1. ***Introduction :***

*L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique en générale. Son importance se Justifie d'une part par la grande difficulté des problèmes d'optimisation [1].Et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire[2] .Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données[3].*

*Étant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO).*

*Ces méthodes peuvent être classées sommairement en deux grandes catégories :*

* *les méthodes exactes (complètes) : qui garantissent la complétude de la résolution ;*
* *les méthodes approchées (incomplètes):qui perdent la complétude pour gagner en efficacité.*

*Le principe essentiel d'une méthode exacte consiste à effectuer une énumération explicite de toutes les solutions (c'est-à-dire de les tester une à une, méthode envisageable pour tous les problèmes à variables à valeurs bornées). Cette méthode montre très vite ses limites dès que le nombre de variables augmente puisque sa complexité est en kn où k représente le nombre de valeurs que peut prendre une variable et n le nombre de variables du problème [4].*

*Parmi les méthodes exactes, on trouve La programmation dynamique et le Branch & Bound etc.*

*Par contre Les méthodes approchées qui a pour but de trouver une solution de bonne qualité (C’est-à-dire assez proche de l'optimale) en un temps de calcul raisonnable sans garantir l'optimalité de la solution obtenue. Elles sont fondées principalement sur diverses heuristiques, souvent spécifiques à un type de problème.*

*Les méta-heuristiques constituent une autre partie importante des méthodes approchées et ouvrent des voies très intéressantes en matière de conception de méthodes heuristiques pour l’optimisation combinatoire.est pour cela on constaté deux approches ont clairement démontré leur utilité dans de nombreux domaines.*

*La première de ces approches, appelée Recherche Locale ou bien (voisinage), est couramment utilisée de puis plus de 20 ans et comprend, entre autres :*

1. *la méthode de descente ;*
2. *Le recuit simulé ;*
3. *Recherche tabou ;*
4. *Méthode du bruitage ;*
5. *Les algorithmes d’acceptation avec seuil.*

*La deuxième approche, appelée Méthode Évolutive, est plus récente puisqu’elle date du début des années 90. Elle s’est particulièrement fait connaître par :*

* *les algorithmes génétiques ;*
* *les algorithmes de colonies de fourmis etc.*

*A cet exposé nous intéressons sur la première approche qui est ‘les méthodes de recherche locale’.*

1. ***notions de base :***

***2.1) Quelques définitions :***

* ***Une solution*** *: est une affectation de toutes les variables du problème.*
* ***Une solution optimale*** *: est une solution de coût minimal (ou maximal).*
* ***Un mouvement*** *: est une opération élémentaire permettant de passer d'une solution a une solution voisine (exemple : changer la valeur d'une variable, échanger deux variables).*
* ***Le voisinage*** *: d'une solution est l'ensemble des solutions voisines, c'est a dire l'ensemble des solutions accessibles par un mouvement (et un seul).*
* ***Un essai*** *: est une succession de mouvements.*
* ***Une recherche locale*** *: est une succession d’essais. [5]*

***2.2) optimisation combinatoire :***

*On peut trouver de nombreuses définitions de problèmes d’optimisation combinatoire.*

*En mathématiques, l'optimisation recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction, avec ou sans contraintes.*

*Dans le cas d’une minimisation, un problème d’optimisation se représentera de la forme suivant :*

*Minimiser f(s) (f : fonction à optimiser) ;*

*Avec g(s)<=0 (m contraintes d’inégalités) ;*

*Et h(s)=0 (p contrainte d’égalités). [6]*

*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*  *X le nombre f (s).*

*Résoudre un problème d’optimisation combinatoire consiste alors à trouver une solution*

*s* *X optimisant la valeur de la fonction de coût f.*

*Formellement, on cherche donc s\**  *X tel que f(s\*) ≤ f(s) pour tout s*  *X.*

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

*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* 
* *un ensemble de domaines de définitions* 
* *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 objective f que l’on cherche à minimiser :* 

*L’ensemble S des solutions possibles peut alors se définir comme :*

*S = {  tels que  , 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… [7]*

***2.3) Les heuristiques :***

*Les méthodes heuristiques sont particulièrement utiles pour les problèmes qui nécessitent une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Certaines sont ciblées sur un problème particulier (heuristiques spécifique). [4]*

***2.4) Les méta-heuristiques :***

***2.4.1)*** ***Étymologie :***

*Ce mot vient de la combinaison de deux mots grecs:*

*-Le verbe heuriskein : « trouver » ;*

*- le suffixe méta : « au-delà », « dans un niveau supérieur ». [4]*

***2.4.2) Définition :***

*Une méta-heuristique[[1]](#footnote-2) est constituée d'un ensemble de concepts fondamentaux.qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation. [4]*

***2.4.3) Propriétés fondamentales et fonctionnement :***

*Elles sont :*

* *Efficaces : elles permettent d’explorer efficacement l’espace de recherche.*
* *Non déterministe : puisqu’elles ne donnent aucune garantie d’optimalité*

*Elles possèdent deux autres propriétés intéressantes :*

* *Elles peuvent des mécanismes qui permettent d’éviter d’être bloqué dans des régions de*

*L’espace de recherche. C’est leur capacité à s’échapper d’un minimum local*

* *Les méta-heuristiques peuvent faire usage de l’expérience accumulée durant la recherche de l’optimum.*

*L'exécution des méta-heuristiques se déroule en trois phases :*

*Diversification – Intensification – Mémoire*

* ***La diversification*** *regroupe les actions ayant pour but de récolter des informations sur le problème dans Un ensemble défini (ou espace local) lors de l'intensification.*
* ***L'intensification*** *vise à utiliser les informations de la mémoire et celles récoltées lors de la phase de Diversification afin de définir les meilleurs espaces de recherche locaux futurs. Et de les faire parcourir de Façon optimale par les fonctions objectifs.*
* ***La mémoire*** *est le support de l'apprentissage permettant à l'algorithme de ne tenir compte que des zones où l'optimum est susceptible de se trouver et de garder en mémoire les résultats passés.*

*Les méta-heuristiques progressent itérativement et alternativement entre les phases de*

*Diversification, d'intensification et d’apprentissage [4].*

*2.4.4) classement :*

*Pour classer l’ensemble ces meta-heuristiques, il existe en realite plusieurs possibilites (ce qui est notamment ce qui du au fait qu’elles sont le fruit de travaux en recherche operationnelle et en intelligence artificielle). Le classement depend donc du point de vue. Ci-apres, une classification possible de ces algorithmes. On distingue :*

* *Constructives (algorithmes glouton, methode Pilote, GRASP)*
* *Recherche locale (algorithmes de descente appeles aussi methodes d’ameliorations iteratives, recuit simule, algorithme a seuil, recherche Tabou, methode de bruitage)*
* *Evolutionnistes (algorithmes genetiques, algorithmes d'evolution, recherche dispersee, methode des chemins, systemes de fourmis)*
* *Reseaux de neurones (Modele de Hopfield-Tank, machine de Boltzmann, reseau auto-adaptatif, reseau elastique)*

1. **La recherche locale :**

***3.1) le principe de recherche locale :***

*Une méthode de recherche locale est basée sur l’évolution itérative d’une solution unique. Le passage d’une solution vers une autre se fait grâce à la définition de structure de voisinage qui est un élément très important dans la définition de ce type de méthode.*

*Nous résumons le principe dans les points suivants :*

1. *partir d'une solution sinon approchée du moins potentiellement bonne et d'essayer de l'améliorer itérativement.*

*Pour améliorer une solution on ne fait que de légers changements (on parle de changement local, ou de solution voisine).*

1. *relancer la méthode plusieurs fois en changeant le point de départ pour avoir plus de Couverture.*
2. *tout problème est considérée comme un problème d'optimisation (même les problèmes de satisfaction : le coût à optimiser est alors le nombre de contraintes insatisfaites).*

## *3.2)* *Structure de voisinage et minimum locale :*

*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**S. Une solution s’* *N(s) est dite voisine de s.*

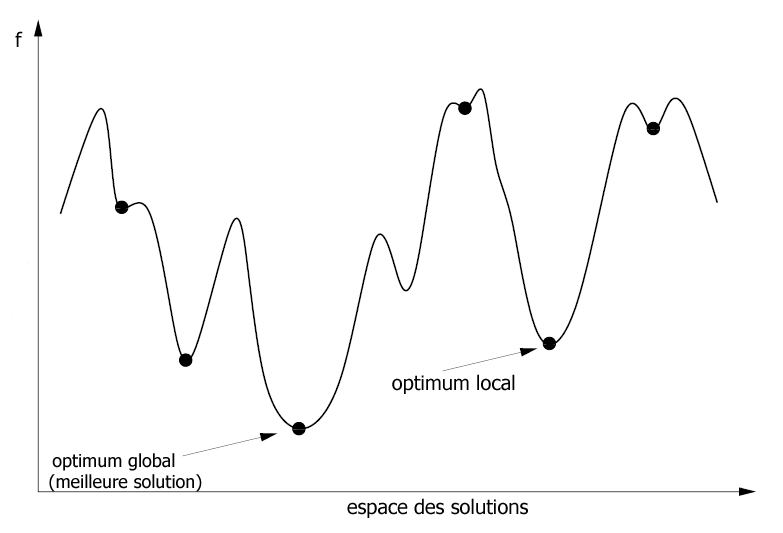
*Une solution s**S est un minimum local relativement à la structure de voisinage N si :*

*f (s) ≤ f (s’) s’* *N(s).*

*Une solution s* *S est un minimum global si f(s) ≤ f (s’) s**’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 méta- heuristiques 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*

***3.3) Les méthodes de recherche locale :***

***3.3.1) La méthode de descente :(hill climbing)***

*Le principe de la méthode de descente (dite aussi basic local search) ou bien (Méthode de voisinage) consiste à partir d’une solution s et à choisir une solution s’ dans un voisinage de s, telle que s’améliore la recherche (généralement telle que f (s’) < f(s)).*

*On peut décider soit d’examiner toutes les solutions du voisinage et prendre la meilleure de toutes (ou prendre la première trouvée), soit d’examiner un sous-ensemble du voisinage.*

*La méthode de recherche locale la plus élémentaire est la méthode de descente. On peut la schématiser comme suit[13] :*

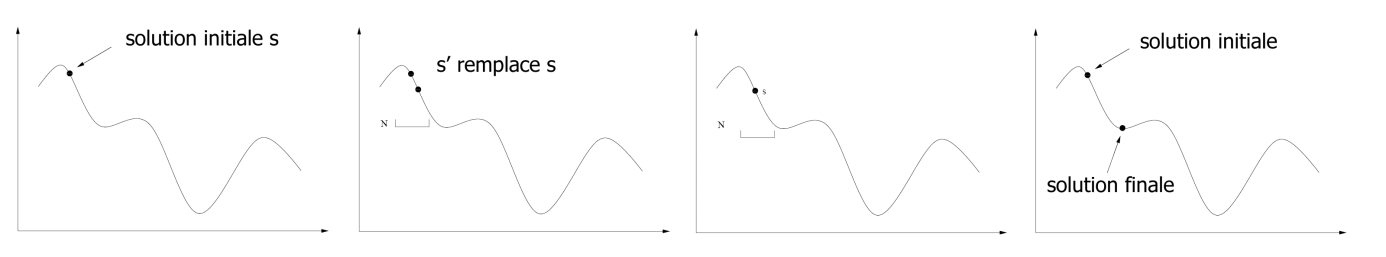
*Procédure descente\_simple (solution initiale s)  
Répéter :*

*Choisir s’ dans N(s)*

*Si f (s’) < f(s) alors s ← s’*

*Jusqu’à ce que f(s’) ≥ f(s), s’**S*

*Fin*

**

*Schéma 3 : évolution d’une solution dans la méthode de descente*

*On peut varier cette méthode en choisissant à chaque fois la solution s’ dans N(s) qui améliore le plus la valeur de f. C’est la méthode de plus grande descente.*

***3.3.1.1) inconvénients :***

*En général, l’efficacité des méthodes de recherche locale simples (descente, ou plus grande descente) est très peu satisfaisante. D’abord, par définition, la recherche s’arrête au premier minimum local rencontré, c’est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l’algorithme en partant d’un jeu de solutions initiales différentes, mais la performance de cette technique décroît rapidement.*

*En revanche, autoriser de temps à autre une certaine dégradation des solutions trouvées, afin de mieux explorer tout l’espace des configurations, a conduit au développement des deux méthodes que nous explorons aux paragraphes suivants, à savoir le recuit simulé et la méthode Tabou.*

***3.3.1.2) Avantages :***

*Le principal avantage de la recherche locale simple est évidemment sa grande simplicité de mise en œuvre: la plupart du temps, elle ne fait que calculer f(s+i)-f(s), où i correspond à un déplacement élémentaire, et si cette expression peut se simplifier algébriquement, alors on pourra évaluer très rapidement cette différence.*

*Il est important de remarquer également l’importance du choix de la fonction de voisinage N : un minimum local pour une certaine structure de voisinage ne l’est pas forcément pour une autre. C’est d’ailleurs ce constat qui est à l’origine de la méthode dite de recherche par voisinage variable, qui repose sur la construction de solutions s parmi plusieurs voisinages Ni, plutôt que dans un seul.*

***3.3.2) La méthode du recuit simulé :******(simulated annealing)***

*Le recuit[[2]](#footnote-3) simulé est souvent présenté comme la plus ancienne des méta-heuristiques, en tout cas, la première à mettre spécifiquement en œuvre une stratégie d’évitement des minima locaux (Kirkpatrick, Gelatt, Vecchi, 1983).*

*Elle s’inspire d’une procédure utilisée depuis longtempspar les métallurgistes qui, pour obtenir un alliage sans défaut, chauffent d'abord à blanc leur morceau de métal, avant de laisser l'alliage se refroidir très lentement (technique du recuit). Pour simuler cette évolution d’un système physique vers son équilibre thermodynamique à une température T, la méthode du recuit simulé [[3]](#footnote-4)exploite l'algorithme de Métropolis.*

*Dans l'algorithme de Métropolis, on part d'une configuration donnée, et on fait subir au système une modification élémentaire. Si cette perturbation a pour effet de diminuer la fonction objectif (ou énergie) du système, elle est acceptée. Sinon, elle est acceptée avec la*

*Probabilité exp(ΔE/T). En appliquant itérativement cette règle, on engendre une séquence de configurations qui tendent vers l'équilibre thermodynamique.*

*On peut systématiser l’algorithme avec le pseudo-code suivant[13] :*

*Procédure recuit\_simulé (solution initiale s)*

*Poser T ← T0*

*Répéter :*

*Choisir aléatoirement s’**N(s)*

*Générer un nombre réel aléatoire r dans [0,1]*

*Si  alors Poser s ← s’*

*Mettre à jour T*

*Jusqu’à ce que le critère de terminaison soit satisfait*

*Fin*

*Et voici l’interprétation de son fonctionnement :*

*• Si f (s’) <f(s) alors  >1, donc r est toujours inférieur à cette valeur, et on accepte la solution s’ (une meilleure solution est donc toujours acceptée, ce qui paraît logique).*

*• Si f (s’)>f(s) et T est très grand, alors* *1, et on il y a de fortes chances d’accepter s’ (bien que la solution s’ soit plus « mauvaise » que s !)*

*• Si f (s’)>f(s) et T est très petit, alors* *0, et on va donc probablement refuser s’*

*Dans un premier temps, T étant généralement choisi très grand, beaucoup de solutions, même celles dégradant la valeur de f, sont acceptées, et l'algorithme équivaut à une visite aléatoire de l'espace des configurations. Mais à mesure que la température baisse, la plupart des mouvements augmentant l'énergie sont refusés, et l'algorithme se ramène à une amélioration itérative classique. A température intermédiaire, l'algorithme autorise de temps en temps des transformations qui dégradent la fonction objectif. Il laisse ainsi une chance au système de s'extraire d'un minimum local.*

*Cet algorithme est parfois amélioré en ajoutant une variable qui mémorise la meilleure valeur rencontrée jusqu’à présent (sans cela, l’algorithme pourrait converger vers un*

*Certaine solution s, alors qu’on avait visité auparavant une solution s’ de valeur inférieure à f(s) !)*

### *3.3.2.1) Paramétrage*

*Le choix de la fonction de température est déterminant.*

*Tout d’abord, le choix de la température initiale : il dépend de la qualité de la configuration de départ. Si cette configuration est choisie aléatoirement, il faut une température relativement élevée. Si au contraire, la solution de départ est déjà assez bonne, parce qu’elle a été choisie par un expert du problème considéré par exemple, une température initiale assez basse sera suffisante.  
On fait souvent suivre à T une loi géométrique décroissante : , avec α = 0.9 par exemple*

*T peut décroître linéairement, à chaque itération, mais on peut aussi envisager une décroissance par paliers, c'est-à-dire en gardant T constant, tant qu’un certain nombre de conditions n’ont pas été remplies, de façon à atteindre l’équilibre thermodynamique.*

*Une fois cet équilibre atteint, on abaisse légèrement la température et on recommence une nouvelle chaîne de calculs à ce nouveau palier.  
 Dans la pratique, le processus est stoppé lorsque le système s’est figé, c'est à dire lorsque la température a atteint la valeur nulle ou bien lorsque plus aucun mouvement accroissant l'énergie n'a été accepté au cours du palier : par exemple, lorsque trois paliers successifs de température ont été descendu sans qu’aucune solution nouvelle n’ait pu être trouvée.*

*Le choix de  pourra se faire expérimentalement, préalablement au processus d’optimisation proprement dit. En calculant l’évolution du système pendant un temps très limité, on peut acquérir une connaissance sommaire de l’espace des configurations, à partir de laquelle on détermine : en maintenant T à 0, on estime la moyenne des variations d’énergie Δ=f (s’)-f(s), et on pose  , avec r>>1 (r=10 typiquement)*

***3.3.2.2) Avantages et inconvénients :***

*Le recuit simulé présente l’avantage d’offrir des solutions de bonne qualité, tout en restant simple à programmer et à paramétrer. Il offre autant de souplesse d’emploi que l’algorithme de recherche local classique : on peut inclure facilement des contraintes dans le corps du programme.*

*Aarts, Korst et Laarhoven (1997) ont par ailleurs démontré que, sous certaines conditions de décroissance de la température, l’algorithme du recuit simulé converge en probabilité vers un optimum global lorsque le nombre d’itérations tend vers l’infini.*

*L’un des inconvénients du recuit simulé est qu’une fois l'algorithme piégé à basse température dans un minimum local, il lui est impossible de s'en sortir tout seul.*

*Plusieurs solutions ont été proposées pour tenter de résoudre ce problème, par exemple en acceptant une brusque remontée de la température, de temps en temps, pour relancer la recherche sur d’autres régions plus éloignées. Il est également possible d’empêcher la température de descendre trop bas : on lui donne une valeur minimale au delà de laquelle on ne change plus de palier de température. Mais si cette valeur est trop grande, l'algorithme passera son temps à augmenter et diminuer son énergie car il acceptera trop de perturbations dégradantes et il n'arrivera pas à explorer à fond une vallée.*

*Ainsi, il est fort possible que l’algorithme arrive à "trouver" la vallée dans laquelle se cache un minimum global, mais il aura beaucoup de mal à l'explorer et donc risque de s'en éloigner sans avoir décelé la solution au problème…*

*C’est là le principal problème du recuit : le paramétrage de la température peut être rebutant et très empirique. On constate également qu’il faut une diminution de la température « suffisamment lente », donc un certain nombre d’itérations, pour voir la solution s’améliorer.*

*Corrélativement, le temps de calcul devient parfois excessif avec certaines applications, et l’algorithme exige en particulier le calcul d’une exponentielle.*

***3.3.2.3) Applications :***

*L’algorithme du recuit simulé a montré son efficacité sur les problèmes combinatoires classiques, surtout sur les échantillons de grande taille. Par exemple, l’expérience a montré qu’il ne devenait vraiment efficace sur le problème du voyageur de commerce qu’au-delà d’environ 800 villes. Il a été testé avec succès sur le problème du partitionnement de graphe. Des analyses ont montré qu’il était efficace avec certaines catégories de problème où l’ensemble des solutions possèdent certaines propriétés particulières. Ceci expliquerait le succès du recuit simulé dans le domaine du placement des circuits électroniques, où il est très employé.*

***3.3.3) La méthode Tabou :******(Tabu Search)***

*La méthode Tabou est une technique de recherche dont les principes ont été proposés pour la première fois par Fred Glover dans les années 80, et elle est devenue très classique en optimisation combinatoire. Elle se distingue des méthodes de recherche locale simples par le recours à un historique des solutions visitées, de façon à rendre la recherche un peu moins « aveugle ». Il devient donc possible de s’extraire d’un minimum local, mais, pour éviter d’y retomber périodiquement, certaines solutions sont bannies, elles sont rendues « taboues ».*

*A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution voisine*  *à chaque itération, Tabou examine un échantillonnage de solutions de N(s) et retient la meilleure s’ même si f (s’)>f(s). La recherche Tabou ne s'arrête donc pas au premier optimum trouvé.*

*Le danger serait alors de revenir à s immédiatement, puisque s est meilleure que s’. Pour éviter de tourner ainsi en rond, on crée une liste T qui mémorise les dernières solutions visitées et qui interdit tout déplacement vers une solution de cette liste. Cette liste T est appelée liste Tabou****.***

### *3.3.3.1) Mémoire à court terme :*

*Mémoriser les configurations telles quelles peut s’avérer extrêmement gourmant en mémoire, et on préférera plutôt conserver une caractéristique, un attribut de chaque configuration, typiquement, la valeur de la fonction objectif prise en chaque configuration. Mais il serait absurde d’interdire définitivement une valeur. Plusieurs configurations différentes peuvent en effet donner la même valeur de fonction objectif, et l’on risquerait ainsi de s’interdire des chemins qui permettraient de s’échapper vers autre vallée, plus profonde et prometteuse. Les solutions ne demeurent donc dans T que pour un nombre limité d’itérations.*

*Une première manière, simple, de réaliser cette mémorisation consiste à construire une table de hachage sous la forme d’un tableau d’entiers, dans lequel on vient stocker une à une les valeurs des solutions que l’on a visitées, comme dans une pile FIFO. Pour autoriser de nouveau une certaine valeur, on enregistre le numéro de l’itération à partir duquel la solution sera de nouveau autorisée, c’est-à-dire la valeur actuelle de l’itération, à laquelle on ajoute une certaine constante t. On évite ainsi la formation de tout cycle de longueur au moins t:*

*Soit T un tableau d’une certaine taille maximale.*

*Si  est la valeur supposée entière de la solution s à l’itération k, on place la valeur k + t dans T [].*

*Il faudra attendre d’arriver à l’itération k + t, pour que la solution de valeur soit autorisée de nouveau.*

*Cette manière de procéder ne marche que si f présente un grand éventail de valeurs différentes, ce qui n’est pas forcément le cas en optimisation combinatoire.*

*On peut donc utiliser autre chose que la fonction objectif comme fonction de hachage, par exemple, on peut recourir à une combinaison de sommes et de produits calculée à partir des coordonnées de chaque solution (si les solutions peuvent s’écrire sous forme d’un vecteur).*

*On peut également utiliser plusieurs fonctions de hachage simultanément.*

*Plutôt que de mémoriser des* ***attributs****, une autre technique consiste à mémoriser les* ***changements*** *opérés dans le passage d’une configuration à une autre, particulièrement dans les problèmes combinatoires où les mouvements peuvent se représenter sous forme de permutations élémentaires.*

*Par exemple, dans l’évaluation de la liste des trajets suivant :*

*A B C D* *B A C D* *B C A D*  *D C A B**A C D B* *A B D C*

*… on aura mémorisé la liste de permutation interdites : BA, CA, DB, DA, et CB*

*Mais il faut remarquer que, ce faisant, on ne s’empêche pas de revenir à une configuration déjà visitée par un autre chemin. Ainsi, dans l’exemple ci-dessus, on voit qu’il est possible de revenir à ABCD, en inversant C et D dans le dernier trajet ABDC… Plutôt que de mémoriser des permutations interdites, on peut alors décider de mémoriser plutôt des couples de positions élémentaires, par exemple en s’interdisant de remettre l’élément i à la place pi, et l’élément j à la place pj.*

*Par ailleurs, en s’interdisant ainsi certaines permutations, un mouvement qui aboutirait à une solution meilleure que toutes celles déjà visitées risqueraient d’être écarté. Pour éviter cela, on incorpore dans l’algorithme des critères d’aspiration, qui autorisent certains mouvements, bien qu’interdits par la liste Tabou, parce qu’on suppose qu’ils vont améliorer la recherche. Typiquement, une solution s est toujours autorisée si sa valeur f(s) est meilleure que toutes les solutions rencontrées jusqu’ici.*

*L’algorithme général peut se représenter avec le pseudo-code suivant :*

*Si on nomme NT (s) toutes les solutions qui ne sont pas taboues ainsi que celles qui le sont mais dont le statut tabou est levé en raison des critères d’aspiration :*

*NT(s) = {s’**N(s) tel que s’**T ou f (s’) < f(s\*)}*

*Procédure methode\_Tabou (solution initiale s)*

*Poser T ←* *et s\* ← s;*

*Répéter*

*Choisir s’qui minimise f(s’) dans NT(s)*

*Si f (s’) < f(s\*) alors poser s\* ← s’*

*Poser s ← s’ et mettre à jour T*

*Jusqu’à ce que le critère de terminaison soit satisfait*

*Fin[13]*

*Comme critère d’arrêt on peut par exemple fixer un nombre maximum d’itérations sans amélioration de s\*, ou bien fixer un temps limite après lequel la recherche doit s’arrêter.*

*S’il était statique dans les premières implémentations sur la méthode Tabou dans les années 80, le nombre d’itérations pendant lesquelles un mouvement est interdit varie généralement au cours du temps, de façon aléatoire, ou en fonction de certains critères.*

### *3.3.3.2) Application au problème d’ordonnancement de tâches*

*Supposons que nous disposions de 6 machines et de 6 pièces, et que chaque pièce doive passer une fois sur chaque machine.  
Pour fixer les idées, supposons de plus que les temps de passage en seconde par pièce et par machine soient les suivants :*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *M1* | *M2* | *M3* | *M4* | *M5* | *M6* |
| *P1* | *29* | *78* | *9* | *36* | *49* | *11* |
| *P2* | *43* | *28* | *90* | *69* | *75* | *46* |
| *P3* | *85* | *91* | *74* | *39* | *33* | *10* |
| *P4* | *71* | *81* | *95* | *92* | *99* | *43* |
| *P5* | *6* | *22* | *14* | *98* | *90* | *61* |
| *P6* | *47* | *2* | *84* | *95* | *6* | *52* |

*Une première solution, évidente, consiste à faire passer sur chaque machine les pièces dans l’ordre des pièces : la machine 1 reçoit la pièce 1, puis la pièce 2, puis la pièce 3, etc. Dès que la pièce 1 a été usinée sur la machine M1, elle passe sur la machine 2, qui reçoit ensuite la pièce 2 lorsque celle-ci a été rendue par la machine 1, et ainsi de suite…*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *Séquences des pièces* | | | | | |
| *M1* | *1* | *2* | *3* | *4* | *5* | *6* |
| *M2* | *1* | *2* | *3* | *4* | *5* | *6* |
| *M3* | *1* | *2* | *3* | *4* | *5* | *6* |
| *M4* | *1* | *2* | *3* | *4* | *5* | *6* |
| *M5* | *1* | *2* | *3* | *4* | *5* | *6* |
| *M6* | *1* | *2* | *3* | *4* | *5* | *6* |

*On calcule le coût de production correspondant à cette solution, f(i) = 1638.*

*Définissons notre voisinage comme l’ensemble des permutations qu’il est possible de faire, sur une machine, entre deux pièces, tout en respectant les contraintes d’ordonnancement.*

*Une implémentation de l’algorithme Tabou pourrait être :*

*Posons i\* = i et OptCourant = 1638*

*Tant qu’il ne se passe pas 100 itérations consécutives sans amélioration de i\**

*Déterminer l’ensemble des permutations autorisées N(i)*

*Poser SolOptLoc = 100000 (une très grande valeur arbitraire)*

*Pour chaque permutation i’ dans N(i), faire :*

*Si f(i’) < SolOptLoc et que la permutation n’est pas tabou, ou que f(i’) < OptCourant, alors on retient la permutation et on pose SolOptLoc = f(i)*

*Modifier la solution courante i avec la meilleure permutation trouvée (meilleure solution voisine)*

*Ajouter cette permutation dans la matrice Tabou (elle sera taboue pour les 6 prochaines itérations)*

*Si f(i)< OptCourant , alors i\* = i et OptCourant = f(i)*

*Fin tant que*

*Une liste Tabou pour un tel problème serait le tableau :*

|  |  |  |  |
| --- | --- | --- | --- |
| *Permutation* | | *Machine* | *Itération* |
| *5* | *6* | *4* | *7* |
| *6* | *5* | *4* | *7* |
| *5* | *4* | *4* | *8* |
| *4* | *5* | *4* | *8* |
| *3* | *4* | *2* | *9* |
| *4* | *3* | *2* | *9* |

*Par exemple, ici, on a commencé par échanger les pièces 5 et 6 sur la machine 4. Tout mouvement entre ces 2 pièces est donc prohibé jusqu’à la 7ème itération.*

### *3.3.3.3) Mémoire à long terme :*

*La prohibition de certains mouvements a un impact négatif : si le nombre t de mouvements interdits est trop court, il y a risque de cyclage, mais s’il est trop grand, cela peut perturber la recherche en empêchant l’algorithme d’exploiter en profondeur une vallée.*

*Une mémoire statistique, basée sur la fréquence d’apparition de certains mouvements, permet de conserver un nombre de mouvements interdits raisonnables, tout en prévenant la formation de cycle de grande largeur. Plutôt que d’interdire un mouvement trop souvent utilisé, on va le pénaliser.*

*En plus de mémoriser la fréquence d’apparition, on peut également invoquer la récence d’une solution (plus un mouvement a été récemment utilisé, plus il est pénalisé), ou bien mémoriser la qualité des mouvements : plus un mouvement a débouché sur des solutions améliorant significativement la valeur de la fonction objectif, plus il est favorisé. Cette dernière technique est connue sous le nom de mémoire adaptative, et a été développée par Rochat et Taillard. On peut l’employer par exemple dans le problème de tournées de véhicule, en mémorisant les routes qui ont donné les meilleurs résultats jusqu’à présent.*

### *3. 3.3.4) Avantages et inconvénients :*

*La méthode Tabou est une méthode de recherche locale, et la structure de son algorithme de base est finalement assez proche de celle du recuit simulé, donc on passera donc de l’un à l’autre facilement, avec l’avantage, par rapport au recuit simulé, d’avoir un paramétrage simplifié : dans un première temps, le paramétrage consistera d’abord à trouver une valeur indicative t d’itérations pendant lesquelles les mouvements sont interdits. Il faudra également décider d’une stratégie de mémorisation à long terme – sur la qualité des solutions, sur leur récence, ou sur leur qualité…*

*En revanche, la méthode Tabou exige une gestion de la mémoire de plus en plus lourde à mesure que l’on voudra raffiner le procédé en mettant en place des stratégies de mémorisation complexe.*

*L’efficacité de la méthode Tabou fait qu’elle est largement employée dans les problèmes d’optimisation combinatoire : elle a été testée avec succès sur les grands problèmes classiques (voyageur de commerce, ordonnancement d'ateliers) et elle est fréquemment appliquée sur les problèmes de constitution de planning, de routage, d’exploration géologique, etc.*

***3.3.4) Les algorithmes d’acceptation avec seuil : (Threshold algorithms)***

*Ces algorithmes sont des variantes du recuit simulé : la différence se situe au niveau de l'acceptation de dégradation à chaque étape. Dans le RS, cette décision est prise selon le critère de Metropolis.*

*Dans un algorithme d'acceptation avec seuil, une telle décision est prise de manière déterministe. A chaque itération k, l'acceptation d'un voisin s’* *N(s) se base uniquement sur une fonction auxiliaire r (s', s) et un seuil Tk : s’est accepté si r (s', s) < Tk.*

*La fonction r (s', s) et le seuil Tk peuvent être définis de nombreuses manières. Dans le cas le plus simple, la fonction r (s', s) est définie par*  *f = f (s') - f(s). Le paramètre de seuil est défini de manière analogue à la température T du recuit simulé : il est initié à une valeur élevée puis décroît progressivement à chaque fois qu'un nombre prédéterminé d'itérations est effectué. Les seuils ainsi générés correspondent à une suite de valeurs positives décroissantes*

*T1* *T2**…* *Tk-1* *Tk* *0 et Tk*  *0. L'idée est de diminuer petit à petit la chance d'accepter des configurations qui dégradent la fonction de coût.*

*Quand Tk tend vers 0, l'algorithme réalise une recherche de descente aléatoire.*

*Ces algorithmes ont été utilisés pour résoudre des problèmes d'optimisation et ont obtenu des résultats intéressants [8, 9,10]. La difficulté essentielle de cette approche se situe au niveau de la détermination des seuils pour une application donnée.*

***3.3.5) Méthode du bruitage :***

*La méthode de bruitage a été introduite sur des problèmes dont la donnée comporte un ensemble de nombres réels [11], par exemple, le voyageur de commerce (arêtes vallées par des réels) et le problème d'affectation quadratique (matrices de réels). Cette méthode fait appel à une notion de bruitage de la donnée qui est définie de la façon suivante : la donnée bruitée est produite à partir de la donnée initiale en ajoutant à chacun des réels une composante calculée comme le produit de trois éléments :*

*a) une fonction aléatoire à valeurs sur l'intervalle [0,1],*

*b) un paramètre permettant de contrôler le niveau du bruit,*

*c) le plus grand des réels Concernés, afin de normaliser le niveau du bruit par rapport à la donnée.*

*L'exécution de l'heuristique comporte plusieurs étapes. Chaque étape consiste à calculer un bruitage de la donnée, puis à effectuer une descente prenant en compte la fonction de coût calculée à partir de la donnée bruitée. Le niveau du bruit est décrémenté au début de chaque nouvelle étape, et la descente s'effectue à partir de la configuration résultant de l'étape précédente. Il existe deux variantes pour le bruitage :*

1. *chaque descente sur la donnée bruitée est suivie par une descente effectuée sur la donnée non bruitée.*

*L'objectif consiste à mieux tenir compte de la donnée réelle puisqu'un véritable optimum local est alors atteint ;*

*2) la configuration courante est régulièrement remplacée par la meilleure configuration obtenue depuis le début.*

*Cette méthode a été appliquée avec succès au problème du voyageur de commerce et a obtenu de très bons résultats selon les auteurs [11,12].*

1. ***Conclusion :***

*Les méthodes de recherche locale constituent une classe de la méta-heuristique. Qui est par la suite appartient des méthodes approchées adaptables à un très grand nombre de problèmes combinatoires et de problèmes d'affectation sous contraintes.*

*Elles ont révélé leur grande efficacité pour fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes d'optimisation classiques et d'applications réelles de grande taille. Si pour cela on vu l'étude de ces méthodes est actuellement en plein de développement.*

*Ainsi, La qualité des solutions trouvées par les méthodes de recherche locale dépend de leur paramétrage (il faut éviter que les algorithmes ne convergent trop rapidement vers un optimum local), et de l’équilibre à trouver entre un balayage de tout l’espace des solutions (diversification) et une exploration locale poussée (l’intensification). Le choix d’une bonne représentation, d’un bon voisinage, sont également, nous l’avons dit, des facteurs influençant grandement l’efficacité de la méthode choisie, quelle qu’elle soit.*

*Finalement, on vu que les différentes méta-heuristiques que nous avons examinées, descente, recuit simulé, Tabou, bruitage et avec seuil sont maintenant perçues comme des points de départ au développement de nouveaux algorithmes d’optimisation. Qui Nous présentons également les possibilités de combiner différentes méthodes pour créer des méthodes hybrides.*

***Bibliographie :***

*[1]* *C.H. PAPADIMITRIOU, K. STEIGLITZ, Combinatorial optimization – algorithms and complexity. Prentice Hall, 1982.*

*[2] C.C. RIBEIRO, N. MACULAN (Eds.), Applications of combinatorial optimization.*

*Annals of Operations Research 50, 1994.*

*[3] M.R. GAREY, D.S. JOHNSON, Computers and intractability: a guide to the theory of NP-completeness, W.H. Freeman and Company, New York, 1979.*

*[6] Y.collette and P.Siarry. Optimisation multiobjectif. Eyrolles 2002.*

*[7] rapport-méta-heuristique-optimisation combinatoire.*

*[8]* *G. DUCEK, T. SCHEUER, Threshold accepting: a general purpose optimization algorithm. Journal of Computational Physics 90: 161-175, 1990.*

*[9]* *G. DUCEK, New optimization heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104:86-92, 1993.*

*[10]* *M. SINCLAIR, Comparison of the performance of modern heuristics for combinatorial problem on real data. Computers & Operations Research 20: 687-695, 1993.*

*[11]* *I. CHARON, O. HUDRY, The noising method: a new method for combinatorial optimization, Operations Research Letters 14: 133-137, 1993.*

*[12]* *I. CHARON, O. HUDRY, Mixing different component of metaheuristics. I.H. Osman, J.P. Kelly (Eds.), Metaheuristics, Theory and Applications, Kluwers Academic Publishers, p. 589-604, 1995.*

***Site web****:*

*[4]*[*http://ww2.eivpparis.fr/chachoua/TER2007/%5BN%B027%5D\_RapportTER\_LE\_GALL\_ROBIN*](http://ww2.eivpparis.fr/chachoua/TER2007/%5BN%B027%5D_RapportTER_LE_GALL_ROBIN)*.*

*pdf*

*[5]* [*http://www.irit.fr/~Philippe.Muller/Cours/Optim\_cb/cours\_0809.pdf*](http://www.irit.fr/~Philippe.Muller/Cours/Optim_cb/cours_0809.pdf)

*[13]* *The Metaheuristic Network*

[*http://www.metaheuristics.net/*](http://www.metaheuristics.net/)

*[14] Laboratoire Cedric. CNAM.*

[*http://cedric.cnam.fr/AfficheEquipe.php?id=2*](http://cedric.cnam.fr/AfficheEquipe.php?id=2)

*[15]* [*http://artur.univ-fcomte.fr/ST/INFOR/these/devarenne\_protege.pdf*](http://artur.univ-fcomte.fr/ST/INFOR/these/devarenne_protege.pdf)

*[16]* *http://www.info.univ-angers.fr/pub/hao/papers/RIA.pdf*

*[17]*  *Simulated Annealing Information. Taygeta Scientific Inc.*

*http://www.taygeta.com/annealing/simanneal.html*

1. Revue d'Intelligence Artificielle, 1999 où ils réalisent une synthèse des principales méta-heuristiques

   JIN-KAO HAO, PHILIPPE GALINIER et MICHEL HABIB proposent cette définition pour une méta-heuristique [↑](#footnote-ref-2)
2. La méthode du recuit tire son nom de l'analogie avec un processus industriel de fabrication d'alliages métalliques qui consiste à refroidir le métal en fusion par paliers de température successifs de façon telle à lui assurer une structure sans défaut. [↑](#footnote-ref-3)
3. Méthode imite le procédé de fabrication, notamment l'un des paramètres de la méthode joue le rôle de la température et reçoit ce nom. La température décroît par paliers successifs. [↑](#footnote-ref-4)