Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option: *Modèle Intelligent et Décision(M.I.D)*

Thème

Application des méthodes méta-heuristiques pour l'apprentissage de structure des réseaux bayésiens

Réalisé par :

- Alioua Mohammed Amine

Présenté le 14 septembre 2017 devant le jury composé de.

- *Mr Benazzouz Mourtada .* *(Président)*
- Mr *Benmouna Youcef .* *(Encadreur)*
- M*r Boudefla Amine .* *(Examinateur)*

Année universitaire : 2016-2017

Remerciements

Je remercie dieu le tout puissant de m'avoir donné le privilège et la chance d'étudier et de suivre le chemin de la science.

J'adresse mes vifs remerciements à mon encadreur ***Mr. Benmouna Youcef****, pour les orientations et les conseils qu'il m'a prodigué durant l'évolution de mon projet.*

Je tiens également à remercier ***Mr.Benazzouz Mourtada .*** *d'avoir accepté de présider le jury de mon projet de fin d'étude.*

Aussi je remercie Mr ***Boudefla Amine*** *.Celui qui a bien voulu examiner mon travail. sa présence va valoriser, de manière certaine, le travail que j'ai effectué.*

J'adresse également ma profonde gratitude envers les professeurs de l'université ABOU BEKR BELKAID, ceux du département de l'informatique en particulier.

Enfin je remercie tous ceux qui m'ont aidé de près ou de loin dans la réalisation de ce projet de fin d'étude.

Merci à tous

Dédicaces

Je dédie ce mémoire a la flamme du souvenir de mon père que dieux ai son âme… au premier homme de ma vie, a celui qui m'appris que la vie est un pari et que chaque être doit le gagner avec son travail, son savoir et aussi courage a toi mon père avec tous mes souvenirs

A mon model d'honneur et de principes, celle qui s'est toujours dévouée et sacrifiée pour moi, celle qui a su être bonne, gentille et compréhensive avec moi, celle qui m'a aidée du mieux qu'elle pouvait pour réussir, celle qui a toujours été là dans mes moments de détresse, ma très chère et adorée mère.

A mes chères sœurs et frère ainsi qu'à toute ma famille.

A mes collègues de travail et a tous les Présidents et Procureurs et les juges du tribunal de nedroma

A tous les informaticiens de la DGMJ, a tous les juges de la promo 23 et 24 et 25.

Tous mes enseignants du primaire à l'université.

A tous ceux que j'aime et qui m'aiment.

A tous mes amis.

Table des Matières

CHAPITRE III : CONCEPTION ET IMPLEMENTATION

Introduction Générale

Introduction générale :

Quelque soit son domaine, l'être humain est confronté à différents problèmes dans toutes les sphères de la société. Un problème donné peut être défini par l'ensemble des propriétés que doivent vérifier ses solutions. Il peut être un problème de décision ou un problème d'optimisation. Un problème de décision peut se ramener à un problème d'existence de solution. Il consiste à répondre à la question « Est-ce qu'il existe une solution basée sur un ensemble d'énoncés et qui satisfait un ensemble de contraintes ? » par une des réponses: « *Oui il existe* » ou bien « *Non il n'existe pas* ». Par contre, un problème d'optimisation peut se ramener à un problème d'existence de solution de bonne qualité. Il consiste à rechercher une solution de qualité suffisante au regard d'un (des) critère(s) donné(s) et des objectifs à satisfaire. Les problèmes d'optimisation peuvent être partagés en deux catégories: des problèmes académiques et des problèmes industriels. En fait, les problèmes industriels sont des projections de problèmes académiques sur le plan de la pratique.

Dans notre projet le travail présenté concerne les modèles graphiques probabilistes (Réseaux Bayésiens), qui sont d'une part des modèles de représentation de la connaissance et de l'incertitude et d'autre part des modèles de raisonnement permettant de propager des observations.

La complexité du problème Réseaux Bayésiens est liée à deux facteurs : le temps de calcul nécessaire pour sa résolution et l'espace de mémoire requis. Ces deniers (temps et mémoire) augmentent en fonction de la taille du problème traité.

Le nombre de graphe acyclique dirigé (DAG) pour représenter toutes les structures possibles pour les Réseaux Bayésiens est de taille super-exponentielle.

Une solution nous est proposée dans ce mémoire et l'un des algorithmes inspiré de la nature qui est la Recherche COUCOU « en anglais : CUCKOO SEARCH » et la

Recherche LUCIOLE « en anglais : FIREFLY SEARCH » pour améliorer les scores des graphes Bayéseins.

CHPITRE I :

INTRODUCTION

ET

GÉNÉRALITÉS

DES RESEAUX

BAYESIENS

I- Introduction :

Les réseaux bayésiens s'appuient sur un théorème : *Le théorème de Bayes*. C'est un résultat de base en théorie des probabilités, issu des travaux du révérend Thomas Bayes (1702-1761), présenté à titre posthume en 1763.

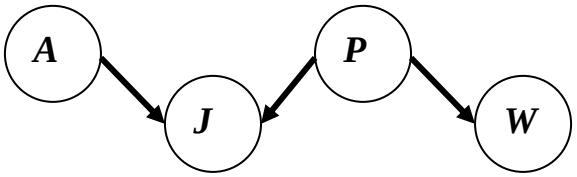
I-1- Exemple :

Nous allons présenter un exemple pour une petite explication concernant un graphe causal, et comment l'information circule au sein du graphe.

Pour cela, nous allons utiliser un exemple, simple dans la littérature sur les réseaux bayésiens, qui a utilisé pour la première fois par Pearl [Pearl, 1988], et repris dans les années 1996 [Finn Jensen, 1996].

Ce matin-là, alors que le temps est clair et sec, M. Holmes sort de sa maison. Il s'aperçoit que la pelouse de son jardin et humide. Il se demande alors s'il a plu pendant la nuit, ou s'il a simplement oublié de débrancher son arroseur automatique. Il jette alors un coup d'œil à la pelouse de son voisin, M. Watson, et s'aperçoit qu'elle est également humide. Il en déduit alors qu'il a probablement plu, et il décide de partir au travail sans vérifier son arroseur automatique.

La représentation graphique du modèle causal par M. Holmes est la suivant :



A J'ai oublié de débrancher mon arroseur automatique.

P Il a plu pendant cette nuit.

J L'herbe de mon jardin est humide.

W L'herbe du jardin de M. Waston est humide.

Figure 1-1 : Représentation graphique du modèle causal par M. Holmes[24]

La lecture du graphe est bien conforme à l'intuition :

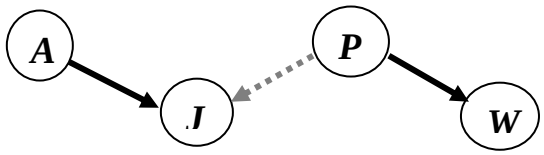
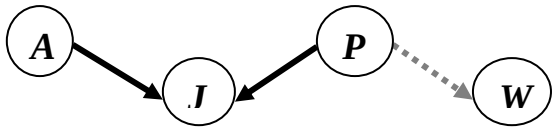
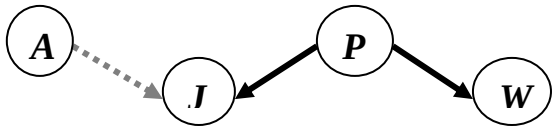
	S'il a plu pendant la nuit, l'herbe de mon jardin est humide.
	S'il a plu pendant la nuit, l'herbe du jardin de M. Watson est également humide.
	Si j'ai oublié de débrancher mon arroseur automatique, l'herbe de mon jardin est humide.

Figure 1-2 : Lecture du graphe du modèle causal par M. Holmes[24]

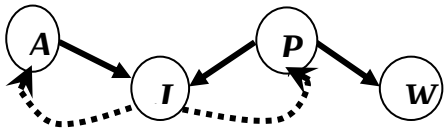
Comment ce graphe est-il utilisé ici pour raisonner ? Autrement dit, comment l'information J, dont on sait qu'elle est vraie, est-elle utilisée ?

Tout d'abord, le modèle nous indique que *J* a du être causé soit par A, soit par *P*.

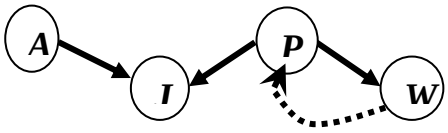
Faute d'information complémentaire, les deux causes sont a *priori* également[1] plausibles[2].

Le fait que *W* soit également vrai renforce la croyance en *P*.

Dans cet exemples simple, on voit que l'information a circulé uniquement dans le sens $effet \rightarrow cause$.

(1) 

La connaissance de *J* renforce la croyance en l'une des deux causes A ou *P*

(2) 

La connaissance de *W* augmente la croyance en la cause *P*. Par rapport à l'état (1) précédent, la cause *A* devient moins plausible.

Figure 1-3 : L'information a circulé uniquement dans le sens $effet \rightarrow cause$. [24]

1- En réalité, cela dépend, bien sur, de la connaissance a priori que M. Holmes a de la météorologie de sa région. Ici, nous supposons qu'il n'en a aucune.

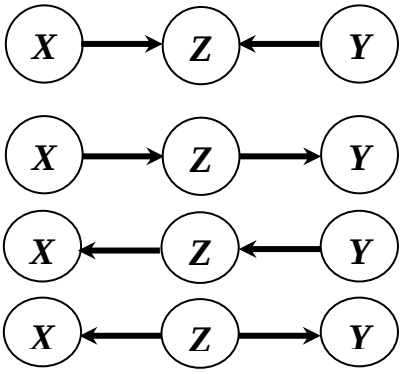
2- Nous utilisons volontairement le mot plausible, au lieu de probable, qui sera utilisé pour la formalisation du raisonnement.

Pour prendre un raccourci, M. Holmes a déduit que son arroseur automatique était à l'arrêt à partir du fait que la pelouse de son voisin était humide !

Cet exemple simple, sur lequel nous n'avons utilisé que du raisonnement de sens commun, nous montre bien que l'information peut suivre des chemins peu intuitifs lorsqu'elle se propage dans un réseau de causalités.

- **Le cas général :**

Nous allons montrer la circulation de l'information dans un graphe causal du point de vue général dans la figure suivant :



Connexion convergente : *X* et *Y* causent *Z*.

Connexion en série : *X* cause *Z*, *Z* cause *Y* (ou le cas symétrique).

Connexion divergente : *Z* cause *X* et *Y*

Figure 1-4 : la circulation de l'information dans un graphe causal[24]

Définition-1 : un Réseau Bayésien[24] est défini par :

- Un graphe orienté sans circuit (DAG) $G = (V, E)$, ou V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G.
- Un *espace probabilisé* fini (Ω, Z, p).
- Un ensemble de variables aléatoires associées aux nœuds du graphe et définies sur (Ω, Z, p), tel que :

$$p(V_1, V_2, \dots, V_n) = \prod_{i=1}^{n} p(V_i | C(V_i)) \qquad \textbf{\textit{Formule I-1}}$$

lorsque les situations sont incertaines ou les données incomplètes la décomposition de la loi jointe permet d'avoir des algorithmes d'inférence puissants qui font des Réseaux Bayésiens des outils de modalisation et de raisonnement très pratique. Pour les problèmes de classification ils sont utiles lorsque les interactions entre les différents critères peuvent être modélisées par des relations de probabilité conditionnelles.

Il est possible d'en faire l'apprentissage à partir d'une base de données lorsque la structure du Réseaux Bayésiens n'est pas fournie *a priori* par un expert. La recherche

structure n'est pas simple d'un Réseaux Bayésiens à cause du fait que l'espace de recherche est de taille super-exponentielle en fonction du nombre de variables.

Nous allons commencer par introduire quelque notions générales sur la structure des Réseaux Bayésiens, la façon de lui associer un score et les propriétés intéressantes de ces scores. Ensuite nous détaillons des méthodes de recherche de structure les plus couramment utilisées, de la recherche de la causalité, au parcours heuristique de l'espace des Réseaux Bayésiens avec les différents problèmes d'initialisation que cela pose. Nous comparerons alors les méthodes grâce à deux séries de teste. La première série de tests permet d'évaluer l'efficacité de ces méthodes à trouver un bon Réseau Bayésien pour des problèmes de classification en utilisant éventuellement certaines connaissances a priori sur la tache à résoudre. Nous conclurons alors sur les avantages et inconvénients des différentes méthodes utilisées et évoquerons certaines techniques prometteuses pour l'apprentissage de structure des Réseaux Bayésien. Nous allons terminer notre chapitre par une problématique qui propose de la recherche de la causalité, au parcours heuristique de l'espace des Réseaux Bayésiens avec les différents problèmes d'initialisation que cela pose on se base sur des algorithmes génétique et précisément sur l'algorithme de recherche Coucou et l'algorithme de recherche Luciole.

II- Généralités :

La première idée pour trouver la meilleure structure d'un Réseau Bayésien, est de parcourir tous les graphes possibles, de leur associer un score, puis de choisir le graphe ayant le score le plus élevé. Une premier approche, proposée initialement par Spirtes et al. D'un coté, et Pearl et Verma de l'autre, consiste à rechercher les différentes indépendances conditionnelles qui existent entre les variables. Les autres approches tentent de quantifier l'adéquation d'un Réseau Bayésien au problème à résoudre, c'est-à-dire d'associer un score à chaque Réseau Bayésien. Puis elles recherchent la structure qui donnera le meilleur score dans l'espace $\mathbb{B}$ des graphes dirigés sans circuits. Une approche exhaustive est impossible en pratique en raison de la taille de l'espace de recherche. la *Formule I-2*[27] démontrée par [Robinson, 1977] prouve que le nombre de structures possibles à partir de n nœuds est super-exponentiel (par exemple, $NS(5) = 29281$ et $NS(10) = 4.2 \times 10^{18}$)

$$NS(n) = \begin{cases} 1, & n = 0 \; ou \; 1 \\ \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} NS(n-i), & n > 1 \end{cases} \qquad \textbf{\textit{Formule I-2}}$$

Pour résoudre ce problème, si le parcours de l'ensemble des *DAG* s'effectue avec des opérateurs du type ajout ou suppression d'arcs, il est avantageux d'avoir un score calculable localement pour limiter les calculs à la variation de ce score entre deux voisins.

Définition-2 : Un score S dit décomposable[10] s'il peut être écrit comme (ou un produit) de mesures qui sont fonction seulement du nœud et de ses parents. En claire, si n est le nombre de nœuds du graphe, pa parents, le score doit avoir une des formes suivant :

$S(\mathcal{B}) = \sum_{i=1}^{n} s\big(X_i, pa(X_i)\big) \;\; ou \; S(\mathcal{B}) = \prod_{i=1}^{n} s(X_i, pa(X_i))$ ***Formule I-3***

Définition-3 : Deux Réseaux Bayésiens $\mathcal{B}_1$ et $\mathcal{B}_2$ sont dit équivalents au sens de Markov[24]$(\mathcal{B}_1 \equiv \mathcal{B}_2)$ s'il représentant les meme relation d'indépendance conditionnelle.

Afin d'illustrer simplement cette notion, montrons que les structures $\mathcal{B}_1$, $\mathcal{B}_2$et $\mathcal{B}_3$ décrites ci-après sont équivalents.

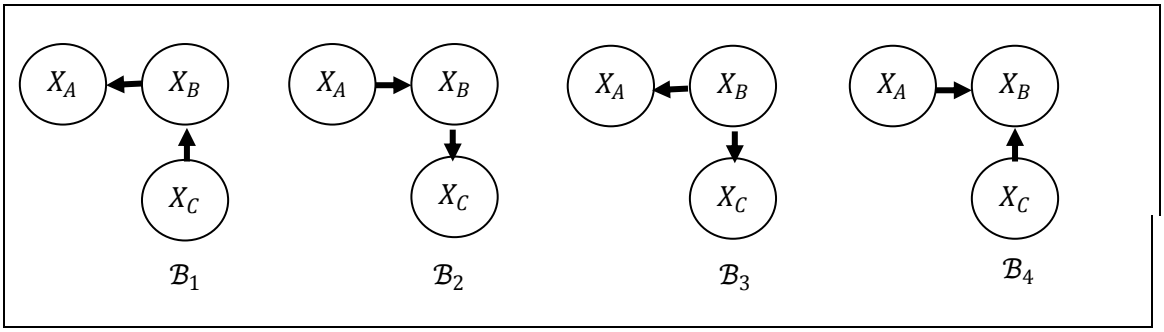


Figure 1-4 : Botion d'équivalents de Markov[24]

Démonstration :

Montrons-le pour $\mathcal{B}_1$ et $\mathcal{B}_2$:

Selon $\mathcal{B}_1$: $P(X_A, X_B, X_C)_{\mathcal{B}_1} = P(X_A|X_B) \times P(X_B|X_C) \times P(X_C)$

Selon $\mathcal{B}_2$: $P(X_A, X_B, X_C)_{\mathcal{B}_2} = P(X_A) \times P(X_B|X_A) \times P(X_C|X_B)$

Mais d'après la définition d'une probabilité conditionnelle

$P(X_A, X_B) = P(X_A|X_B) \times P(X_B) \times P(X_A) \times P(X_B|X_A)$

$P(X_B, X_C) = P(X_C|X_B) \times P(X_B) \times P(X_C) \times P(X_B|X_C)$

Et donc : $= P(X_A|X_B) \times P(X_B) \times P(X_C|X_B)$

$= P(X_A|X_B) \times P(X_B|X_C) \times P(X_C)$

$= P(X_A, X_B, X_C)_{\mathcal{B}_1}$

Les réseaux bayésiens $\mathcal{B}_1$ et $\mathcal{B}_2$ sont donc équivalents (id. avec $\mathcal{B}_3$).

Par contre, ces trois structures ne sont pas équivalentes à la V-structure $\mathcal{B}_4$. En effet, nous avons $P(X_A, X_B, X_C) = P(X_A) \times P(X_C) \times P(X_B|X_A, X_C)$ et le terme $P(X_B|X_A, X_C)$ ne peut pas simplifier.

Définition-4 :

Un score qui associe une même valeur à deux graphes équivalents est dit score équivalent[24]. Les approches suivantes vont soit chercher la structure qui va maximiser un certain score, soit chercher les meilleures structures et combiner leurs résultats.

Par exemples le score AIC [Akaike, 1970], et le score BIC est à la fois décomposable, et équivalent. Il est issu de principes énoncés dans [Schwartz, 1978] et a la forme suivant :

$$Score\ AIC(\mathcal{B}, \mathcal{D}) = logL(\mathcal{D}|\theta^{MV}, \mathcal{B}) - Dim(\mathcal{B}) \qquad \textbf{Formule I-4}$$

$$Score\ BIC(\mathcal{B}, \mathcal{D}) = logL(\mathcal{D}|\theta^{MV}, \mathcal{B}) - \frac{1}{2} Dim(\mathcal{B}) logN \qquad \textbf{Formule I-5}$$

BIC (Bayesian Information Criterion, 1978)

Ou D est notre base d'exemples, θ^{MV} est la distribution des paramètres obtenue par *maximum de vraisemblan*ce pour le réseau $\mathcal{B}$, et ou $Dim(\mathcal{B})$ est la dimension du réseau bayésien, définie comme suit.

Soit X_i un nœud du réseau bayésien de taille r_i, et $pa(X_i)$ ses parents. Le nombre de paramètres nécessaires pour représenter la distribution de probabilité $P(X_i|pa(X_i) = x_j)$ est égal à $r_i - 1$. Pour représenter $P(X_i|pa(X_i))$, il faudra donc $Dim(X_i, \mathcal{B})$ paramètres, avec :

$$Dim(X_i, \mathcal{B}) = (r_i - 1) \prod_{X_j \in pa(X_i)} r_j = (r_i - 1) q_i \qquad \textbf{Formule I-6}$$

Ou q_i est le nombre de configurations possibles pour les parents de X_i. Le nombre de paramètres nécessaires pour représenter toutes les distributions de probabilités du réseau $\mathcal{B}$ est $Dim(\mathcal{B})$:

$$Dim(\mathcal{B}) = \sum_{i=1}^{n} Dim(X_i, \mathcal{B}) = \sum_{i=1}^{n} (r_i - 1) q_i \qquad \textbf{Formule I-7}$$

Le score AIC, BIC est la somme d'un terme de vraisemblance du réseau par rapport aux données, et d'un terme qui pénalise les réseaux complexes. Ces scores est score équivalent car deux graphes équivalents ont la même vraisemblance (ils représentent les mêmes indépendances conditionnelles) et la même complexité.

En utilisant des scores ayant cette propriété, il devient possible de faire de la recherche de structure dans l'espace des équivalents de Markov. Cela s'avère être un

choix judicieux, car là ou un algorithme à base de score dans l'espace des DAG peut boucler sur plusieurs réseaux équivalents, la même méthode utilisé dans l'espace des équivalents sera soit à l'optimum, soit pourra trouver un représentant d'une classe d'équivalence qui augmente le score.

III- les algorithmes :

III-1- L'algorithme PC, recherche de causalité :

Spirates et al [Spirates et al, 2000] ont proposé une variation de SGS L'algorithme PC[24] a été introduire par Peter Spirtes, Clark Glymour, et Richard Scheines [Spirtes, Glymour et Scheines, 1993] détaillé dans l'algorithme 1-1 ci-après qui limite les tests d'indépendance aux indépendances d'ordre $0(X_A \perp X_B)$ puis aux indépendances conditionnelles d'ordre 1 $(X_A \perp X_B|X_C)$, et ainsi de suite.

- *Construction d'un graphe non orienté*
  Soit $\mathcal{G}$ le graphe reliant complétement tous les nœuds $\mathcal{X}$
  $i \leftarrow 0$
  Répéter
    Recherche des indépendances cond. D'orde i
    $\forall (X_A, X_B) \in \mathcal{X}^2 tels\ que\ X_A - X_B\ \ et\ Card(Adj(\mathcal{G}, X_A, X_B)) \geq i$
    $\forall S \subset Adj(\mathcal{G}, X_A, X_B) tel\ que\ Card(S) = i$
      $si\ X_A \perp X_B|S\ alors$
      - *surpression de l'arête $X_A - X_B$ dans $\mathcal{G}$*
      - *SepSet* $(X_A, X_B) \leftarrow SepSet\ (X_A, X_B) \cup S$
      - *SepSet* $(X_B, X_A) \leftarrow SepSet\ (X_B, X_A) \cup S$

    $i \leftarrow i + 1$
  Jusqu'à $Card\big(Adj(\mathcal{G}, X_A, X_B)\big) < i, \forall (X_A, X_B) \in \mathcal{X}^2$
- *Recherche des V-structures*
  $\forall (X_A, X_B, X_C) \in \mathcal{X}^3\ tel\ que\ \neg\overline{X_A X_B}\ et\ X_A - X_C - X_B,$
    $Si\ X_C \notin sepset(X_A, X_B)\ alor\ rajouter\ X_A \rightarrow X_C \leftarrow X_B\ (V - structure)$
- *Ajouter récursif de →*
  Répéter $\forall (X_A, X_B) \in \mathcal{X}^2,$
    $si\ X_A - X_B\ et\ X_A \rightsquigarrow X_B\ alor\ rajouter\ X_A \longrightarrow X_B$
    $si\ \neg\overline{X_A X_B}, \forall X_C\ tel\ que\ X_A \longrightarrow X_C\ et\ X_C - X_B\ alor\ rajouter X_C \longrightarrow X_B$
  Tant qu'il est possible d'orienter des arrêtes.

ALGO 1-1 : algorithme PC

Notion de l'algorithme PC

$\mathcal{X}$	Ensemble de tous les nœuds
$\mathrm{Adj}(\mathcal{G}, \mathrm{X_A})$	Ensemble des nœuds adjacents à $\mathrm{X_A}$ dans $\mathcal{G}$
$\mathrm{Adj}(\mathcal{G}, \mathrm{X_A}, \mathrm{X_B})$	$\mathrm{Adj}(\mathcal{G}, \mathrm{X_A})\{\mathrm{X_B}\}$
$\mathrm{X_A} - \mathrm{X_B}$	Il existe une arête entre $\mathrm{X_A}$ et $\mathrm{X_B}$
$\mathrm{X_A} \longrightarrow \mathrm{X_B}$	Il existe un arc de $\mathrm{X_A}$ vers $\mathrm{X_B}$
$\overline{\mathrm{X_A X_B}}$	$\mathrm{X_A}$ et $\mathrm{X_B}$ adjacents $\mathrm{X_A} - \mathrm{X_B}$, $\mathrm{X_A} \longrightarrow \mathrm{X_B}$ ou $\mathrm{X_B} \longrightarrow \mathrm{X_A}$
$\mathrm{X_A} \rightsquigarrow \mathrm{X_B}$	Il existe un chemin dirigé reliant $\mathrm{X_A}$ et $\mathrm{X_B}$

III-2- restriction à l'espace des arbres :

L'arbre de recouvrement maximal (*Maximum Weight Spanning Tree « **MWST** »)* [24] *:* cette méthode parcourir tous les nœuds et maximise un score définit pour tous les arcs possibles.

Chow et Liu [CL1968] ont proposé d'utiliser un score basé sur un critére d'information mutuelle :

$$W_{CL}(X_A, X_B) = \sum_{a,b} P(X_A = a, X_B = b) \log \frac{P(X_A = a, X_B = b)}{P(X_A = a)P(X_B = b)}$$
$$= \sum_{a,b} \frac{N_{ab}}{N} \log \frac{N_{ab}\, N}{N_a N_B} \qquad \textbf{\textit{Formule I-8}}$$

Heckerman [HGC, 1994] propose d'utiliser un score quelconque, localement décomposable, en définissant le poids d'une arrête par :

$$W(X_A, X_B) = score(X_A, X_B) - score(X_A, \phi) \qquad \textbf{\textit{Formule I-9}}$$

Ou $score(X_A, X_B)$ est le score local en X_A en supposant que X_B est son parent, et $score(X_A, \phi)$ est le score local en X_A en supposant qu'il ne possède aucun parent.

L'orientation de cette arbre non orienté pourrait donc se faire en utilisant l'algorithme 1-2 suivant :

- *Construction de l'arbre optimal(Kruskal)*
  $\forall X_i, \mathcal{T}(X_i) = \{X_i\}$
  $\mathcal{B}^0 \longleftarrow \phi$
  $\forall (X_i, X_j) \in \mathcal{A}$
  $si\ \mathcal{T}(X_i) \neq \mathcal{T}(X_j)\ alor$
  - $\mathcal{B}^0 \longleftarrow \mathcal{B}^0 \cup (X_i, X_j)$
  - $\mathcal{T}' \longleftarrow \mathcal{T}(X_i) \cup \mathcal{T}(X_j)$
  - $\mathcal{T}(X_i) \longleftarrow \mathcal{T}'$
  - $\mathcal{T}(X_j) \longleftarrow \mathcal{T}'$
- *Orientation des arêtes*
  $\mathcal{B} \longleftarrow \phi$
  $\{pa_i\} \longleftarrow parcours\ Profondeur\ (\mathcal{B}^0, X_r)$
  $\forall X_i$
  $si\ pa_i \neq \phi\ alors\ ajout\ de\ pa_i \longrightarrow X_i\ dans\ \mathcal{B}$

ALGO 1-2 : algorithme MWST dirigé

Notion de l'algorithme MWST dirigé

$\mathcal{A}$	Liste des arêtes (X_i, X_j) dans l'ordre décroissant des $\mathcal{W}$
$\mathcal{T}(X_i)$	Arbre passant par le nœud X_i
X_r	Racine choisie pour orienter l'arbre
pa_i	Parent du nœud X_i
$\mathcal{B}^0$	Arbre optimal non orienté
$\mathcal{B}$	Structure finale obtenue par l'algorithme

III-3- l'algorithme K2 :

L'algorithme K2[24] de Cooper et Herskovites [CH, 1992] détaillé dans l'ALGO 1-3, ajoute progressivement des arcs, en ne conservant un arc qui vient d'être ajouté que s'il améliore la performance du réseau suivant une métrique donnée.

Voici comment cet algorithme construit le graphe $X_1 \longrightarrow X_2 \longrightarrow X_3$. Apres avoir commencé avec un réseau sans arc, K2 essaie d'ajouter l'arc $X_1 \longrightarrow X_2$. Comme cet arc améliore la performance, il est conservé. Ensuite, K2 essaie d'ajouter l'arc $X_1 \longrightarrow X_3$, puis l'arc $X_2 \longrightarrow X_3$. C'est ce dernier qui obtient le meilleur score, par rapport à la métrique donnée. Et ainsi de suite.

Pour calculer $P\ (\mathcal{B}_S, \mathcal{D})$ en doit commencer par la remarque suivant :

$$\frac{P(\mathcal{B}_{S1}/\mathcal{D})}{P\ (\mathcal{B}_{S2}/\mathcal{D})} = \frac{\frac{P(\mathcal{B}_{S1}, \mathcal{D})}{P(\mathcal{D})}}{\frac{P\ (\mathcal{B}_{S2}, \mathcal{D})}{P(\mathcal{D})}} = \frac{P(\mathcal{B}_{S1}, \mathcal{D})}{P\ (\mathcal{B}_{S2}, \mathcal{D})}$$

Théorème :

Soit $\mathcal{D}$ la base de données et $\mathcal{N}$ le nombre de cas dans $\mathcal{D}$, et soit $\mathcal{G}$ la structure du réseau[10] sur X. De plus, soit pa_{ij} la j^{ieme} instanciation de $Pa(X_i)$, et $\mathcal{N}_{ijk}$ le nombre de cas dans $\mathcal{D}$ ou X_i a la valeur x_{ik} et que $Pa(X_i)$ est instancié en pa_{ij}. Si $\mathcal{N}_{ij} = \sum_{k=1}^{r_i} \mathcal{N}_{ijk}$ alors

$P(\mathcal{G},\mathcal{D}) = P(\mathcal{G})P(\mathcal{D}|\mathcal{G})\ avec\ P(\mathcal{D}|\mathcal{G}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(\mathcal{N}_{ij}+r_i-1)!}\prod_{k=1}^{r_i}\mathcal{N}_{ijk}!$ ***Formule-I-10***

Et ou $P(\mathcal{G})$ représente la probabilité a priori affectée à la structure $\mathcal{G}$.

L'équation 1-10 peut être vue comme une mesure de qualité du réseau par rapport aux données et appelée la mesure bayésienne.

En supposant un a priori uniforme sur les structures, la qualité d'un jeux de parents pour un nœud fixé pourra donc être mesurée par le score locale de l'équation 1-11

$s\big(X_i, Pa(X_i)\big) = \prod_{j=1}^{q_i}\frac{(r_i-1)!}{(\mathcal{N}_{ij}+r_i-1)!}\prod_{k=1}^{r_i}\mathcal{N}_{ijk}!$ ***Formule I-11***

Pour $i = 1$ *à* n

$pa_i \leftarrow \phi$

$g_{old} \leftarrow g(i, pa_i)$

$OK \leftarrow vrai$

Répéter

- *Chercher* $X_I \in Pred(X_i)\backslash pa_i$ *qui maximise* $g\big(i, pa_i \cup \{X_j\}\big)$
- $g_{new} \leftarrow g(i, pa_i) \cup \{X_j\}$
- *Si* $g_{new} > g_{old}$ *alors*
  $g_{old} \leftarrow g_{new}$
  $pa_i \leftarrow pa_i \cup \{X_j\}$

Sinon $OK \leftarrow faux$

Tan que OK et $|pa_i| < u$

ALGO 1-3 : algorithme K2[24]

Notion de l'algorithme K2 :

$Pred(\ \ )$	Relation d'ordre sur les nœuds X_i
u	Born sup. du nombre de parents possibles pour un noeud
pa_i	Ensemble des parents du nœud X_i
$g(i, pa_i)$	Score local défini dans l'équation 1-11

Ainsi l'algorithme K2 reprend le score bayésien Dirichlet équation 1-12[24] avec un a priori uniforme sur les structures. Ce score peut s'écrire (équation 1-13) [24].

$$Score\ \mathcal{BD}(\mathcal{B},\mathcal{D}) = P(\mathcal{B}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij}+\alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk}+\alpha_{ijk})}{\Gamma(\alpha_{ijk})} \qquad \textbf{\textit{Formule I-12}}$$

Et

$$Score\ \mathcal{BD}(\mathcal{B},\mathcal{D}) \alpha \prod_{i=1}^{n} g(i, pa_i)$$

Avec

$$g(i, pa_i) = \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij}+\alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk}+\alpha_{ijk})}{\Gamma(\alpha_{ijk})} \qquad \textbf{\textit{Formule I-13}}$$

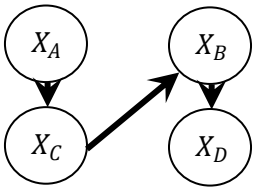
Ou Γ est la fonction *Gamma*.

III-4- Recherche Gloutonne $\mathbb{B}$:

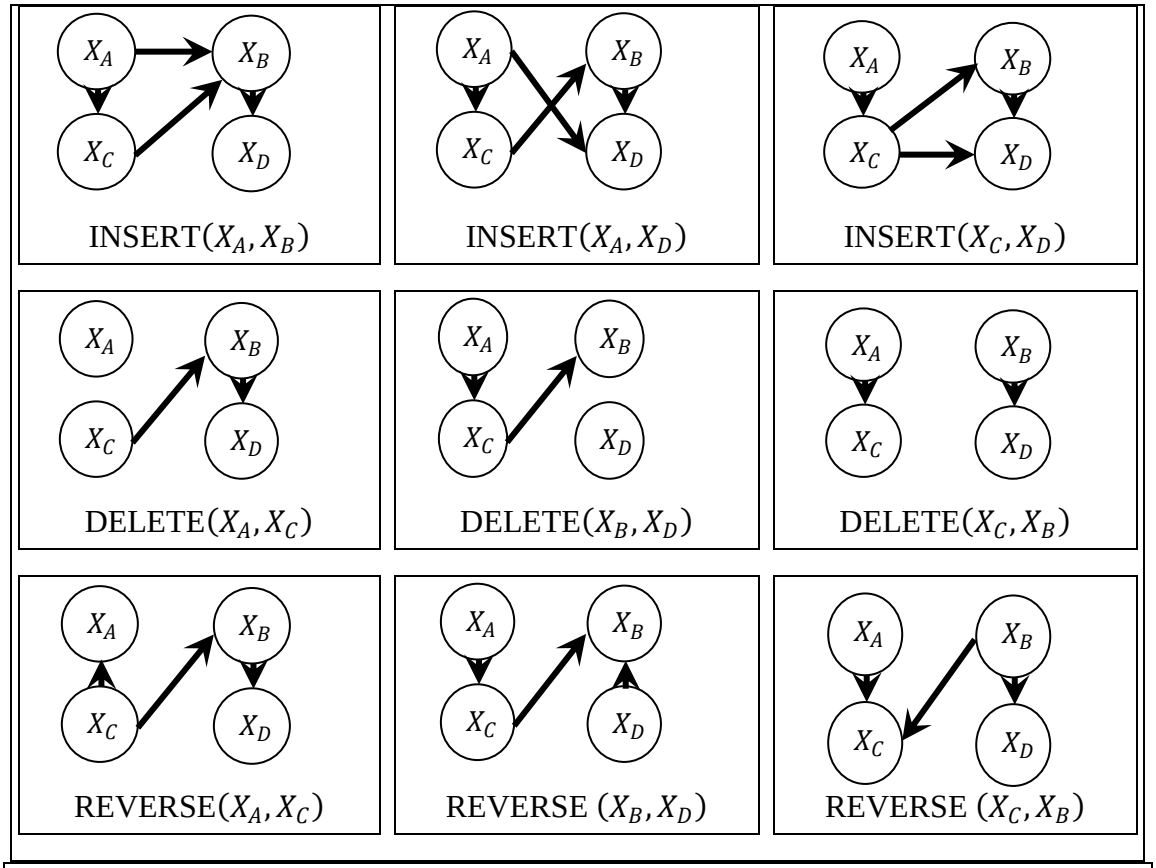
Le principales différences entre les méthodes proposées résident dans la façon de parcourir les opérateurs permettant de générer le voisinage d'un graphe, et l'utilisation d'heuristiques supplémentaires pour simplifier le voisinage obtenu.

Chickering et al. [CGH, 1995] utilisent l'algorithme classique de recherche gloutonne (*Greedy Search*) [24] dans l'espace des réseaux bayésiens décrit dans l'ALGO 1-4. La notion de voisinage utilisée, définie à l'aide de trois opérateurs : ajout, suppression, ou inversion d'arc, est illustrée dans Figure 1-6. L'utilisation d'un score décomposable localement nous permet de calculer rapidement la variation du score pour les structures obtenues avec ces trois opérateurs voir TAB 1-1.

Considérons le graphe $\mathcal{B}$ suivant ainsi qu'un voisinage défini par les trois opérateurs ajout (INSERT), suppression (DELETE), et retournement (REVERSE) d'arc. Remarquons que les graphes résultants ne sont retenue que s'ils sont sans circuit.



Génération du voisinage de $\mathcal{B}$



Notons que pour cet exemple de petite taille, le voisinage comprend déjà neuf DAG dont il va falloir maintenant évaluer la qualité. Pour des structures plus complexes, la taille du voisinage devient beaucoup plus importante, ce qui rend nécessaire l'utilisation de scores locaux pour limiter les calculs et l'implémentation d'un système de cache pour ne pas recalculer plusieurs fois chaque score local.

Figure 1-6 : Exemple de Voisinage GS

Opérateur	INSERT(X_A, X_B)	DELETE (X_A, X_B)	REVERSE (X_A, X_B)
Variance de score	$s\left(X_B, Pa_{X_B}^{+X_A}\right)$ $-s\left(X_B, Pa_{X_B}\right)$	$s\left(X_B, Pa_{X_B}^{-X_A}\right)$ $-s\left(X_B, Pa_{X_B}\right)$	$s\left(X_B, Pa_{X_B}^{-X_A}\right)$ $-s\left(X_B, Pa_{X_B}\right)$ $+s\left(X_A, Pa_{X_A}^{+X_B}\right)$ $-s\left(X_A, Pa_{X_A}\right)$

TAB 1-1 Exemple d'opérateurs dans l'espace des réseaux bayésiens et calcul de la variation du score pour chacun des opérateurs

- *Initialisation du graphe* $\mathcal{B}$
  (graphe vide, aléatoire, donnée par expert ou arbre obtenu par MWST)
- *Continuer* $\leftarrow vrai$
- $Score_{max} \leftarrow score(\mathcal{B})$
- *Répéter*
  - *Génération de* $V_{\mathcal{B}}$*, voisinage de* $\mathcal{B}$*, à l'aide d'opérateurs :*
    -Ajoute d'arc, suppression d'arc, inversion d'arc
    (les graphes ainsi obtenus doivent être sans circuit)
- *Calcul du score pour chaque graphe de* $V_{\mathcal{B}}$
- $\mathcal{B}_{new} \leftarrow argmax_{\mathcal{B},\in V_{\mathcal{B}}}(score(\mathcal{B}^{'}))$
- *Si* $score\ (\mathcal{B}_{new} \geq score_{max})$ *alors*
  $score_{max} \leftarrow score\ (\mathcal{B}_{new})$
  $\mathcal{B} \leftarrow \mathcal{B}_{new}$
  Sinon
  $continuer \leftarrow faux$

Tan que continuer

ALGO 1-4 Algorithme Recherche Gloutonne (GS)

Notion de l'algorithme Recherche Gloutonne (GS)

$score()$	Fonction de score sur les structures possibles
$V_{\mathcal{B}}$	Ensemble des DAG voisins du DAG $\mathcal{B}$ courant
$\mathcal{B}$	Structure finale obtenue par l'algorithme

III-5- L'algorithme EM :

L'algorithme itératif EM[24] aux réseaux bayésiens a été proposée dans [CDLS, 1999] et [NH, 1998] puis adapté aux grandes bases de données dans [TMH, 2001]. Nous allons présenter les grandes lignes de cet algorithme dans le cas de l'apprentissage bayésien, pour des paramètres, il suffit de remplacer le maximum de vraisemblance de l'étape M par un maximum (ou une espérance) a *posteriori*. Nous obtenons dans le cas de l'espérance a *posteriori :*

$$\theta_{i,j,k}^{(t+1)} = \frac{N_{i,j,k}^{*} + \alpha_{i,j,k}}{\sum_k \left(N_{i,j,k}^{*} + + \alpha_{i,j,k}\right)} \qquad \textbf{\textit{Formule 1-14}}$$

Ou $N_{i,j,k}^{*} = E_{\theta^*}\left[N_{i,j,k}\right] = N * P\left(X_i = x_k, Pa(X_i) = pa_j \middle| \theta^*\right)$ est obtenu par inférence dans le réseaux de paramètres θ^* si les $\{X_i, Pa(X_i)\}$ ne sont pas complètement mesurés, et par simple comptage sinon. $\theta_{i,j,k}^{(t+1)}$ les paramètres du réseau bayésiens à l'itération t.

- *Initialiser* $i \longleftarrow 0$
- *Initialisation du graphe* $\mathcal{G}^0$
  (graphe vide, aléatoire, donné par expert ou arbre obtenu par MWST-EM)
- *Initialisation des paramétres* θ^0
- *Répéter*
  - $i \longleftarrow i + 1$
  - $\left(\mathcal{B}^i, \theta^i\right) = argmax_{\mathcal{B},\theta}\ Q\left(\mathcal{B}, \theta : \mathcal{B}^{i-1}, \theta^{i-1}\right)$

Tan que $\left|Q\left(\mathcal{B}^i, \theta^i : \mathcal{B}^{i-1}, \theta^{i-1}\right) - \left(\mathcal{B}^{i-1}, \theta^{i-1} : \mathcal{B}^{i-1}, \theta^{i-1}\right)\right| > \epsilon$

ALGO 1-5 Algorithme EM structure génétique[24]

Notion de l'algorithme EM :

$Q(\mathcal{B}, \theta : \mathcal{B}^*, \theta^*)$	Espérance de la vraisemblance d'un réseau bayésiens $< Q, \theta >$ calculée à partir de la distribution de probabilité des données manquantes $P(\mathcal{D}_m \mid \mathcal{B}^*, \theta^*)$

IV- Apprentissage des paramètres :

IV-1- A partir de données complètes :

IV-1-1- Apprentissage statistique :

Nous cherchons ici à estimer les distributions de probabilités[24] (ou les paramètres des lois correspondantes) à partir de données disponibles. L'estimation de distributions de probabilités, paramétriques ou non, est un sujet très vaste et complexe. Nous décrirons ici les méthodes les plus utilisées dans le cadre des réseaux bayésiens, selon que les données à notre disposition sont complètes ou non.

IV-1-2- Apprentissage statistique :

Dans le cas où toutes les variables sont observées, la méthode la plus simple et la plus utilisée est l'estimation statistique qui consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données. Cette approche, appelée maximum de vraisemblance (MV), nous donne alors :

$$\hat{P}\left(X_i = x_k \middle| pa(X_i) = x_j\right) = \hat{\theta}_{i,j,k}^{MV} = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}} \qquad \textit{Equation.I-15}$$

Ou $N_{i,j,k}$ est le nombre d'evenements dans la base de données pour lesquels la variable X_i est dans l'etat x_k et ses parents sont dans la configuration x_j.

Soit $X^{(l)} = \left\{x_{k_1}^{(l)} \dots x_{k_n}^{(l)}\right\}$ un exemple de notre base de données. La vraisemblance de cet exemple conditionnellement aux paramétres θ du réseau est :

$$P\left(\mathcal{X} = x^{(l)} \middle| \theta\right) = P\left(X_1 = x_{k_1}^{(l)}, \dots, X_n = x_{k_n}^{(l)} \middle| \theta\right)$$

$$= \prod_{i=1}^{n} P\left(X_i = x_{k_i}^{(l)} \middle| pa(X_i) = x_j^{(l)}, \theta\right)$$

$$= \prod_{i=1}^{n} \theta_{i,j(l),k(l)}$$

La vraisemblance de l'ensemble des données $\mathcal{D}$ est :

$$L(\mathcal{D}|\theta) = \prod_{l=1}^{N} P\left(\mathcal{X} = x^{(l)} \middle| \theta\right) = \prod_{i=1}^{n} \prod_{l=1}^{N} \theta_{i,j(l),k(l)}$$

L'examen détaillé du produit $\prod_l \theta_{i,j(l),k(l)}$ nous montre que le terme $\theta_{i,j,k}$ (pour i, j, k fixés) apparaît autant de fois que l'on trouve la configuration $X_i = x_k$ et $pa(X_i) = x_j$ dans les données, soit $N_{i,j,k}$. La vraisemblance des données peut donc se réécrire :

$$L(\mathcal{D}|\theta) = \prod_{i=1}^{n}\prod_{l=1}^{N}\theta_{i,j(l),k(l)} = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{i,j,k}^{N_{i,j,k}} \qquad \textbf{Equation I-16}$$

La log-vraisemblance s'écrit alors :.

$$LL(\mathcal{D}|\theta) = \log L(\mathcal{D}|\theta) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{i,j,k}\ \log\theta_{i,j,k} \qquad \textbf{Equation I-17}$$

Nous savons aussi que les $\theta_{i,j,k}$ sont liés par la formule suivante :

$\sum_{k=1}^{r_i}\theta_{i,j,k} = 1$ soit $\theta_{i,j,r_i} = 1 - \sum_{k=1}^{r_i-1}\theta_{i,j,k}$

Réécrivons la log-vraisemblance à partir des $\theta_{i,j,k}$ indépendants :

$$LL(\mathcal{D}|\theta) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\sum_{k}^{r_i-1} N_{i,j,k}\ log\theta_{i,j,k} + N_{i,j,r_i}\log\left(1 - \sum_{k=1}^{r_i-1}\theta_{i,j,k}\right)\right)$$

Et sa dérivée par rapport à un paramètre $\theta_{i,j,k}$ est :

$$\frac{\partial LL(\mathcal{D}|\theta)}{\partial\theta_{i,j,k}} = \frac{N_{i,j,k}}{\theta_{i,j,k}} - \frac{N_{i,j,r_i}}{\left(1 - \sum_{k=1}^{r_i-1}\theta_{i,j,k}\right)} = \frac{N_{i,j,k}}{\theta_{i,j,k}} - \frac{N_{i,j,r_i}}{\theta_{i,j,r_i}}$$

La valeur $\hat{\theta}_{i,j,k}$ du paramètre $\theta_{i,j,k}$ maximisant la vraisemblance doit annuler cette dérivée et vérifie donc :

$$\frac{N_{i,j,k}}{\hat{\theta}_{i,j,k}} = \frac{N_{i,j,r_i}}{\hat{\theta}_{i,j,r_i}} \qquad \forall\, k \in \{1,\dots,r_i - 1\}$$

Soit

$$\frac{N_{i,j,1}}{\hat{\theta}_{i,j,1}} = \frac{N_{i,j,2}}{\hat{\theta}_{i,j,2}} = \cdots = \frac{N_{i,j,r_i-1}}{\hat{\theta}_{i,j,r_i-1}} = \frac{N_{i,j,r_i}}{\hat{\theta}_{i,j,r_i}} = \frac{\sum_{k=1}^{r_i} N_{i,j,k}}{\sum_{k=1}^{r_i}\hat{\theta}_{i,j,k}} = \sum_{k=1}^{r_i} N_{i,j,k}$$

D'où

$$\hat{\theta}_{i,j,k} = \frac{N_{i,j,k}}{\sum_{k=1}^{r_i} N_{i,j,k}} \qquad \forall k \in \{1,\dots,r_i\}$$

IV-1-3- Apprentissage Bayésien :

L'estimation Bayésien suit un principe quelque peu différent. Il consiste à trouver les paramètres θ les plus probables sachent que les données ont été observées, en utilisant des a priori sur les paramètres. La règle de bayes nous énonce que :

$$P(\theta|\mathcal{D}) \ \propto\ P(\mathcal{D}|\theta)P(\theta) = L(\mathcal{D}|\theta)P(\theta)$$

Lorsque la distribution de l'échantillon suit une loi multinomiale (voire équation 1-16), la distribution a priori conjuguée est la distribution Dirichlet :

$$P(\theta) \propto \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\left(\theta_{i,j,k}\right)^{\propto_{i,j,k}-1}$$

Ou $\propto_{i,j,k}$ sont les coefficients de la distribution de Dirichlet associée à la loi a priori $P\left(X_i = x_k \middle| pa(X_i) = x_j\right)$. Un des avantages des distributions exponentielles celle

de Dirichlet est qu'elle permet d'exprimer facilement la loi a posteriori des paramètres $P(\theta|\mathcal{D})$:

$$P(\theta|\mathcal{D}) \propto \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}+\propto_{i,j,k}-1}$$

En posant $N'_{i,j,k} = N_{i,j,k} + \propto_{i,j,k} - 1$, on retrouve le méme genre de formule que dans l'equation 1-16. Un raisonnement identique permet de trouver les valeurs des parametres $\theta_{i,j,k}$ qui vont maxmiser $P(\theta|\mathcal{D})$.

L'approche de maximum a posteriori (MAP) nous donne alors :

$$\hat{P}(X_i = x_k | Pa(X_i) = x_j) = \hat{\theta}_{i,j,k}^{MAP} = \frac{N_{i,j,k}+\propto_{i,j,k}-1}{\sum_k (N_{i,j,k}+\propto_{i,j,k}-1)} \qquad \textbf{Equation I-18}$$

Où $\propto_{i,j,k}$ sont les paramètres de la distribution de Dirichlet associée à la loi a priori $P(X_i = x_k | pa(X_i) = x_j)$.

Une autre approche Bayésien consiste à calculer l'espérance a posteriori des paramètres $\theta_{i,j,k}$au lieu d'en chercher le maximum. Nous obtenons alors par cette approche d'espérance a posteriori (EAP).

$$\hat{P}(X_i = x_k | pa(X_i) = x_j) = \hat{\theta}_{i,j,k}^{EAP} = \frac{N_{i,j,k}+\propto_{i,j,k}}{\sum_k (N_{i,j,k}+\propto_{i,j,k})} \qquad \textbf{Equation I-19}$$

Les estimations que nous venons d'évoquer (maximum de vraisemblance, maximum a posteriori et espérance a posteriori) ne sont valables que si les variables sont entièrement observées. Les méthodes suivantes vont donc s'appliquer aux cas où certaines données sont manquantes.

IV-2 A partir de données incomplètes :

Dans les applications pratiques, les bases de données sont très souvent incomplètes. Certaines variables ne sont observées que partiellement ou même jamais, que ce soit en raison d'une panne de capteurs, d'une variable mesurable seulement dans un contexte bien précis, d'une personne sondée ayant oublié de répondre à une question, etc...

Après avoir constaté l'existence de différents types de données incomplètes, nous aborderons les deux cas traitables automatiquement, pour ensuite nous concentrer sur un des algorithmes les plus utilisés, l'algorithme EM.

IV-2-1- Nature des données manquantes :

Notons $\mathcal{D} = \{X_i^l\}\ 1 \leq i \leq n, 1 \leq l \leq N$ notre ensemble de données, avec $\mathcal{D}_0$ la partie observée mais incomplète de $\mathcal{D}$, et $\mathcal{D}_m$ la partie manquante. Notons aussi $\mathcal{M} = \{\mathcal{M}_{il}\}$ avec $\mathcal{M}_{il} = 1$ si X_i^l est manquant, et 0 sinon.

Le traitement des données manquantes dépend de leur nature. Distingue plusieurs types de données manquantes :

- MCAR (Missing Completly At Random) : $P(\mathcal{M}|\mathcal{D}) = P(\mathcal{M})$, la probabilité qu'une donnée soit manquante ne dépend pas de $\mathcal{D}$,
- MAR (Missing At Random) : $P(\mathcal{M}|\mathcal{D}) = P(\mathcal{M}|\mathcal{D}_0)$, la probabilité qu'une donnée soit manquante dépend des données observées,
- NMAR (Not Missing At Random) : la probabilité qu'une donnée soit manquante dépend à la fois des données observées et manquantes.

Les situations MCAR et MAR sont les plus faciles à résoudre car les données observées contiennent toutes les informations nécessaires pour estimer la distribution des données manquantes. La situation NMAR est plus délicate car il faut alors faire appel à des informations extérieures pour réussir à modéliser la distribution des données manquantes et revenir à une situation MCAR ou MAR.

IV-2-2- Traitement des données MCAR :

Lorsque les données manquantes sont de type MCAR, la première approche possible et la plus simple est l'analyse des exemples complets. Cette méthode consiste à estimer les paramètres à partir de $\mathcal{D}_{co}$ ensemble des exemples complètement observés dans $\mathcal{D}_o$. Lorsque D est MCAR, l'estimateur basé sur $\mathcal{D}_{co}$ n'est pas biaisé. Malheureusement, lorsque le nombre de variables est élevé, la probabilité qu'un exemple soit complètement mesuré devient faible, et $\mathcal{D}_{co}$ peut être vide ou insuffisant pour que la qualité de l'estimation soit bonne.

Une autre technique, l'analyse des exemples disponibles, est particulièrement intéressante dans le cas des réseaux bayésiens. En effet, puisque la loi jointe est décomposée en un produit de probabilités conditionnelles, il n'est pas nécessaire de mesurer toutes les variables pour estimer la loi de probabilité conditionnelle $P\big(X_i\big|Pa(X_i)\big)$, mais seulement des variables Xi et Pa(Xi). Il suffit donc d'utiliser tous les exemples où X_i et $Pa(X_i)$ sont complètement mesurés pour l'estimation de $P\big(X_i\big|Pa(X_i)\big)$.

IV-2-3- Traitement des données MAR

De nombreuses méthodes tentent d'estimer les paramètres d'un modèle à partir de données MAR. Citons par exemple le sequential updating, l'échantillonnage de Gibbs, et l'algorithme expectation maximisation (EM).

Plus récemment, les algorithmes bound and collapse et robust bayesian estimator cherchent à résoudre le problème quel que soit le type de données manquantes.

L'application de l'algorithme itératif EM aux réseaux bayésiens a été proposée et puis adaptée aux grandes bases de données. Nous allons présenter les grandes lignes de cet algorithme dans le cas de l'apprentissage statistique puis de l'apprentissage bayésien.

V- Problématique :

Les réseaux bayésiens permettent de représenter les dépendances probabilistes entre des variables par le biais d'un graphe acyclique dirigé. Ce type de modèle, basé sur la théorie probabiliste, permet d'élaborer un système d'aide à la décision.

Le nombre de graphes acycliques dirigés (DAG) pour représenter toutes les structures possibles pour les réseaux bayésiens est de taille super-exponentielle. En effet, Robinson (1977) a prouvé qu'il était possible de donner ce nombre grâce à la formule récursive suivante :

$$NS(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} NS(n-i)$$

Ce qui donne :

$NS(1) = 1$

$NS(2) = 3$

$NS(3) = 25$

$NS(5) = 29281$

$NS(10) \simeq 4{,}2 \times 10^{18}$

Comme l'équation est super exponentielle, il est impossible d'effectuer un parcours exhaustif en un temps raisonnable dés que le nombre de nœuds dépasse 7.

Plusieurs méta-heuristiques ont été utilisé dans la littérature pour extraire la structure d'un Réseau bayésien.

Dans notre travail nous proposons comme méthode de recherche génétique, l'Algorithme de CouCou et Algorithme Luciole.

CHAPITRE II
ALGORITHME COUCOU ET LUCIOLE

I- Introduction :

L'optimisation et l'intelligence artificielle sont des domaines de recherche actifs avec une littérature en pleine expansion. Pour la plupart des applications, le temps, l'argent et les ressources sont toujours limités, et leur utilisation optimale devient donc de plus en plus importante. Dans les applications de conception moderne, il faut un changement de paradigme dans la pensée et la conception pour trouver des solutions d'économie d'énergie et de conception plus écologique. Par exemple, il est bien connu que des problèmes d'optimisation combinatoire tels que Le problème du voyageur de commerce (en anglais : Travelling Salesman Problem « TSP ») sont NP-Difficiles [Garey and Johnson, 1979], ce qui signifie qu'il n'y a pas d'algorithmes efficaces en général.

Même sans algorithmes efficaces, nous avons encore à résoudre ces problèmes difficiles dans la pratique. Cela conduit au développement des diverses techniques exotiques et des méthodes spécifiques aux problèmes afin que des connaissances spécifiques aux problèmes (comme les dégradés) puissent être utilisées pour guider des meilleures procédures de recherche. En général, les algorithmes qui utilisent des connaissances liées à des problèmes devraient fonctionner mieux que les méthodes de type boîte noire qui n'utilisent pas du tout la connaissance du problème. Mais l'incorporation de connaissances spécifiques à un problème limite souvent l'utilisation d'une méthode ou d'un algorithme. En outre, il n'est pas facile ou trop informatisé d'intégrer ces connaissances. Parfois, il peut être impossible d'incorporer ces connaissances, soit parce que nous ne pouvons pas avoir un aperçu du problème ou parce que nous ne savons pas comment. Dans ce cas, la seule option est d'utiliser des algorithmes de type boîte noire qui n'assument aucune connaissance du problème d'intérêt.

Dans la plupart des cas, pour les problèmes NP-Difficiles, la seule alternative est d'utiliser des méthodes heuristiques et métaheuristiques par essais et erreurs. Les méthodes heuristiques sont la recherche de solutions par essais et faux, alors que les méthodes métaheuristiques peuvent être considérées comme des méthodes heuristiques de niveau supérieur qui utilisent l'information et la sélection des solutions pour guider le processus de recherche. Les algorithmes d'optimisation sont les outils et les techniques permettant de résoudre des problèmes d'optimisation dans le but de trouver son optimalité, même si cette optimalité n'est pas toujours accessible. Cette recherche de l'optimalité est encore compliquée par le fait que l'incertitude est presque toujours

présente dans les systèmes réels. Par conséquent, nous recherchons non seulement la conception optimale mais aussi la conception robuste dans l'ingénierie et l'industrie. Des solutions optimales, qui ne sont pas assez robustes, ne sont pas pratiques en réalité. Des solutions suboptimales ou de bonnes solutions robustes sont souvent le choix dans de tels cas. Au cours des vingt dernières années, les algorithmes métaheuristiques inspirés de la nature ont gagné une grande popularité dans les domaines de l'optimisation, de l'intelligence artificielle, de l'apprentissage automatique, de l'intelligence computationnelle, de l'exploration des données et des applications d'ingénierie.

I-1- Problèmes d'optimisation:

Les problèmes d'optimisation[43] peuvent être formulés de plusieurs façons. Par exemple, la méthode couramment utilisée des moindres carrés est un cas spécial de formulations à maximum de vraisemblance. De loin, la formulation la plus utilisée est d'écrire un problème d'optimisation non linéaire comme:

$minimum\ f_m(x)\ , (m = 1,2, \dots, M)$ ***Formule II-1***

Sous réserve des contraintes

$h_j(x) = 0\ , (j = 1,2, \dots, J)$ ***Formule II-2***

$g_k(x) \leq 0\ , (k = 1,2, \dots, K)\ )$ ***Formule II-3***

Où f_m, h_j et g_k sont en général des fonctions non linéaires. Ici, le vecteur de conception ou les variables de conception $x = x_1, x_2, \dots, x_d$ Peut être continu, discrèt ou mélangé dans l'espace d-dimensionnel. Les fonctions f_m sont appelées fonctions objectives ou coût, et lorsque $M > 1$, le problème d'optimisation est appelé multi-objectif ou multi critère.

Il vaut la peine de souligner que nous écrivons ici le problème au temps que minimisation, il peut également être écrit comme maximisation en remplaçant simplement $f_m(x)$ par $-f_m(x)$. Lorsque toutes les fonctions sont non linéaires, il s'agit de problèmes contraints non linéaires. Dans certains cas particuliers où f_m, h_j et g_k sont linéaires, le problème devient linéaire, et nous pouvons utiliser les techniques de programmation largement linéaires telles que la méthode simplex. Lorsque certaines variables de conception ne peuvent prendre que des valeurs discrètes (souvent des nombres entiers), alors que d'autres variables sont réelles continues, le problème est de type mixte, souvent difficile à résoudre, en particulier pour des problèmes d'optimisation à grande échelle

En général, l'optimisation multi-objectifs nécessite des techniques multi-objectives pour obtenir les fronts de Pareto d'un problème d'intérêt. Cependant, il est possible de combiner différents objectifs en un seul objectif en utilisant la somme pondérée

$f(x) = \sum_{m=1}^{M} \propto_m f_m \, , \propto_m > 0, \sum_{m=1}^{M} \propto_m = 1$ ***Formule II-4***

Il s'agit d'une approche simple qui nous permet d'utiliser des algorithmes d'optimisation à un seul objectif sans grandes modifications. Cependant, cette approche présente certains inconvénients puisqu'elle ne peut traiter que les fronts convexes de Pareto. En outre, une véritable approche multi-objective de l'optimisation multi-objective peut donner beaucoup plus d'informations et de perspicacité dans le problème.

I-2- Définition d'un Algorithme d'Optimisation :

Avant d'aller plus loin pour discuter des algorithmes nature-inspirés tels que la recherche de coucou et l'algorithme de luciole, nous analysons l'essence d'un algorithme d'optimisation.

Essentiellement, l'optimisation est un processus de recherche des solutions optimales à un problème d'intérêt particulier, et ce processus de recherche peut être réalisé en utilisant des agents multiples qui forment essentiellement un système d'agents évolutifs. Ce système peut évoluer par itérations selon un ensemble de règles ou d'équations mathématiques. Par conséquent, un tel système montrera quelques caractéristiques émergentes conduisant à des états auto-organisés qui correspondent à certains optima du paysage objectif. Une fois que les états auto-organisés sont atteints, nous disons que le système converge. Par conséquent, concevoir un algorithme d'optimisation efficace équivalent à imiter l'évolution d'un système auto-organisé.

II- Recherche et analyse de coucou :

II-1- La recherche coucou :

La recherche de coucou **[43]** (CS) est l'un des derniers algorithmes métaheuristiques inspirés de la nature, développés en 2009 par Xin-She Yang et Suash Deb. CS est basé sur le parasitisme de couvaison de certaines espèces de coucou. De plus, cet algorithme est renforcé par les soi-disant vols de Lévy, plutôt que par de simples randonnées aléatoires isotropes. Des études récentes montrent que CS est potentiellement beaucoup plus efficace que le PSO et les algorithmes génétiques. Les coucous sont des oiseaux fascinants, non seulement à cause de leurs sons magnifiques qu'ils peuvent faire, mais aussi en raison de leur stratégie de reproduction agressive.

Certaines espèces, comme les cousins *Ani* et *Guira*, pondent leurs œufs dans des nids communs, bien qu'ils puissent enlever les autres œufs pour augmenter la probabilité d'éclosion de leurs propres œufs. Un bon nombre d'espèces s'engagent dans le parasitisme obligatoire de la couvée en pondant leurs œufs dans les nids d'autres oiseaux hôtes (souvent d'autres espèces).

Pour simplifier la description de la recherche de coucou standard, nous utilisons maintenant les trois règles idéales suivantes:

- Chaque coucou pond un œuf à la fois et le dépose dans un nid choisi au hasard.
- Les meilleurs nids avec des œufs de haute qualité seront transférés aux générations suivantes.
- Le nombre de nids d'hôtes disponibles est fixe et l'œuf pondu par un coucou est découvert par l'oiseau hôte avec une probabilité $p_a \in (0,1)$. Dans ce cas, l'oiseau hôte peut soit se débarrasser de l'œuf, soit simplement abandonner le nid et construire un nid complètement nouveau.

Comme approximation, cette dernière hypothèse peut être approchée en remplaçant une fraction p_a des n nids d'hôtes par des nouveaux nids (avec des nouvelles solutions aléatoires). Pour un problème de maximisation, la qualité ou l'aptitude d'une solution peut simplement être proportionnelle à la valeur de la fonction objective. D'autres formes de conditionnement physique peuvent être définies d'une manière similaire à la fonction de condition physique dans les algorithmes génétiques. Du point de vue de la mise en œuvre, on peut utiliser les représentations simples suivantes que chaque œuf dans un nid représente une solution, et chaque coucou ne peut poser qu'un œuf (représentant ainsi une solution), l'objectif est d'utiliser les nouvelles solutions potentiellement meilleures (Coucous) pour remplacer une solution pas si bonne dans les nids. Évidemment, cet algorithme peut être étendu au cas plus complexe où chaque nid comporte plusieurs œufs représentant un ensemble de solutions. Ici, nous allons utiliser l'approche la plus simple où chaque nid n'a qu'un œuf. Dans ce cas, il n'y a pas de distinction entre un œuf, un nid ou un coucou, car chaque nid correspond à un œuf qui représente aussi un coucou.

Cet algorithme utilise une combinaison équilibrée d'une marche aléatoire locale et de la marche exploratoire globale, contrôlée par un paramètre de commutationp_a. La marche aléatoire locale peut être

$x_i^{t+1} = x_i^t + \propto s \otimes H(p_a - \epsilon) \otimes (x_j^t - x_k^t)$ ***Formule II-5***

Où x_j^t et x_k^t sont deux solutions différentes choisies aléatoirement par permutation aléatoire, $H\ (u)$ est une fonction de Heaviside, ϵ Est un nombre aléatoire tiré d'une distribution uniforme, et s est la taille du pas. D'autre part, la randonnée aléatoire globale est réalisée en utilisant les vols de Lévy

$x_i^{t+1} = x_i^t + \propto L(s, \lambda)$ ***Formule II-6***

Où

$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}} \quad (s \gg s_0 > 0)$ ***Formule II-7***

Ici $\propto > 0$ est le facteur d'échelle de taille d'échelon, qui doit être lié aux échelles du problème d'intérêt. Dans la plupart des cas, on peut utiliser $\propto = O(\frac{L}{10})$, où L est l'échelle caractéristique du problème d'intérêt, alors que dans certains cas $\propto = O(\frac{L}{100})$, peut être plus efficace et éviter de voler trop loin. De toute évidence, la valeur α de ces deux équations de mise à jour peut être différente, conduisant ainsi à deux paramètres différents $\propto_1$ et $\propto_2$. Ici, nous utilisons $\propto_1 = \propto_2 = \propto$ pour simplifier.

L'équation ci-dessus est essentiellement l'équation stochastique pour une marche aléatoire. En général, une marche aléatoire est une chaîne de Markov dont le prochain état / emplacement dépend uniquement de l'emplacement actuel (le premier terme de l'équation ci-dessus) et de la probabilité de transition (le second terme). Cependant, une fraction substantielle des nouvelles solutions devrait être générée par une randomisation sur le terrain lointain et leurs emplacements devraient être assez loin de la meilleure solution actuelle; Ceci assurera que le système ne sera pas piégé dans un optimum local.

La littérature sur la recherche du coucou est en pleine expansion. Il a reçu beaucoup d'attention et il y a beaucoup d'études récentes utilisant la recherche de coucou avec une gamme diverse d'applications. Par exemple, Walton et coll. A amélioré l'algorithme en formulant un algorithme de recherche de coucou modifié, tandis que Yang et Deb l'ont étendu à l'optimisation multi objective.

II-2- Cas particuliers de la Recherche coucou :

La recherche du coucou comme algorithme méta-heuristique présente des caractéristiques étonnamment riches[43]. Si l'on regarde les Eqs. *Formule II-6* et *Formule II-7* plus proches, nous pouvons découvrir une richesse si subtile. A partir de *Formule II-7*, on peut regrouper certains facteurs en posant $Q = \propto s \otimes H(p_a - \epsilon)$, Alors on a $Q > 0$. En conséquence, l'équation *Formule II-8* devient l'équation de mise à jour majeure de l'évolution différentielle (DE). En outre, nous remplaçons encore x_j^t par le meilleur courant g^* et on pose $k = i$, on a

$$x_i^{t+1} = x_i^t + Q(g^* - x_i^t) \qquad \textbf{\textit{Formule II-8}}$$

Qui est une variante essentielle de l'optimisation de l'essaim de particules sans meilleur historique individuel. Dans ce cas, le cas ci-dessus est très semblable à l'optimisation de l'accélération de la fragmentation des particules (APSO) développée par Yang et al.

D'autre part, à partir de *Formule II-8*, cette marche aléatoire est en fait le recuit simulé (SA) avec une probabilité de transition Lévy-vol. Dans ce cas, nous avons un recuit simulé avec une programmation de refroidissement stochastique contrôlée par p_a. Par conséquent, l'évolution différentielle, l'optimisation des essaims de particules et le recuit simulé peuvent être considérés comme des cas particuliers de recherche de coucou. Inversement, on peut dire que la recherche de coucou est une bonne et efficace combinaison de DE, PSO et SA dans un algorithme. Par conséquent, il n'est pas surprenant que la recherche de coucou est très efficace.

II-3- Pourquoi la recherche coucou est si efficace?

En plus de l'analyse précédente montrant que DE, PSO et SA sont particulièrement des cas de recherche de coucou, des études théoriques récentes indiquent aussi que la recherche de coucou a une convergence globale.

Des études théoriques sur l'optimisation des essaims de particules ont suggéré que la PSO peut converger rapidement vers la meilleure solution actuelle, mais pas nécessairement les meilleures solutions globales. En fait, certains analystes suggèrent que les équations de mise à jour des PSO ne satisferont pas aux conditions de convergence globale, et donc il n'y a aucune garantie pour la convergence globale. D'autre part, il a prouvé que la recherche du coucou satisfait aux exigences de convergence globale et à ainsi garantit des propriétés. Cela implique que pour l'optimisation multimodale, la PSO peut converger prématurément vers un optimum

local, alors que la recherche de coucou peut généralement converger vers l'optimalité globale. En outre, la recherche de coucou a deux capacités de recherche: recherche locale et recherche globale, contrôlée par une probabilité de commutation / découverte. la recherche locale est très intensive avec environ 1/4 du temps de recherche (pour $p_a = 0.25$), tandis que la recherche globale prend environ 3/4 du temps de recherche total. Cela permet d'explorer l'espace de recherche de manière plus efficace à l'échelle globale et, par conséquent, l'optimalité globale peut être trouvée avec une probabilité plus élevée. Un autre avantage de la recherche de coucou est que sa recherche globale utilise des vols Lévy ou processus, plutôt que des randonnées aléatoires standard. Comme les vols de Lévy ont une moyenne et une variance infinies, CS peut explorer l'espace de recherche plus efficacement que les algorithmes utilisant des processus gaussiens standard. Cet avantage, conjugué à la fois aux capacités locales et de recherche et à la convergence globale garantie, rend la recherche de coucou très efficace. En effet, diverses études et applications ont démontré que la recherche du coucou est très efficace.

III- Applications de la recherche coucou :

Évidemment, l'optimisation de l'ingénierie fait partie des diverses applications. En fait, la recherche de coucou et ses variantes ont été appliquées dans presque tous les domaines des sciences, de l'ingénierie et de l'industrie. Certaines des études d'application sont résumées dans le tableau II-1.

Application	Auteur	Référence
Seuil d'image multi niveau	Brajevic et al.	[1]
Prévision des inondations	Chaowanawatee and Heednacram	[2]
Réseaux de capteurs sans fil	Dhivya and Sundarambal	[6]
La fusion des données	Dhivya et al.	[7]
Cluster dans les réseaux sans fil	Dhivya et al.	[8]
Clustering	Goel et al.	[11]
Expédition des eaux souterraines	Gupta et al.	[12]
Choix du fournisseur	Kanagaraj et al.	[14]
Prévision de charge	Kavousi-Fard and Kavousi-Fard	[15]
Rugosité de surface	Madic et al.	[16]
Planification de magasin de flux	Marichelvam	[17]
Remplacement optimal	Mellal et al.	[18]
Allocation DG dans le réseau	Moravej and Akhlaghi	[19]
Optimisation du filtre Floraison	Natarajan et al.	[20–21]
Réseau neuronal BPNN	Nawi et al.	[22]
Problème Voyageur de commerce	Ouaarab et al.	[23]
Composition du service Web	Pop et al.	[27]

Composition du service Web	Chifu et al.	[3, 4]
Correspondance ontologique	Ritze and Paulheim	[30]
Reconnaissance des locuteurs	Sood and Kaur	[32]
Tests automatisés de logiciels	Srivastava et al.	[35–36]
Optimisation de fabrication	Syberfeldt and Lidberg	[37]
Reconnaissance de visage	Tiwari	[38]
Formation des modèles neuronaux	Vázquez	[39]
Expédition économique non convexe	Vo et al.	[40]
Planification des trajets UCAV	Wang et al.	[41, 42]
Optimisation des activités	Yang et al.	[44]
Sélection des paramètres d'usinage	Yildiz	[45]
Planification des travaux en grille	Prakash et al.	[28]
Affectation quadratique	Dejam et al.	[5]
Problème d'emboîtement de feuilles	Elkeran	[9]
Optimisation des requêtes	Joshi and Srivastava	[13]
N-Queens puzzle	Sharma and Keswani	[31]
Jeux d'ordinateur	Speed	[33, 34]

TAB II-1 : Applications de la recherche coucou[43]

III-1- Analyse théorique et mise en œuvre :

Comme nous l'avons vu, les applications de la recherche de coucou sont très diverses. En revanche, les études théoriques sont très limitées. Ce résumé peut mettre en évidence la nécessité de poursuivre les recherches sur les aspects théoriques de la recherche du coucou.

III-2- Améliorations et autres études :

Comme mentionné précédemment, il n'est pas toujours clair comment classer certains papiers. De nombreuses études portent sur les améliorations de l'algorithme de recherche coucou standard. Donc, nous mettons quelques papiers ici et les résumons ainsi:

- ✓ Entropie de Tsallis.
- ✓ Amélioration de la recherche de diffusion en utilisant la recherche de coucou.
- ✓ Evolution différentielle améliorée via l'opérateur de recherche de coucou.
- ✓ Recherche du coucou par la méthode du gradient conjugué.
- ✓ Recherche de coucou avec PSO.

III-3- Implémentations :

Quels que soient les algorithmes, les implémentations appropriées sont très importantes. Yang fournir une démo standard de la mise en œuvre de la recherche de coucou[1]. Des mises en œuvre importantes telles que l'approche orientée objet et la parallélisassions ont été réalisées, qui peuvent être résumées comme suit:

- ✓ Mise en œuvre orientée objet de CS.
- ✓ Parallélisassions de CS.

IV- Algorithme et analyse de Luciole (Firefly Algorithm) :

IV-1- Algorithme de Luciole :

Algorithme luciole[43] « Firefly Algorithm » (FA) a été développé pour la première fois par Xin-She Yang à la fin de 2007 et 2008 à l'université de Cambridge, qui était basée sur les modèles clignotants et le comportement des lucioles. La littérature FA a considérablement augmenté au cours des 5 dernières années avec plusieurs centaines de documents publiés sur les algorithmes Luciole, Foster et al. A fourni un examen approfondi. Essentiellement, la FA utilise les trois règles idéales suivantes:

- Les lucioles sont unisexes de sorte qu'une luciole sera attirée par d'autres lucioles indépendamment de leur sexe.
- L'attractivité est proportionnelle à la luminosité, et elles diminuent à mesure que leur distance augmente. Ainsi, pour deux lucioles clignotantes, celle qui est moins brillante va se rapprocher de la plus brillante. S'il n'y en a pas plus brillant qu'une luciole particulière, il se déplace aléatoirement.
- La luminosité d'une luciole est déterminée par le paysage de la fonction objective.

Comme l'attrait d'une luciole est proportionnel à l'intensité lumineuse observée par les lucioles adjacentes, on peut maintenant définir la variation de l'attractivité β avec la distance r par

$$\beta = \beta_0 e^{-\gamma r^2} \qquad \textbf{\textit{Formule II-9}}$$

Où β_0 est l'attractivité à $r = 0$.

Le Mouvement d'une luciole *i* est attiré par une autre luciole plus attrayante (plus lumineuse) *j* est déterminé par

1-http://www.mathworks.co.uk/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm (18/03/2017).

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r^2 ij}\left(x_j^t - x_i^t\right) + \propto_t \epsilon_i^t \qquad \textbf{Formule II-10}$$

Où le second terme est dû à l'attraction. Le troisième terme est la randomisation avec $\propto_t$ étant le paramètre de randomisation, et ϵ_i^t est un vecteur de nombres aléatoires tiré d'une distribution gaussienne ou d'une distribution uniforme au temps *t*. Si $\beta_0 = 0$, il devient une marche aléatoire simple. D'autre part, si $\gamma = 0$, FA se réduit à une variante de l'optimisation des essaims de particules. En outre, la randomisation ϵ_i^t peut facilement être étendue à d'autres distributions telles que les vols de Lévy. Une version de démonstration de la mise en œuvre de l'algorithme de luciole par Xin-She Yang, sans les vols de Lévy pour plus de simplicité, peut être trouvée au site d'échange de fichiers Mathworks[1].

IV-2- Les Paramètres :

Comme $\propto_t$ contrôle essentiellement l'aléatoire (ou, dans une certaine mesure, la diversité des solutions), on peut régler ce paramètre[43] pendant les itérations de sorte qu'il puisse varier avec le compteur d'itérations *t*. Donc, une bonne façon d'exprimer $\propto_t$ est d'utiliser

$$\propto_t = \propto_0 \delta^t, 0 < \delta < 1 \qquad \textbf{Formule II-11}$$

Où $\propto_0$ est le facteur de mise à l'échelle du caractère aléatoire initial, et δ est essentiellement un facteur de refroidissement. Pour la plupart des applications, on peut utiliser δ = 0,95 à 0,97.

En ce qui concerne $\propto_0$ initial, les simulations montrent que la FA sera plus efficace si $\propto_0$ est associée à la mise à l'échelle des variables de conception. Soit L l'échelle moyenne du problème d'intérêt, on peut définir $\propto_0 = 0{,}01L$ initialement. Le facteur 0.01 vient du fait que les randonnées aléatoires nécessitent un certain nombre d'étapes pour atteindre la cible tout en équilibrant l'exploitation locale sans trop sauter en quelques pas. Le paramètre β_0 contrôle l'attractivité, et les études paramétriques suggèrent que $\beta_0 = 1$ peut être utilisé pour la plupart des applications. Cependant, γ devrait également être lié à la mise à l'échelle L. En général, on peut définir $\gamma = \frac{1}{\sqrt{L}}$. Si les variations d'échelle ne sont pas significatives, alors on peut fixer $\gamma = O(1)$.

Pour la plupart des applications, nous pouvons utiliser la taille de population $n = 15$ à 100, bien que la meilleure gamme soit $n = 25$ à 40.

1 - http://www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm (18/03/2017).

IV-3- Complexité d'Algorithme :

Presque tous les algorithmes métaheuristiques sont simples en termes de complexité, et ils sont donc faciles à mettre en œuvre. Algorithme Firefly a deux boucles intérieures en passant par la population$n$, et une boucle externe pour l'itérateur t. La complexité dans le cas extrême est donc $O(n^2 t)$. Comme n est petit (typiquement $n = 40$) et t est grand (par exemple, $t = 5000$), le coût de calcul est relativement peu coûteux parce que la complexité de l'algorithme est linéaire en termes de t. Le coût de calcul principal sera dans les évaluations de fonctions objectives, en particulier pour les objectifs de type boîte noire externe. Ce dernier cas est également vrai pour tous les algorithmes métaheuristiques. Après tout, les problèmes d'optimisation, la partie la plus coûteuse en termes de calcul est l'évaluation objective. Si n est relativement grand, il est possible d'utiliser une boucle interne en classant l'attractivité ou la luminosité de toutes les lucioles en utilisant des algorithmes de tri. Dans ce cas, la complexité algorithmique de l'algorithme de luciole sera $O(nt\log(n))$.

IV-4- Cas particuliers de FA :

Algorithme luciole est en effet riche de nombreuses[43] façons. Premièrement, il utilise l'attraction pour influencer le comportement de la population. Comme l'attraction locale tend à être plus forte que l'attraction à longue distance, la population de FA peut automatiquement se subdiviser en sous-groupes, selon la nature du problème, ce qui permet à FA de traiter naturellement des problèmes d'optimisation non linéaires multimodaux.

De plus, nous examinons de plus près l'équation de mise à jour *Formule II-10*, cette équation non linéaire fournit des caractéristiques beaucoup plus riches. Tout d'abord, si γ est très grand, alors l'attractivité ou l'intensité lumineuse diminue trop rapidement, ce qui signifie que le second terme dans *Formule II-10* devient négligeable, conduisant au recuit simulé standard (SA). Deuxièmement, si γ est très petit (c'est-à-dire, $\gamma \to 0$), alors le facteur exponentiel $\exp[-\gamma r_{ij}^2] \to 1$. Nous avons

$$x_i^{t+1} = x_i^t + \beta_0\left(x_j^t - x_i^t\right) + \propto_t \epsilon_i^t \qquad \textbf{\textit{Formule II-12}}$$

Ici, si l'on pose encore $\propto_t = 0$, alors l'équation ci-dessus *Formule II-12* devient une variante de l'évolution différentielle. D'autre part, si nous remplaçons x_i^t par la meilleure solution global actuelle g^*, nous avons

$$x_i^{t+1} = x_i^t + \beta_0(g^* - x_i^t) + \propto_t \epsilon_i^t \qquad \textbf{\textit{Formule II-13}}$$

Qui est essentiellement l'accélération de l'optimisation des essaims de particules (APSO).

En troisième lieu, on positionne $\beta_0 = 0$, et on laisse ϵ_i^t se rapporter à x_i, alors *Formule II-13* devient une variante d'ajustement de hauteur de la recherche d'harmonie (HS).

Par conséquent, on peut dire essentiellement que DE, APSO, SA et HS sont des cas particuliers d'algorithme de luciole. Inversement, la FA peut être considérée comme une bonne combinaison de tous les quatre algorithmes (DE, APSO, SA et HS), dans une certaine mesure. De plus, FA utilise une équation de mise à jour non linéaire, qui peut produire un comportement riche et une convergence plus élevée que les équations de mise à jour linéaire utilisées dans le PSO et le DE standard. Par conséquent, il n'est pas surprenant que la FA puisse surpasser d'autres algorithmes dans de nombreuses applications telles que l'optimisation multimodale, les classifications, le traitement d'image et la sélection de caractéristiques comme nous le verrons plus loin dans les applications.

IV-5- Variantes d'Algorithme Luciole « Firefly Algorithm » :

Pour les problèmes discrets et l'optimisation combinatoire, des versions discrètes de l'algorithme Luciole « FA » ont été développées avec des performances supérieures, qui peuvent être utilisées pour les problèmes de voyageur, la coloration des graphiques et d'autres applications.

De plus, l'extension de l'algorithme de la luciole à l'optimisation multi-objectifs a également été étudiée.

Quelques études montrent que le choix peut améliorer la performance de l'algorithme de luciole, tandis que d'autres études ont tenté d'hybrider FA avec d'autres algorithmes pour améliorer leur performance.

IV-6- Attraction et diffusion :

La nouvelle idée d'attraction[43] par l'intensité lumineuse comme mécanisme d'exploitation a été utilisée pour la première fois par Yang dans l'algorithme de luciole (FA) en 2007 et 2008. Dans FA, l'attractivité (et l'intensité lumineuse) est intrinsèquement liée à la loi inverse-carrée des variations d'intensité lumineuse et du coefficient d'absorption. Il en résulte un nouveau terme non linéaire de $\beta_0 \exp[-\gamma r^2]$ où β_0 est l'attractivité à la distance $r = 0$ et $\gamma > 0$ le coefficient d'absorption de la lumière.

D'autres algorithmes ont également utilisé des lois inverses carrées, dérivées de la nature. Par exemple, la recherche par système chargé (CSS) a utilisé la loi de Coulomb, tandis que l'algorithme de recherche gravitationnelle (GSA) a utilisé la loi de gravitation de Newton.

La principale fonction de cette attraction est de permettre à un algorithme de converger rapidement parce que ces systèmes multi-agents évoluent, interagissent et s'attirent, conduisant à un certain comportement et attracteurs. Au fur et à mesure que les agents d'essaimage évoluent, il est possible que leurs états d'attracteur évoluent vers l'optimalité globale réelle.

Ce nouveau mécanisme d'attraction est le premier de ce type dans la littérature du calcul inspiré de la nature et de l'intelligence computationnelle. Cela a également motivé et inspiré d'autres à concevoir des mécanismes similaires ou d'autres types d'attraction. Quel que soit le mécanisme d'attraction, du point de vue métaheuristiques, les principes fondamentaux sont les mêmes: C'est-à-dire qu'ils permettent aux agents d'essaimage d'interagir les uns avec les autres et fournissent un terme forçant pour guider la convergence de la population. Attraction Fournit principalement les mécanismes seulement pour l'exploitation, mais, avec la randomisation appropriée, il est également possible d'effectuer un certain degré d'exploration. Cependant, l'exploration est mieux analysée dans le cadre des randonnées aléatoires et de la randomisation par diffusion. Du point de vue de la chaîne de Markov, les randonnées aléatoires et la diffusion sont toutes les deux des chaînes de Markov. En fait, la diffusion brownienne telle que la dispersion d'une goutte d'encre dans l'eau est une marche aléatoire. Par exemple, les randonnées aléatoires les plus fondamentales pour un agent ou une solution x_i peuvent s'écrire sous la forme suivante:

$$x_i^{(t+1)} = x_i^{(t)} + \epsilon \qquad \textbf{\textit{Formule II-14}}$$

Où t est un compteur d'étapes. Ici, ϵ est un nombre aléatoire tiré d'une distribution normale de Gauss avec une moyenne nulle. Ceci donne une distance de diffusion moyenne d'une particule ou d'un agent qui est une racine carrée du nombre fini d'étapes t. C'est-à-dire que la distance est l'ordre de $\sqrt{Dt}$ où D est le coefficient de diffusion. Pour être plus précis, la variance des marches aléatoires dans un cas d-dimensionnel peut être écrite comme

$$\sigma^2(t) = |v_0|^2 t^2 + (2dD)t \qquad \textbf{\textit{Formule II-15}}$$

Où v_0 est la vitesse de dérive.

Cela signifie qu'il est possible de couvrir l'ensemble du domaine de recherche si t est suffisamment grand. Par conséquent, les étapes du mouvement brownien $B(t)$ obéissent essentiellement à une distribution gaussienne avec une moyenne nulle et une variance dépendante du temps. Un processus de diffusion peut être considéré comme une série de mouvement brownien, qui obéit à une distribution gaussienne. Pour cette raison, la diffusion standard est souvent appelée la diffusion gaussienne. Si le mouvement à chaque pas n'est pas gaussien, alors la diffusion est appelée diffusion non gaussienne. D'autre part, randonnées aléatoires peuvent prendre de nombreuses formes. Si les longueurs des pas obéissent à d'autres distributions, nous devons faire face à des randonnées aléatoires plus généralisées. Un cas très particulier est lorsque les longueurs d'échelon obéissent à la distribution de Lévy, une telle randonnée aléatoire est appelée Lévy ou Lévy promenades.

IV-7- Pourquoi SA est efficace :

Comme la littérature sur l'algorithme de luciole se développe et de nouvelles variantes ont émergé, tous souligné l'algorithme de luciole peut surpasser de nombreux autres algorithmes[43]. Maintenant, nous pouvons demander naturellement «Pourquoi est-il si efficace?» Pour répondre à cette question, analysons brièvement l'algorithme luciole « FA » lui-même. FA est basé sur le renseignement d'essaim, ainsi il a les avantages semblables que d'autres intelligences d'essaim basés sur des algorithmes. En fait, une simple analyse des paramètres suggère que certains variants PSO tels que Accélérer PSO sont un cas particulier de l'algorithme luciole lorsque $\gamma = 0$.

Cependant, la FA a deux avantages majeurs par rapport aux autres algorithmes: la subdivision automatique et la capacité de traiter la multi-modalité. Tout d'abord, FA est basé sur l'attraction et l'attractivité diminue avec la distance. Cela conduit au fait que toute la population peut automatiquement subdiviser en sous-groupes, et chaque groupe peut essaim autour de chaque mode ou optimum local. Mathématiquement, $1/\sqrt{\gamma}$ contrôle la distance moyenne d'un groupe de lucioles qui peuvent être vus par les groupes adjacents. Par conséquent, toute une population peut se subdiviser en sous-groupes avec une distance moyenne donnée. Dans le cas extrême où $\gamma = 0$, toute la population ne subdivise pas. Cette capacité de subdivision automatique le rend particulièrement adapté aux problèmes d'optimisation multimodaux très non linéaires.

En outre, les paramètres de FA peuvent être réglés pour contrôler le caractère aléatoire à mesure que les itérations se poursuivent, de sorte que la convergence peut aussi être accélérée en accordant ces paramètres. Ces avantages ci-dessus le rendent flexible pour traiter les problèmes continus, le regroupement et les classifications, et l'optimisation combinatoire aussi bien. Par exemple, utilisons deux fonctions pour démontrer le coût de calcul économisé par FA. Pour la fonction de De Jong avec $d = 256$ dimensions

$f(x) = \sum_{i=1}^{256} x_i^2$ ***Formule II-16***

Les algorithmes génétiques ont nécessité 25412 ± 1237 évaluations pour obtenir une précision de 10^{-5} de la solution optimale, alors que la PSO a nécessité 17040 ± 1123 évaluations. Pour FA, nous avons obtenu la même précision de 5657 ± 730. Cela permet d'économiser environ 78 et 67% du coût de calcul, par rapport à GA et PSO, respectivement.

Pour la fonction forestière de Yang

$f(x) = \left(\sum_{i=1}^{d} |x_i|\right) exp\left[-\sum_{i=1}^{d} \sin(x_i^2)\right], -2\pi \leq x_i \leq +2\pi$ ***Formule II-17***

GA a exigé 37079 ± 8920 avec un taux de réussite de 88% pour $d = 16$ et PSO a nécessité 19725 ± 3204 avec un taux de réussite de 98%. FA a obtenu un taux de réussite de 100% avec seulement 5152 ± 2493. Comparé à GA et PSO, FA économisé environ 86 et 74%, respectivement, de l'ensemble des efforts de calcul.

En bref, FA a trois avantages distincts:

- Subdivision autonome de la population entière en sous-groupes.
- La capacité naturelle de faire face à l'optimisation multimodale.
- Haute périodicité et diversité dans les solutions.

Tous ces avantages rendent la FA très unique et très efficace.

V- Applications :

L'algorithme luciole « FA » a attiré beaucoup d'attention et a été appliqué[43] à des nombreuses applications. Horng et al. A démontré que l'algorithme basé sur luciole utilisait le moins de temps de calcul pour la compression d'image numérique, tandis que Banati et Bajaj utilisaient l'algorithme de luciole pour la sélection des caractéristiques et

montraient que l'algorithme luciole « FA » produisait des performances cohérentes et meilleures en termes de temps et d'optimalité que d'autres algorithmes.

Dans les problèmes de conception technique, Gandomi et al. et Azad et Azad ont confirmé que l'algorithme luciole peut résoudre efficacement les problèmes de conception multimodaux hautement non linéaires. Basu et Mahanti ainsi que Chatterjee et al. Ont appliqué FA dans l'optimisation de la conception de l'antenne et ont montré que la FA peut surpasser l'algorithme de la colonie artificielle d'abeilles (ABC). En outre, Zaman et Martin ont également constaté que FAR peut surperformer PSO et obtenu les meilleurs résultats globaux. Sayadi et al. A développé une version discrète de FA qui peut résoudre efficacement les problèmes d'ordonnancement NP-Difficile, tandis qu'une analyse détaillée a démontré l'efficacité de la FA sur une large gamme de problèmes d'essai, y compris les problèmes multi objectifs d'expédition de charge. En outre, FA peut également résoudre problème de planification et de voyage d'une manière prometteuse.

La classification et le regroupement sont un autre domaine important d'applications de bonnes performances. Par exemple, Senthilnath et al. A fourni une étude de performance approfondie par FA comparé avec 11 algorithmes différents et a conclu que l'algorithme de luciole peut être efficacement utilisé pour le regroupement. Dans la plupart des cas, l'algorithme luciole « FA » peut surpasser tous les autres 11 algorithmes. En outre, l'algorithme luciole « FA » a également été appliqué à la formation des réseaux neuronaux.

Pour l'optimisation dans des environnements dynamiques, la FA peut également être très efficace comme le montre Farahani et al. et Abshouri et al.

VI- Conclusions :

Les algorithmes basés sur l'intelligence d'essaim comme la recherche de coucou et l'algorithme de luciole sont très efficaces pour résoudre un large éventail de problèmes d'optimisation non linéaire et ont donc diverses applications en sciences et en ingénierie. Certains algorithmes (par exemple, la recherche de coucou) peuvent avoir une très bonne convergence globale. Cependant, il reste encore des questions difficiles à résoudre dans les études futures. L'un des principaux problèmes est qu'il existe un écart important entre la théorie et la pratique. La plupart des algorithmes métaheuristiques ont de bonnes applications dans la pratique, mais l'analyse mathématique de ces algorithmes manque loin derrière. En fait, en dehors de quelques résultats limités sur la convergence

et la stabilité sur les algorithmes tels que l'essaim de particules, les algorithmes génétiques et la recherche de coucou, de nombreux algorithmes n'ont pas d'analyse théorique. Par conséquent, nous pouvons savoir qu'ils peuvent bien fonctionner dans la pratique, nous comprenons difficilement pourquoi il fonctionne et comment les améliorer avec une bonne compréhension de leurs mécanismes de travail.

Une autre question importante est que tous les algorithmes métaheuristiques ont des paramètres dépendant de l'algorithme, et les valeurs réelles / le paramétrage de ces paramètres influenceront largement la performance d'un algorithme. Par conséquent, le bon paramétrage lui-même devient un problème d'optimisation. En fait, le paramétrage est un domaine important de la recherche, qui mérite plus d'attention à la recherche. En outre, même avec de très bonnes applications de nombreux algorithmes, la plupart de ces applications concernent les cas où le nombre de variables de conception est inférieur à quelques centaines. Il serait plus avantageux pour les applications réelles si le nombre de variables peut augmenter à plusieurs milliers ou même à l'ordre de millions.

Il convient de souligner que les nouveaux efforts de recherche devraient se concentrer sur les questions importantes telles que celles décrites ci-dessus, il ne devrait pas se concentrer sur le développement de divers nouveaux algorithmes pour les distractions. La nature a évolué sur des milliards d'années, fournissant une source d'inspiration vaste; Cela ne signifie pas que nous devrions développer tous les nouveaux types d'algorithmes, comme l'algorithme de l'herbe, l'algorithme du tigre, l'algorithme de l'arbre, l'algorithme du ciel, l'algorithme du vent, l'algorithme de l'océan ou l'algorithme de l'univers. Ceux-ci pourraient conduire à l'incompréhension et à la confusion de la vraie nature des algorithmes métaheuristiques et de l'optimisation. En fait, les études devraient essayer de répondre à des questions vraiment importantes, y compris l'équilibre optimal de l'exploration et de l'exploitation, la convergence globale, le contrôle optimal des paramètres et l'accord, et les applications réelles à grande échelle. Toutes ces questions difficiles peuvent motiver davantage de recherches. Il ne fait aucun doute que d'autres applications de la recherche de coucou et de l'algorithme de luciole seront visibles dans la littérature en pleine expansion dans un proche avenir.

Il est fortement attendu que des résultats plus théoriques sur l'optimisation métaheuristiques apparaîtront dans les prochaines années.

CHAPITRE III

Conception et implémentation

1- Introduction

Ce chapitre décrit notre méthode de recherche COUCOU et LUCIOLE et. Nous commençons par décrire le principe de ces méthodes et ses détails et nous terminons présentation graphique de notre programme.

1-1- Algorithme de recherche coucou (AC)

Sur la base de la description présentée dans l'algorithme recherche coucou (CS) est une nouvelle méta-heuristique proposée par Xin-She Yang. Cet algorithme a été inspiré par le parasitisme obligé des espèces de coucou en pondant leurs œufs dans les nids des autres oiseaux hôtes. Un certain nid hôte peut s'engager dans un conflit direct. Si un oiseau hôte découvre que les œufs ne sont pas les leurs, ils élimineront ces œufs étrangers ou simplement abandonnent leur nid et construisent un nouveau nid ailleurs. Certaines autres espèces ont évolué de telle sorte que les coucou parasites féminins sont souvent très spécialisés dans le mimétisme de la couleur et le schéma des œufs de quelques espèces hôtes choisies. Cela réduit la probabilité que leurs œufs soient abandonnés et augmente ainsi leur reproductivité. D'autre part, plusieurs études ont montré que le comportement de vol de nombreux animaux et insectes présente les caractéristiques typiques des vols Lévy. Compte tenu de ces comportements de reproduction et de vol, les auteurs de [49] ont proposé l'algorithme CS. L'algorithme CS suit les trois règles idéales suivantes: (1) Chaque coucou pose un œuf à la fois et déverse son œuf dans un nid choisi au hasard; (2) Les meilleurs nids avec une grande qualité d'œufs se transmettront aux prochaines générations; (3) Le nombre de nids d'hôtes disponibles est fixé et l'œuf posé par un coucou est découvert par l'oiseau hôte avec une probabilité $p_a \in [0, 1]$. Dans ce cas, l'oiseau hôte peut soit jeter l'œuf ou abandonner le nid, et construire un nid complètement nouveau.

Sur la base de ces trois règles, les étapes de base de Cuckoo Search (CS) peuvent être résumées comme le pseudo-code suivant:

1 :	*initialiser la population de N nid hôte x_i $\forall i, i = 1, \dots, n$*
2 :	***Tan que*** *$t < MaxGénération$ or $(critère\ d'arrêt)$* ***faire***
3 :	*Obtenez un coucou au hasard par les vols de lévy et évaluez sa forme F_i.*
4 :	*Choisissez au hasard un nid j parmi N.*
5 :	***si*** *$F_i > F_j$* ***alor***
6 :	*Remplacez j par la nouvelle solution*
7 :	***Fin si***
8 :	*Une fraction (p_a) de pire nid sont abandonnés et de nouveaux sont construits.*
9 :	*Conservez les meilleures solutions (ou nid avec des solutions de qualité).*
10 :	*Ranger les solutions et trouver le meilleur en cours.*
11 :	***Fin tan que***

Algorithme 3-1 : RECHERCHE COUCOU[43]

1-2- Algorithme original luciole :

L'intensité lumineuse I de la luciole clignotante diminue lorsque la distance de la source r diminue en fonction de $I \propto 1 / r^2$. De plus, l'absorption de l'air rend la lumière plus faible lorsque la distance de la source augmente. Cette lumière clignotante a représenté l'inspiration pour développer le FA algorithme par Yang en 2008. Ici, l'intensité lumineuse est proportionnelle à la fonction objective du problème. Ici, l'intensité de la lumière est proportionnelle à la fonction objective du problème à optimiser (c'est-à-dire $I(s) \propto f(s)$, où $s = S(x)$ représentent une solution candidate).
Afin de formuler la FA, certaines caractéristiques clignotantes des lucioles ont été idéalisées, comme suit:

- Toutes les lucioles sont unisexes.
- Leur attrait est proportionnel à leur intensité lumineuse.
- L'intensité lumineuse d'une luciole est affectée ou déterminée par le paysage de la fonction objective.

Notez que l'intensité lumineuse I et l'attractivité sont en quelque sorte synonymes. Bien que l'intensité I soit considérée comme une mesure absolue de la lumière émise par la luciole, l'attractivité est une mesure relative de la lumière qui devrait être vue aux yeux des observateurs et jugée par les autres lucioles.

$t = 0;\ s^* = \emptyset;\ \gamma = 1,0$;*//initialiser: gen.compteur, meilleure solution, attractivité*
$\boldsymbol{P}^{(t)} = InitFA()$; *// initialiser la population de luciole* $s_i^{(0)} \in \boldsymbol{P}^{(0)}$
tan que $t \leq MAX\_GEN$ ***faire***
  $\propto^{(t)} = AmphaNew()$; *// déterminer une nouvelle valeur de* $\propto$
  Évaluer $FA\left(\boldsymbol{P}^{(t)},\ f(\boldsymbol{s})\right)$; *// Évaluer* $\boldsymbol{s}_i^{(t)}$ *selon* $f(\boldsymbol{s}_i)$
  Commande $FA\left(\boldsymbol{P}^{(t)},\ f(s)\right)$; *// Trier* $\boldsymbol{P}_i^{(t)}$ *selon* $f(\boldsymbol{s}_i)$
  $s^* =$ *Trouvez le meilleur*$FA\left(\boldsymbol{P}^{(t)},\ f(\boldsymbol{s})\right)$;*//déterminer les meilleures solutions* s^*
  $\boldsymbol{P}^{(t+1)} = Déplacer\ fFA\left(\boldsymbol{P}^{(t)}\right)$; *// varier l'attractivité selon Eq.(14)*
  $t = t + 1$;
Fin tan que
Retour $s^*, f(s)$; *// processus postérieur*

Algorithme 3-2 : ORIGINAL LUCIOLE[43]

II- Documents BNT_StructureLearning_v1.3[26]:

Cette décomposition permet la création de quelques algorithmes d'inférence puissants pour lesquels les réseaux bayésiens sont devenus des modèles de modélisation et d'adaptation simples lorsque la situation est incertaine ou que les données sont incomplètes. Ensuite, les réseaux bayésiens sont utiles pour les problèmes de classification lorsque les interactions entre les caractéristiques peuvent être modélisées par une probabilité conditionnelle. Lorsque la structure du réseau n'est pas donnée (par un expert), il est possible de l'apprendre à partir des données. Cette tâche d'apprentissage est difficile, en raison de la complexité de l'espace de recherche.

Beaucoup de logiciels traitent des réseaux bayésiens, par exemple:

- Hugin [Andersen et al., 1989]

- Netica [Norsys, 2003]

- Bayesia Lab [Munteanu et al., 2001]

- TETRAD [Scheines et al., 1994]

- DEAL [Bøttcher & Dethlefsen, 2003]

- LibB

- the Matlab Bayes Net Toolbox [Murphy, 2001a]

Pour nos expériences, nous utilisons Matlab avec Bayes Net Toolbox [Murphy, 2001a] et l'Ensemble d'apprentissage de structure que nous développons et proposons sur notre memoire.

Ce document est organisé comme suit. Nous présentons quelques concepts généraux concernant les structures de réseau bayésiennes, comment évaluer ces structures et certaines propriétés intéressantes de la fonction de notation. Nous décrivons les méthodes communes pour effectuer l'apprentissage de la structure, de la recherche de causalité aux recherches heuristiques dans l'espace du réseau bayésien, et nous discutons des problèmes d'initialisation de telles méthodes. Nous comparons ces méthodes à l'aide de deux séries de tests. Dans la première série, nous voulons récupérer une structure connue, et dans l'autre, nous voulons obtenir un bon réseau bayésien pour les tâches de classification. Nous concluons ensuite sur les avantages et les inconvénients de ces méthodes, et discuterons des recherches futures pertinentes.

Nous décrivons la syntaxe d'une fonction telle qu'elle :

[out1, out2] = function(in1, in2)
Brève description de la fonction.
Ver dans le coin supérieur droit, spécifiez l'emplacement de la fonction: BNT s'il s'agit d'une fonction native de la boîte à outils BNT ou v1.3 si elle peut la trouver dans la dernière version de l'emballage
Les champs suivants sont des options:
INPUTS :
in1 - Description de l'argument d'entrée in1
in2 - Description de l'argument d'entrée in2
OUTPUTS :
out1 - Description de l'argument de sortie out1
out2 - Description de l'argument de sortie out2
Par exemple, out = function (in), Un exemple de la syntaxe d'appel.

Gs = mk_all_dags(n, order).
Génère tous les DAGs avec n nœuds selon la commande facultative

dag = pdag_to_dag(pdag)

Donne une instanciation d'un PDAG dans l'espace DAG tandis que c'est possible.

cpdag = dag_to_cpdag(dag)

Donne le PDAG complet d'un DAG (fonctionne également avec un réseau cellulaire de cpdags, renvoyant un ensemble de cellules de dags).

dag = cpdag_to_dag(cpdag)

Donne une instanciation d'un CPDAG dans l'espace DAG (fonctionne également avec un réseau cellulaire de cpdags, renvoyant un ensemble de cellules de dags).

D = compute_bnet_nparams(bnet)
donne le nombre de paramètres du réseau bayésien bnet

score = score_dags(Data, ns, G)
calculer le score ('bayesian' par défaut ou 'BIC' score) d'un dag G
Cette fonction existe dans BNT, mais la nouvelle version disponible dans le Package Structure utilise un cache pour éviter de recomposer tout le score local dans la score_family sous-fonction lorsque nous calculons un nouveau score global.
INPUTS :
Data{i,m} - valeur du nœud i dans le cas m (peut être un réseau de cellules).
ns(i) – taille du noeud i.
dags{g} – g'eme dag
Les arguments optionnels suivants peuvent être spécifiés sous la forme de paires ('nom', valeur): [valeur par défaut entre parenthèses]
scoring_fn - 'Bayesian' ou 'bic' ['bayesian'] actuellement, seuls les réseaux avec tous les noeuds tabulaires prennent en charge la notation bayésienne.
type - type {i} est le type de CPD à utiliser pour le nœud i, où le type est une chaîne de la forme 'tabular', 'noisy_or', 'gaussian', etc. [toutes les cellules contiennent 'tabular']
params - params{i} contient des arguments optionnels transmis au constructeur CPD pour le nœud i ou [] s'il n'y en a pas.
[toutes les cellules contiennent {'prior', 1}, ce qui signifie utiliser des priors Dirichlet uniformes] discrets - la liste des nœuds discrets [1: N]
clamped - clamped(i,m) = 1 Si le nœud i est serré dans le cas où m [zéros (N, ncases)]
cache - structure de données utilisée pour mémoriser les calculs de score local (voir la fonction SCORE_INIT_CACHE) [[]]
OUTPUT :
Le score (g) est le score du i'eme dag
par exemple score = score_dags (Data, ns, mk_all_dags (n), 'scoring_fn', 'bic', 'params', [], 'cache', cache);

En particulier, les cpdags peuvent être marqués par

```
score = score_dags (Data, ns, cpdag_to_dag (CPDAG), 'scoring_fn', 'bic')
```

Étant donné que le score global d'un DAG est le produit (ou la somme est notre cas en prenant le logarithme) des scores locaux. Il est judicieux d'oublier les scores locaux calculés. Nous pouvons le faire en utilisant une matrice de cache.

```
cache = score_init_cache(N,S);
INPUTS:
N – le nombre de nœuds
S - la longueur du cache
OUTPUT:
cache - les entrées sont le jeu parent, le noeud fils, le score de la famille et la méthode de notation
```

III- Méthode proposée :

L'idée principale de la méthode proposée est de trouver le meilleur score dans un réseau Bayésien. On utilise des méthodes pour la construction des graphes (DAG) et pour choisir permis ces graphes qu'est le meilleurs on utilisant l'algorithme COUCOU et on Comparer de notre résultat avec l'algorithme LUCIOLE

III-1- La formulation du problème :

La génération de la base " heart " passe par trois étapes :

1- importation des données.

2- construction des graphes (DAG).

3- trouver le meilleur graphe (DAG).

Etape1 : importer des données : figure 3-1 montre nous les ensembles des données contient notre base de test « HEART ».

Etape2 : pour la construction des graphes on a utilisée le fichier BNT_StructureLearning_v1.3 (voire page de 47 à 49).

Etape3 : pour trouver le meilleur score on a appliquer l'algorithme COUCOU et on fait un Comparaison avec l'algorithme LUCIOLE

IV- Expérimentation et résultats :

Nous avons teste notre méthodes sur la base de teste HEART.

IV-1- Langage Matlab 07 :

MATLAB (« matrix laboratory ») est un langage de programmation de quatrième génération émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs. Matlab peut s'utiliser seul ou bien avec des toolbox (« boîte à outils »).

IV-2- La base de teste HEART :

Soit un base « heart » : Cet ensemble de données, disponible dans le projet Statlog [Sutherland & Henery, 1992, Michie et al., 1994], est un ensemble de données de diagnostic médical avec 14 attributs (les attributs continus ont été discrétisés). Cet ensemble de données a 270 données que nous nous décomposons en 189 données d'apprentissage et 81 données de test.

```
x =

          N: 14
      NbApp: 189
     NbTest: 81
    melange: [1x270 double]
     BDMapp: {14x189 cell}
        app: [14x189 double]
       test: [14x81 double]
       diag: 14
      class: 14
   discrete: [1 2 3 4 5 6 7 8 9 10 11 12 13 14]
node_sizes: [5 2 4 5 4 2 3 5 2 5 3 4 7 2]
node_types: {1x14 cell}
```

Figure 3-1 : Ensemble de données de la base heart

V- Présentation de l'application :

Notre application permet de :

1- ouvrir pour importer la base heart.

2- ensembles des données de la base « heart » .

3- Affichage des scores

4- les 05 meilleurs scores pour la recherche COUCOU

5- les 05 meilleurs scores pour la recherche LUCIOLE

6- affiche les graphes.

7- Nombre itération

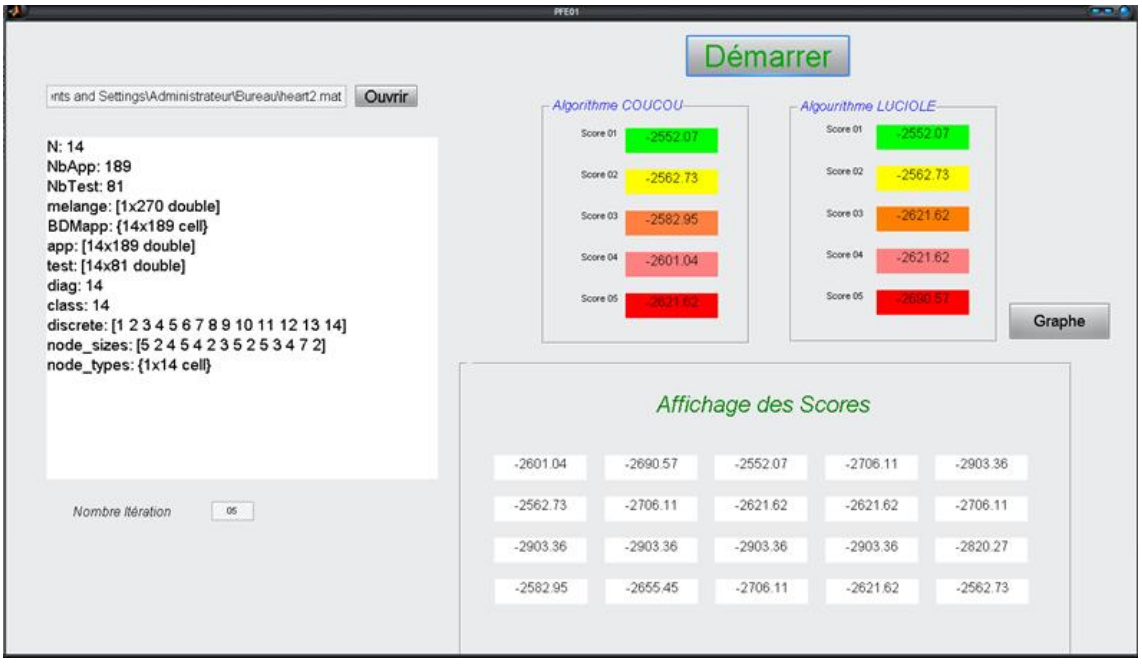
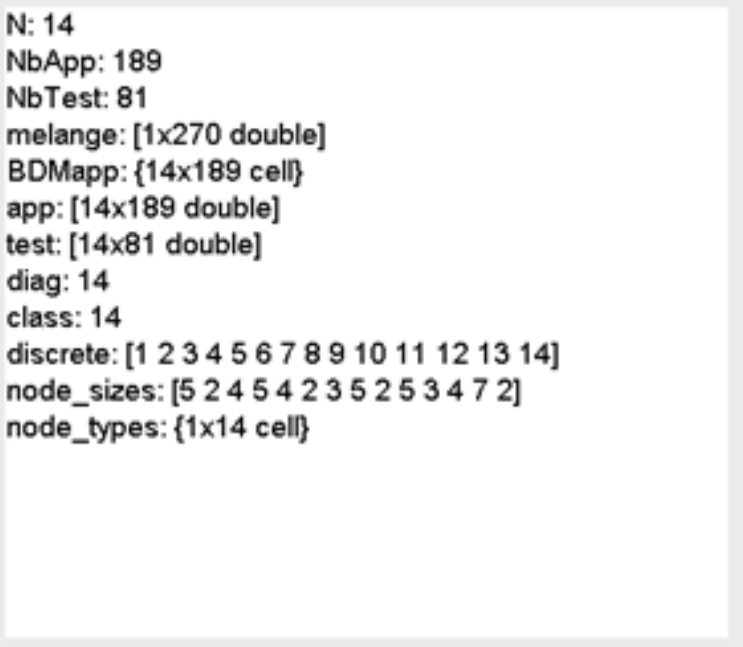


Figure 3-2 : Représentation de l'application

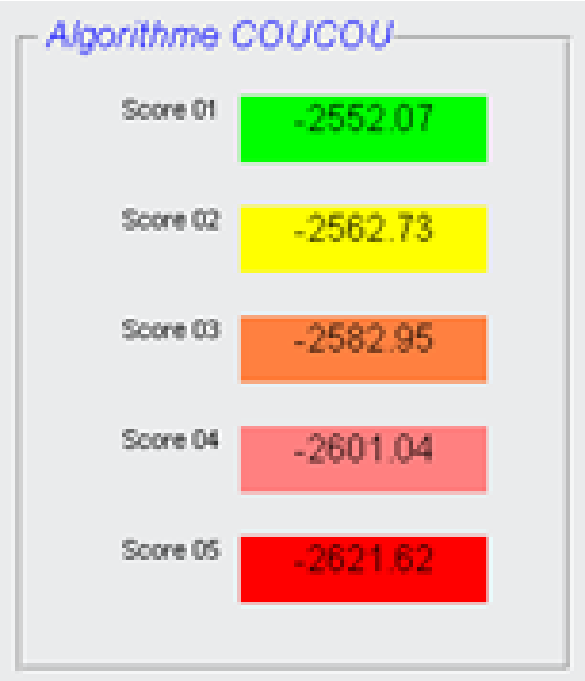
1- ouvrir pour importer la base heart.

2- affichage des ensembles de donnée de la base « heart » .

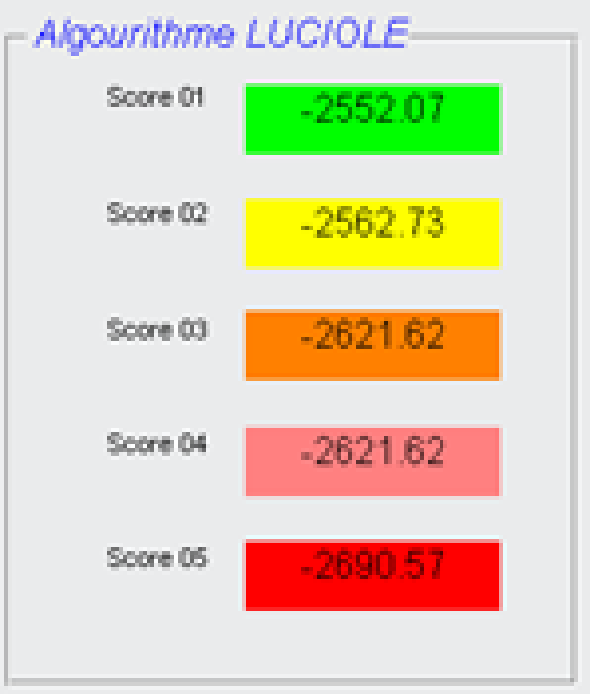


3- Affichage des meilleurs scores

4- affichage les 05 meilleurs scores pour la recherche COUCOU



5- affichage les 05 meilleurs scores pour la recherche LUCIOLE



6- affichage des graphes

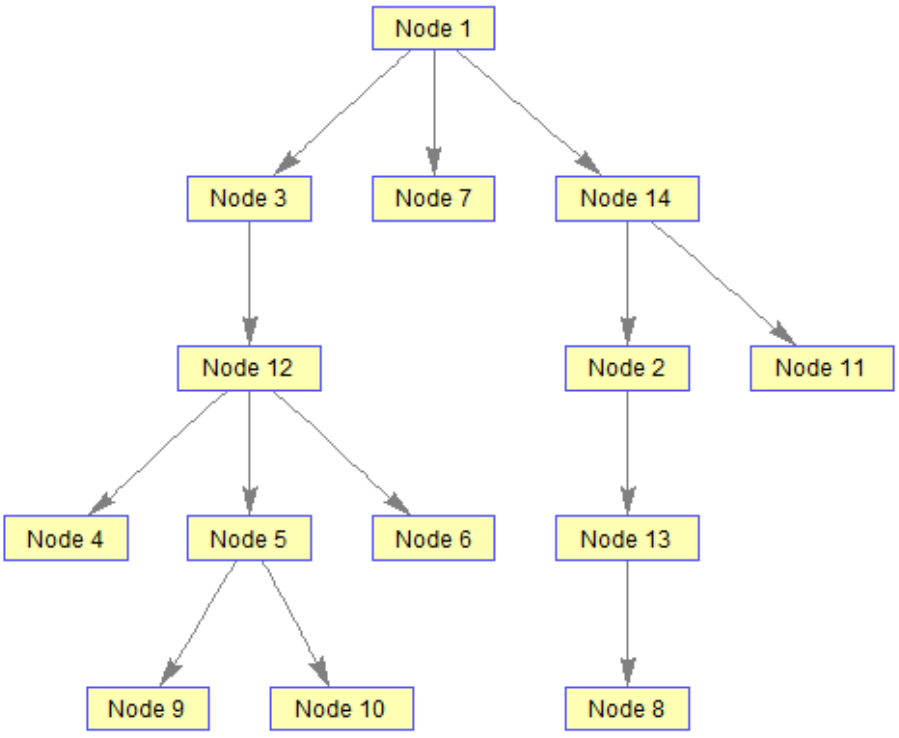
.

7- Nombre itération (on peut changer le nombre d'itération)

Présentation quelques graphes :

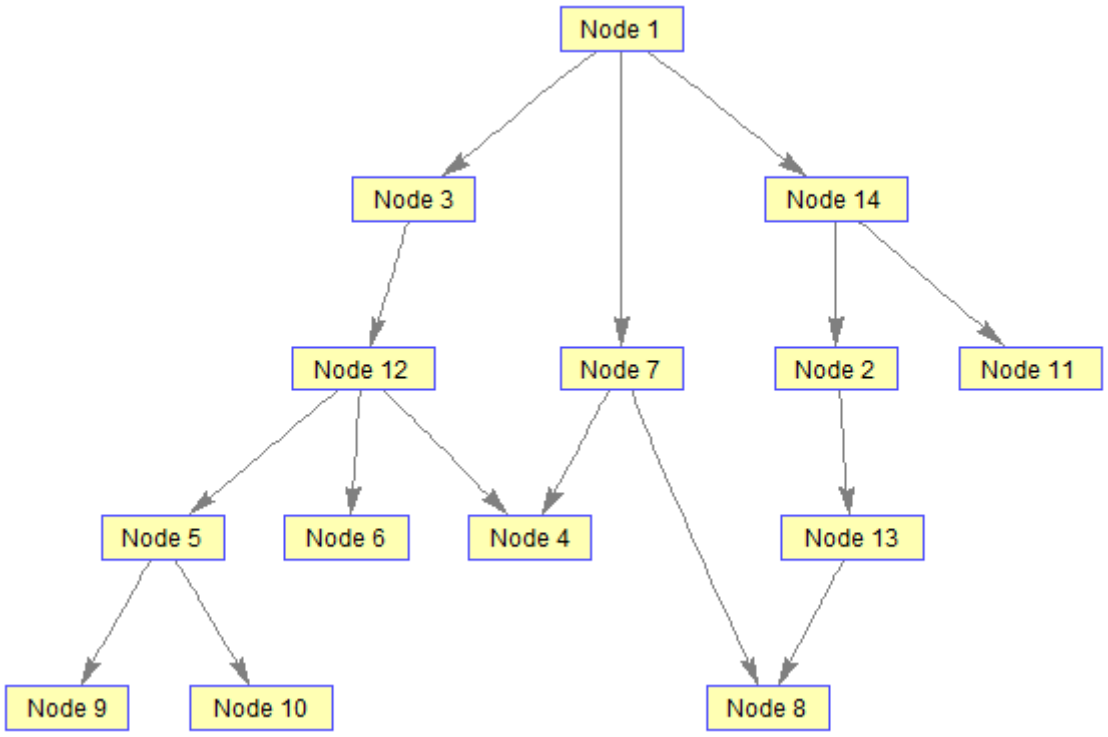
```
G =

     0     0     1     0     0     0     1     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     1     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     1     0     0     0
```

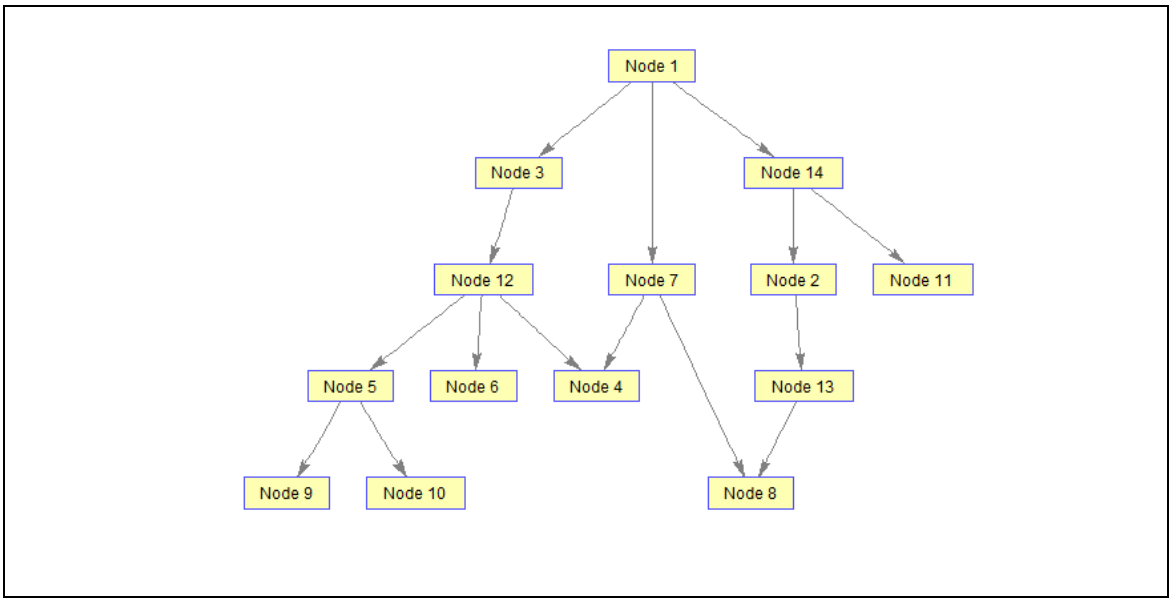


G =

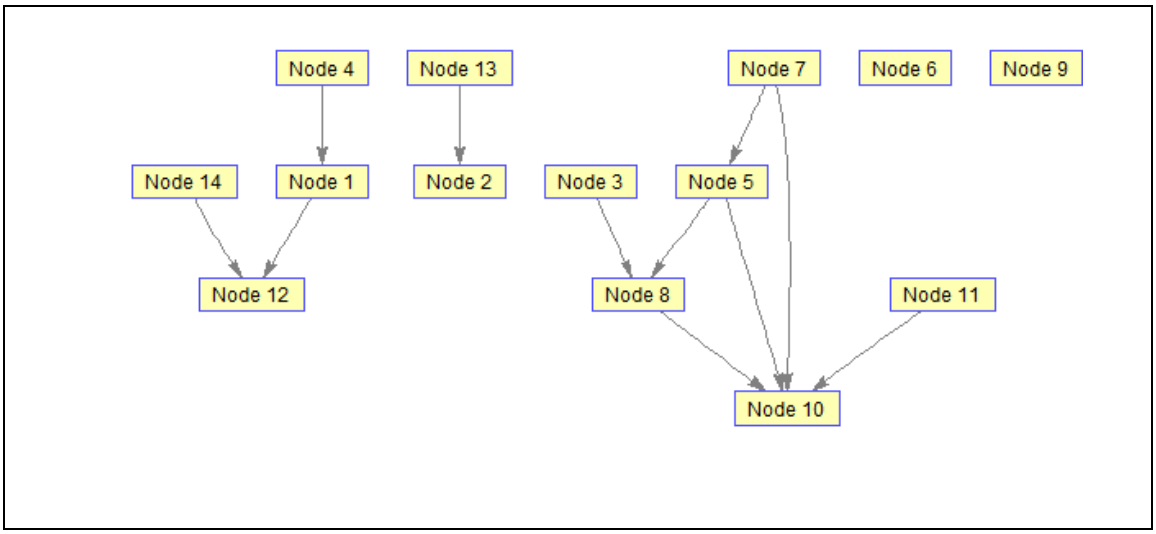
0	0	1	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0



Comparaison entre l'algorithme COUCOU et LUCIOLE :



Comparaison du temps d'exécution pour un graphe 1 (t/s)	
Temps d'exécution Algorithme COUCOU	**Temps d'exécution Algorithme LUCIOLE**
1.9011	**1.9049**



Comparaison du temps d'exécution pour un graphe 2 (t/s)	
Temps d'exécution Algorithme COUCOU	**Temps d'exécution Algorithme LUCIOLE**
1.8153	**1.8455**

VI- Conclusion :

L'apprentissage de la structure du réseau bayésien à partir des données est un problème difficile pour lequel nous avons examiné les principales méthodes existantes.

Notre expérience nous a permis d'évaluer ces méthodes en récupérant un graphique connu. Pour la plupart des méthodes, les initialisations aléatoires peuvent être remplacées efficacement par des initialisations issues d'un algorithme simple comme MWST.

Dans notre travail on a montré que l'algorithme coucou et plus efficace pour atteindre le meilleurs scores (score maximum) dans une base de données connue par contre la recherche Luciole et moins d'efficacité pour aboutir aux meilleurs scores bayésiens (score maximum).

Conclusion

Générale

Conclusion générale :

Nous nous sommes intéressés dans ce mémoire à l'Application de la méthode de recherche COUCOU et LUCIOLE pour l'apprentissage de structure des réseaux bayésien, à partir des données est un problème difficile pour lequel nous avons examiné les principales méthodes existantes.

D'après la comparaison faite entres les deux méthodes pour extraire la structure d'un Réseaux Bayésiens, nous constatons que :

1) La méthode de recherche COUCOU est plus rapide que la méthode LUCIOLE.
2) Ces deux méthodes ont obtenu presque les mêmes résultat.

Comme perspectives nous envisageons de :

- Tester notre méthode sur d'autres bases,
- Introduire de nouveaux mécanismes pour les scores d'un graphe bayésien (DAG).
- La comparaison des résultats de notre travail avec d'autres travaux.

Comparer avec d'autre méta-heuristique (Algorithme BAT Search « chauve souris » , WASP SWARM BASED Algorithme « essaim de guêpe », …).

Références

Bibliographiques

Références Bibliographiques

[1] Brajevic et al : Multilevel image thresholding selection based on the cuckoo search algorithm. In: Proceedings of the 5th InternationalConference onVisualization, Imaging and Simulation (VIS'12), Sliema, Malta, pp. 217–222 (2012).

[2] Chaowanawatee et al : Implementation of cuckoo search in rbf neural network for flood forecasting. In: 2012 Fourth InternationalConference on Computational Intelligence, Communication Systems and Networks (CICSyN), pp. 22–26. IEEE (2012).

[3] Chifuet al : Bio-inspired methods for selecting the optimal web service composition: Bees or cuckoos intelligence? Int. J. Bus. Intell. Data Min. 6(4), 321–344 (2011).

[4] Chifu et al : Optimizing the semantic web service composition process using cuckoo search. In: Intelligent Distributed Computing V, pp. 93–102. Springer (2012).

[5] Dejam et al : Combining cuckoo and tabu algorithms for solving quadratic assignment problems. J. Acad. Appl. Stud. 2(12), 1–8 (2012).

[6] Dhivya et al : Cuckoo search for data gathering in wireless sensor networks. Int. J. Mob. Commun. 9(6), 642–656 (2011).

[7] Dhivyaet al : Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (cbpa). IJCNS 4(4), 249–255 (2011).

[8] Dhivya et al : Energy efficient cluster formation in wireless sensor networks using cuckoo search. In: Swarm, Evolutionary, and Memetic Computing, pp. 140–147. Springer (2011).

[9] Elkera et al: A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. Eur. J. Oper. Res. (2013) http://dx.doi.org/10.1016/j.ejor.2013.06.020.

[10] Etude comparative d'algorithmes d'apprentissage de structure dans les Réseaux Bayésiens, Olivier Françoit et Philippe Leray , Laboratoire Peception, Systèmes, Information –FRE CNRS 2645

[11] Goel et al : Cuckoo search clustering algorithm: a novel strategy of biomimicry. In: World Congress on Information and Communication Technologies (WICT), 2011, pp. 916–921. IEEE (2011).

[12] Gupta et al : Applying case based reasoning in cuckoo search for the expedition of groundwater exploration. In: Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), pp. 341–353. Springer (2013).

[13] Joshi et al : Query optimization: an intelligent hybrid approach using cuckoo and tabu search. Int. J. Intell. Inf. Technol. (IJIIT) 9(1), 40–55 (2013).

[14] Kanagaraj et al : Supplier selection: reliability based total cost of ownership approach using cuckoo search. In: Trends in Intelligent Robotics, Automation, and Manufacturing, pp. 491–501. Springer (2012).

[15] Kavousi-Fard et al : A new hybrid correction method for short-term load forecasting based on arima, svr and csa. J. Exp. Theor. Artif. Intell. 2013(ahead-of-print), 1–16 (2013).

[16] Madi´c et al : Application of cuckoo search algorithm for surface roughness optimization in co2 laser cutting. Ann. Fac. Eng. Hunedoara Int. J. Eng. 11(1), 39–44 (2013).

[17] Marichelvam et al : An improved hybrid cuckoo search (ihcs) metaheuristics algorithm for permutation flow shop scheduling problems. Int. J. Bio-Inspired Comput. 4(4), 200–205 (2012).

[18] Mellal et al : Optimal replacement policy for obsolete components using cuckoo optimization algorithm based-approach: dependability context. J. Sci. Ind. Res. 71, 715–721 (2012).

[19] Moravej et al : A novel approach based on cuckoo search for dg allocation in distribution network. Int. J. Electr. Power Energy Syst. 44(1), 672–679 (2013)

[20] Natarajan et al : An enhanced cuckoo search for optimization of bloom filter in spam filtering. Global J. Comput. Sci. Technol. 12(1), 1–9 (2012).

[21] Natarajan et al : A comparative study of cuckoo search and bat algorithm for bloom filter optimisation in spam filtering. Int. J. Bio-Inspired Comput. 4(2), 89–99 (2012).

[22] Nawi et al : A new back-propagation neural network optimized with cuckoo search algorithm. In: Computational Science and Its Applications-ICCSA 2013, pp. 413–426. Springer (2013).

[23] Ouaarab et al : Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. 1–11. Springer (2013).

[24] Patrick Naim et al -réseaux Bayésiens, EYROLLES- 3eme édition- EYROLLES.

[25] Philippe LERAY, these Réseaux bayésiens : apprentissage et modélisation de systèmes complexes, Université de Rouen, novembre 2006

[26] Philippe Leray et al ; BNT Structure Learning Package : Documentation and Experiments ; 8th November 2004, Version 1.3

[27] Pop et al: Cuckoo-inspired hybrid algorithm for selecting the optimal web service composition. In: IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2011, pp. 33–40. IEEE (2011).

[28] Prakash et al : An optimal job scheduling in grid using cuckoo algorithm. Int. J. Comput. Sci. Telecomm. 3(2), 65–69 (2012).

[29] R. Robinson. Counting unlabeled acyclic digraphs. In C. Little, editor, Combinatorial Mathematics V, volume 622 of Lecture Notes in Mathematics, pages 28–43, Berlin, 1977. Springer.

[30] Ritze et al : Towards an automatic parameterization of ontology matching tools based on example mappings. In: Proceedings of the Sixth InternationalWorkshop on Ontology Matching at ISWC, vol. 814, p. 37 (2011).

[31] Sharma et al : Impelementation of n-queens puzzle using metaheuristic algorithm (cuckoo search). Int. J. Latest Trends Eng. Technol. (IJLTET) 2(2), 343–347 (2013).

[32] Sood et al : Speaker recognition based on cuckoo search algorithm. Int. J. Innovative Technol. Explor. Eng. (IJITEE) 2(5), 311–313 (2013).

[33] Speed et al : Evolving a mario agent using cuckoo search and softmax heuristics. In: International IEEE Consumer Electronics Society's Games Innovations Conference (ICE-GIC), 2010, pp. 1–7. IEEE (2010).

[34] Speed et al : Artificial intelligence for games. US Patent App. 13/309,036, 1 Dec 2011.

[35] Srivastava et al : Automated test data generation using cuckoo search and tabu search (csts) algorithm. J. Intell. Syst. 21(2), 195–224 (2012).

[36] Srivastava et al : Software test effort estimation: a model based on cuckoo search. Int. J. Bio-Inspired Comput. 4(5), 278–285 (2012).

[37] Syberfeldt et al : Real-world simulation-based manufacturing optimization using cuckoo search. In: Proceedings of the Winter Simulation Conference, pp. 1–12. Winter Simulation Conference (2012).

[38] Tiwari et al : Face recognition based on cuckoo search algorithm. Image 7(8), 9 (2012).

[39] Vázquez et al : Training spiking neural models using cuckoo search algorithm. In: IEEE Congress on Evolutionary Computation (CEC), 2011, pp. 679–686. IEEE (2011).

[40] Vo et al : Cuckoo search algorithm for non-convex economic dispatch. IET Gener. Transm. Distrib. 7(6), 645–654 (2013).

[41] Wang et al : A hybrid meta-heuristic de/cs algorithm for ucav path planning. J. Inf. Comput. Sci. 5(16), 4811–4818 (2012).

[42] Wang et al : A hybrid metaheuristic de/cs algorithm for ucav three-dimension path planning. Sci. World J. (2012) doi:10.1100/2012/
583973.

[43] Xin-She Yang (auth.), Xin-She Yang (eds.); Cuckoo Search and Firefly Algorithm: Theory and Applications; Springer International Publishing; 2014

[44] Yang et al : Cuckoo search for business optimization applications. In: Computing and Communication Systems (NCCCS), 2012, pp. 1–5. IEEE (2012).

[45] Yildiz et al : Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. Int. J. Adv. Manuf. Technol. 64(1–4), 55–61 (2013).

Liste des Figures

Liste des Figures

Liste des Tableaux

Liste des Tableaux

Liste des Algorithmes

Liste des Algorithmes

Liste des Abréviations

Liste des Abréviations

ABC : la colonie artificielle d'abeilles

BCS : binaire couocu cherche

CSS : recherche par système chargé

CS : coucou search

DAG : graphe acyclique dirigé

DE : Evolution Différentielle

EM : Expectation Maximisation

FA : algorithme luciole (Firefly Algorithm)

GS : Algorithme Recherche Gloutonne

GSA : algorithme de recherche gravitationnelle

IFA : Intelligent Algorithme Luciole

K2 : l'algorithme K2

MAP : approche de maximum a posteriori

MAR : Missing At Random

MCAR : Missing Completly At Random

MWST : Maximum Weight Spanning Tree

NMAR : Not Missing At Random

PC : [Peter Spirtes, Clark Glymour] recherche de causalité

PSO : L'optimisation par essaim de particules (de l'anglais: Particle Swarm Optimization)

RB : réseaux bayésiens

SA : Le recuit simulé (de l'anglais: Simulated Annealing)

SAFE : l'algorithme de firefly asynchrone séquentiel

SGS : [Spirtes, Glymour et Scheines] recherche SGS

TSP : Travelling Salesman Problem

Abstract:

The aim of this thesis is to propose methods for solving difficult academic optimization problems. In order to achieve our goal, we proposed two algorithms inspired by nature. Our first methods to propose are the CUCKOO Algorithm. Our second method is the FIREFLY Algorithm. The objective of these two methods and allowing solving problems of structure learning Bayesian.

Keywords: optimization problems, CUCKOO algorithm, FIREFLY algorithm, Bayesian networks

Résumé :

L'objectif de ce mémoire est de proposer des méthodes pour la résolution de problèmes d'optimisation académiques difficiles. Dans le but de réaliser notre objectif, nous avons proposé deux algorithmes inspiré de la nature. Notre premier méthodes a proposer est l'Algorithme COUCOU. Notre deuxième méthode est l'Algorithme LUCIOLE. L'objectif de ces deux méthodes et permettant la résolution de problème d'apprentissage des structures Bayésiens (score maximum).

Mots clés: problèmes d'optimisation, Algorithme COUCOU, Algorithme LUCIOLE, Réseaux Bayésiens

ملخص:

الهدف من هذه المذكرة هو اقتراح أساليب لحل مشاكل التحسين الأكاديمي الصعب. من أجل تحقيق هدفنا، اقترحنا خوارزميتين مستوحاة من الطبيعة. تطرقنا في الأولى لاقتراح خوارزمية كوكو. وفي الثانية تطرقنا الى خوارزمية لوسيول. والهدف من هاتين الطريقتين هو السماح لحل مشاكل التحسين من المحنى البياني الذي يمثل الشبكات البايزيان .

الكلمات الرئيسية: مشاكل التحسين، خوارزمية كوكو، خوارزمية لوسيول، شبكات بايزيان.