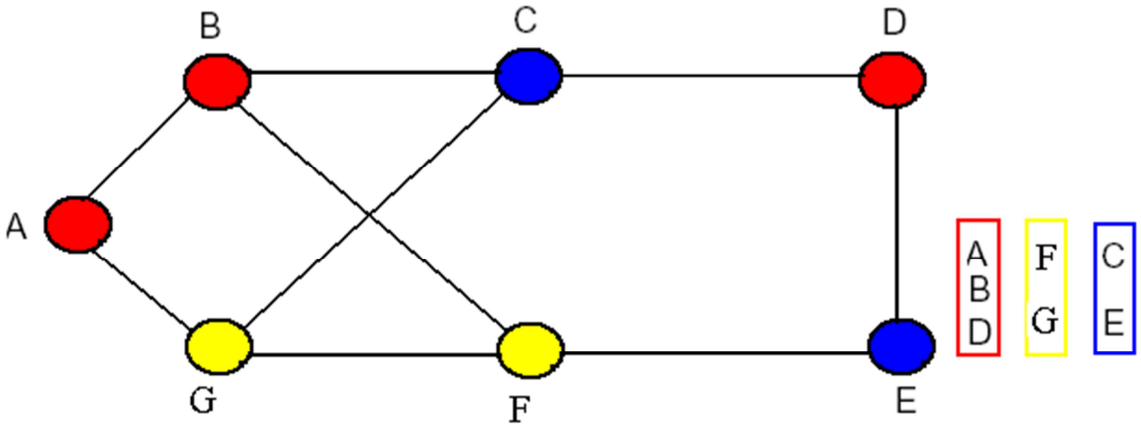
Considérons un problème de coloriage des sommets d'un graphe G (V, E). Supposons que toute fonction c : V ⟶ IN est considérée comme une solution.

Une ***coloration*** est une attribution d'une couleur à chaque sommet.
Une coloration est dite ***légale*** si aucune arête n'a la même couleur à ses deux extrémités.
Une coloration est dite ***partielle*** si une partie des sommets n'a aucune couleur.
Une coloration utilisant les couleurs 1, . . . , k est une k-coloration.
Le plus petit nombre de couleurs n´ecessaire pour colorer G est le nombre chromatique de G, $\gamma(G)$.

Une arête est dite en conflit si ses deux extrémités sont de même couleur. De même un sommet est en conflit s'il est l'extrémité d'une arête en conflit.

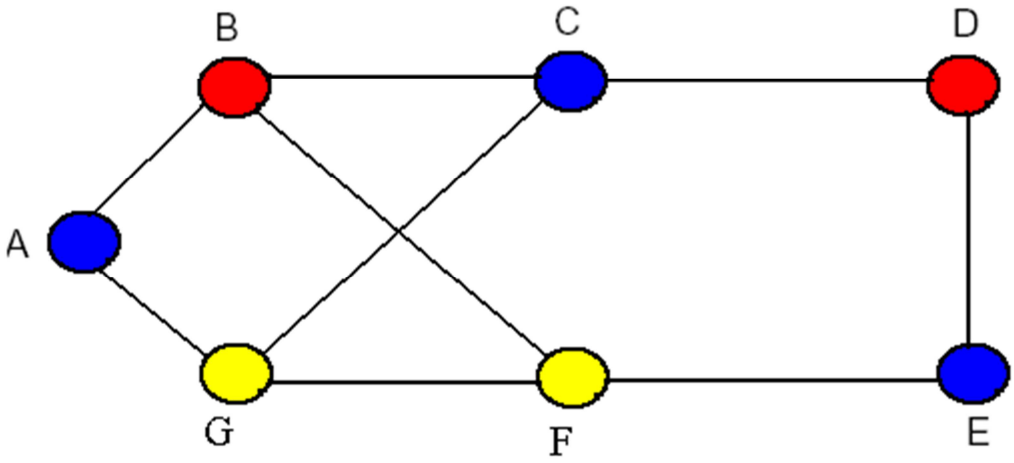
Considérons la fonction F qui compte le nombre de sommets en conflit. Etant donné une coloration c considérons deux voisinages :

- Le voisinage N1consiste à changer la couleur d'un sommet en conflit (la nouvelle couleur doit être l'une des couleurs utilisées dans le graphe).
- Le voisinage N 2 consiste à choisir un sommet W voisin du sommet V en conflit, à condition que W soit libre et de permuter les couleurs de V et W.



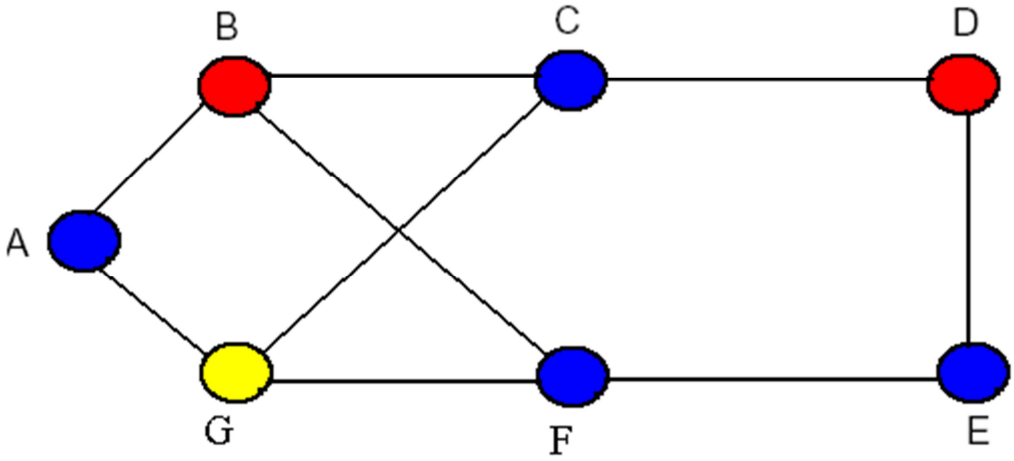
F(x)= 2 : (A,B) et (F , G)

En utilisant le voisinage N^1 :



F(x) = 1 : (F ,G)

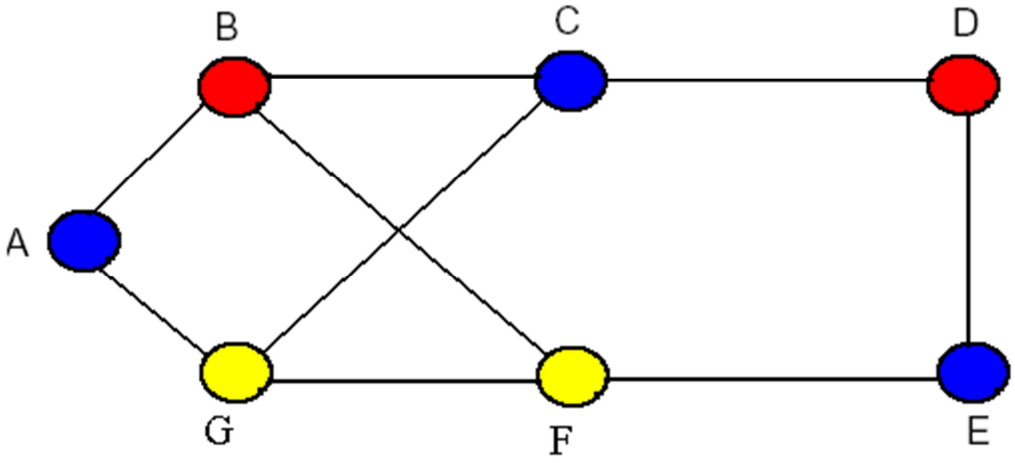
En utilisant pour la deuxième fois le voisinage N^1 :



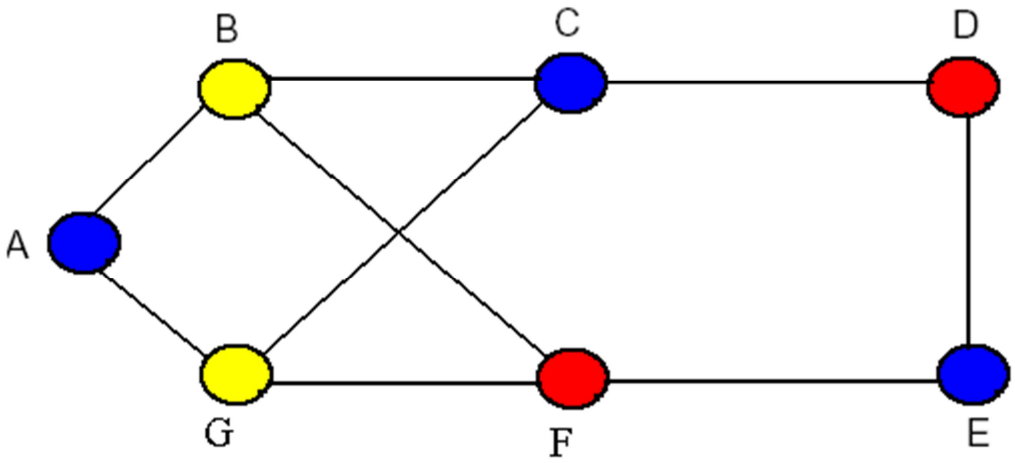
F(x) =1 : (F, E)

On remarque que cette solution n'a pas amélioré la solution précédente, le problème est reste toujours (un autre conflit) ,alors on garde notre solution précédente et on lui applique le deuxième voisinage.

La solution précédente est :



En utilisant le voisinage N^2 :



F(x) =0 : pas de conflit

III -5/ Avantages et inconvénients de RVV :

III-5.1/ Les Avantages :

- Elle est simple à utilisée puisqu'elle est basée sur un principe simple qui nous permet de changer systématiquement de voisinage lorsqu'on se retrouve bloqué dans un minimum local.
- Etant très généraliste, elle s'applique à un grand nombre de problèmes d'optimisation combinatoire.

- Elle est facile à utiliser : les différentes étapes de l'algorithme sont faciles à comprendre et à mettre en œuvre.

- Elle est efficace : les meilleures solutions sont obtenues en un temps de calcul modéré.

- Le fait d'utiliser plusieurs voisinages permet de diversifier l'exploration de l'espace de solution afin d'accéder à un plus grand nombre de régions intéressantes, ce qui conduit à une méthode plus robuste que le recuit simulé ou la recherche tabou.

- Un grand nombre de problèmes réels n'est pas optimisable efficacement par des approches purement mathématiques, la recherche à voisinage variable peut être alors utilisée avec profit. **[NET 1]**

III-5.2/Les Inconvénients :

- Elle est souvent moins puissante que des méthodes exactes sur certains types de problèmes.

- Elle ne garantie pas non plus la découverte d'un optimum global en un temps fini.

- Généralement, un choix doit être fait quant au critèr d'arrêt adéquat. Un nombre d'évaluation ou un temps imparti est souvent utilisé, mais on peut également choisir d'atteindre une valeur donnée de la fonction objective (le but étant alors de trouver une solution associée). Il est également possible de choisir des critères dépendants du comportement de l'algorithme, comme une dispersion minimale de la population de points ou un paramètre interne approprié. En tout état de cause, le choix du critère d'arrêt influencera la qualité de l'optimisation. **[NET 1]**

IV- Conclusion :

Les métaheuristiques étant très généralistes, elles peuvent être adaptées à tout type de problème d'optimisation pouvant se réduire à une « boîte noire ». Elles sont souvent moins puissantes que des méthodes exactes sur certains types de problèmes. Elles ne garantissent pas non plus la découverte de l'optimum global en un temps fini. Cependant, un grand nombre de problèmes réels n'est pas optimisable efficacement par des approches purement mathématiques, les métaheuristiques peuvent alors être utilisées avec profit.

La notion d'efficacité se rapporte généralement à deux objectifs contradictoires : la vitesse et la précision. La vitesse est souvent mesurée en nombre d'évaluations de la fonction objectif,qui est la plupart du temps la partie la plus gourmande en temps de calcul. La précision se rapporte à la distance entre l'optimum trouvé par la métaheuristique et l'optimum réel, soit du point de vue de la solution, soit de celui de la valeur. Bien souvent, un algorithme rapide est peu précis, et inversement.

BLIOGRAPHIE

1.P.Hansen and N.Mladenovité. An introduction to variable neighbourhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, Meta-heuristics, Advances and trends in local search paradigms for optimization, pages 433-458. Kluwer Academic Publishers.1999.

2. P. Hansen and N. Mladenovité. variable neighbourhood search. In F . Glover and G. A. Kochenberger , editors , *Handbook of Metaheuristics* ,chapter 6. Kluwer , 2003.

3. C. Helmberg and F . Rendl. Solving quadratic (0,1)- problems by semidefinite programs and cutting plantes .*Mathematical Programming* , 82 :291-315, 1998.

4. N.Mladenovié and P.Hansen. variable neighbourhood search. *Comput. Oper . Res .;* 4(11) :1097-1100,1997

5.[Net 1] : http://fr.wikipedia.org/wiki/M%C3%A9taheuristique

Sites Internet :

http://www.univ-valenciennes.fr/LGIL/sevaux/
http://tew.ruca.ua.ac.be/eume/