الجمـــهوريـــة الجـــزائريـــة الديـــمقراطيـــة الشعبيـــة
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـــم العـالـــي و البـــحث العلـــمي
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـــامعـة عمـــار ثليـجي بالأغـــواط
UNIVERSITÉ AMMAR TELIDJI LAGHOUAT

كـلـيـــة العلـــوم و الـهـنـدسـة
FACULTÉ DES SCIENCES ET SCIENCES DE L'INGÉNIEUR
DÉPARTEMENT DE GÉNIE INFORMATIQUE

MÉMOIRE DE MASTER

DOMAINE : MATHÉMATIQUES ET INFORMATIQUE **(MI)**
FILIÈRE : INFORMATIQUE
OPTION. : RÉSEAUX, SYSTÈMES ET APPLICATIONS RÉPARTIS **(ReSar)**

Présenté par :
Matallah Souad

Thème :

Sélection Des Index De jointure Binaires Par Algorithme Génétique

Soutenu devant le jury:

B. KERROUCHE	MA	U. Laghouat	(Président)
L. CHELLAMA	MA	U. Laghouat	(Examinateur)
Y. Ouinten	MC	U. Laghouat	(Encadreur)
B. Ziani	MA	U. Laghouat	(Co-Encadreur)

ANNÉE UNIVERSITAIRE 2010/2011

Remerciements

Je remercie ALLAH le tout puissant qui m'a guidé et qui m'a donné la force et la volonté de réaliser ce modeste travail.

Je remercie Monsieur Ouinten Youcef et Monsieur Ziani Benameur, de m'avoir proposé ce sujet de mémoire. Je les remercie infiniment pour leurs précieux conseils et leurs disponibilités.

Je tiens également à remercier le président et les membres de jury de soutenance, pour l'honneur qu'ils me font en acceptant d'examiner ce mémoire, et de consacrer une partie de leurs temps.

Enfin, toute ma gratitude, ma reconnaissance à tous ceux qui ont contribué de prés ou de loin et en particulier l'ensemble des enseignants du département d'informatique, à ma formation de Master.

Dédicaces

Je dédie le présent travail

A mes très chers parents

A mes sœurs et mes frères

A l'ensemble de ma famille et mes amies

A mes collègues de travail

Chapitre I :

État de l'Art

Introduction

Les entrepôts de données stockent un volume de données très important essentiellement dans des modèles logiques relationnels comme les schémas en étoile ou flocon de neige[1]. Ces derniers sont accédés par des requêtes décisionnelles complexes caractérisées par de multiples opérations de jointure, de sélection et d'agrégation. Les requêtes définies sur un schéma en étoile sont appelées requêtes de jointure en étoile comportant un nombre de prédicats de sélection définis sur des tables de dimension et des prédicats de jointures entre la table des faits et les tables de dimension.

Interroger des tables volumineuses via un ensemble de requêtes pour accéder à un certain nombre de n-uplets est une tâche fréquente dans un environnement d'entrepôt de données. Répondre efficacement à ces requêtes est souvent difficile compte tenu de la nature complexe des requêtes et des volumes de données. La manière la plus facile de procéder consiste à effectuer un balayage complet des tables et vérifier pour chaque n-uplet s'il satisfait le prédicat de la requête. Ce balayage peut être très coûteux lorsque les tables scannées sont volumineuses. Pour pallier ce problème, plusieurs techniques d'indexation ont été proposées. Un index est une structure redondante ajoutée à la base de données pour permettre les accès rapides aux données. Il permet à partir d'une clé d'index de trouver l'emplacement physique des n-uplets recherchés. Parmi les techniques d'indexation proposées dans le cadre des bases de données classiques, nous pouvons citer l'index B-tree, l'index de hachage, l'index de projection, l'index de jointure, etc. La majorité de ces index est aussi utilisée dans le cadre des entrepôts relationnels. Certaines techniques d'indexation sont apparues dans le contexte

d'entrepôts de données comme les index binaires, les index de jointure binaires, les index de jointure en étoile, etc. Nous pouvons classer les index proposés en deux catégories, les index mono-table et les index multi-tables. Les index mono-table sont des index définis sur un ou plusieurs attributs de la même table, comme les index B-tree, de hachage, binaires, etc. Les index multi-tables sont des index définis sur plusieurs tables comme les index de jointure standards, en étoile et binaires.

I .1 Les index de jointure binaires

Nous présentons dans cette section une techniques d'optimisation redondante étudiée dans le cadre de cette thèse : les index de jointure Binaire .

I.1.1 Définition

L'IJB sert à précalculer les jointures entre une ou plusieurs tables de dimension et la table des faits dans les entrepôts de données modélisés par un schéma en étoile [2] ;[3]. Au contraire des index binaires standards où les attributs indexés appartiennent à la même table, l'IJB est défini sur un ou plusieurs attributs appartenant à plusieurs tables. Plus précisément, il est défini sur la table des faits en utilisant des attributs appartenant à une ou plusieurs tables de dimension.

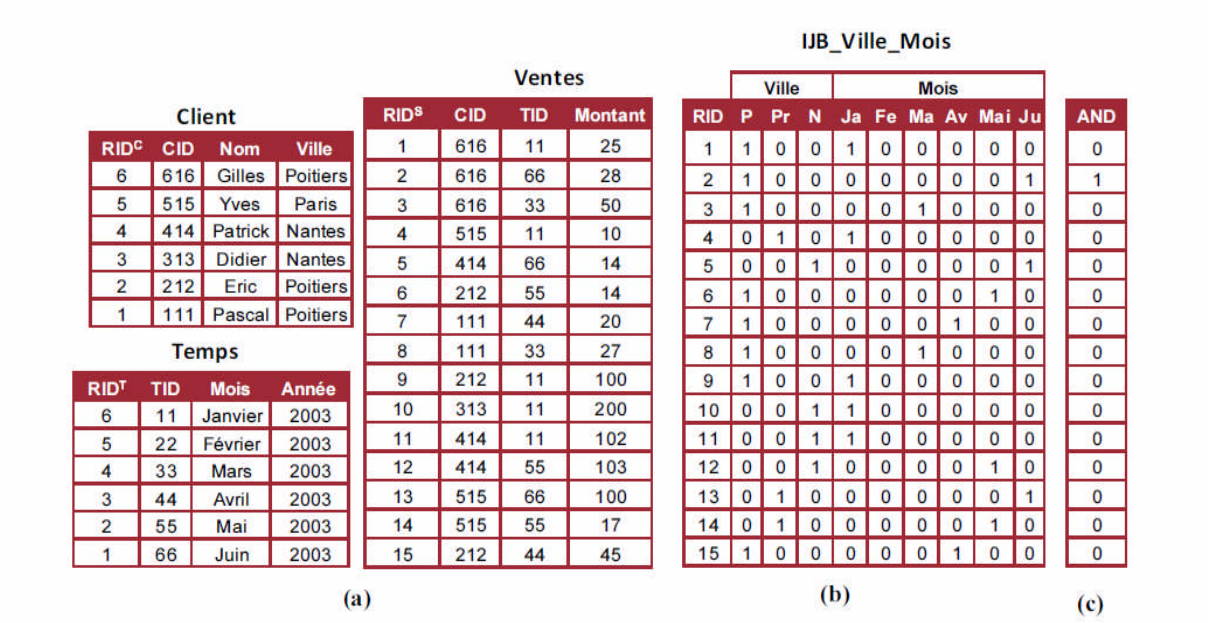
Pour montrer comment un IJB est construit, supposons un attribut *A* ayant *n* valeurs distinctes v_1 , v_2 , …, v_n appartenant à une table de dimension *D*. Supposons que la table des faits *F* est composée de m instances. La construction de l'index de jointure binaire défini sur l'attribut *A* se fait de la manière suivante :

1. Créer n vecteurs composés chacun de m entrées ;
2. Le ième bit du vecteur correspondant à une valeur v_k est mis à 1 si le n-uplet de rang *i* de la table des faits est joint avec un n-uplet de la table de dimension *D* tel que la valeur de *A* de ce n-uplet est égale à v_k . Il est mis à 0 dans le cas contraire.

La nature binaire des IJB permet d'améliorer les performances des requêtes en permettant d'appliquer des opérations logiques AND, OR, NOT, etc. Ces opérations permettent de rechercher des n-uplets vérifiant des conjonctions ou des disjonctions de prédicats. Les IJB sont très bénéfiques pour les requêtes de type Count(*) où l'accès à l'index binaire seulement permet de répondre à ces requêtes.

Notons que la taille d'un IJB est proportionnelle à la cardinalité des attributs indexés (nombre de valeurs distinctes). Pour cette raison, il sont souvent recommandés pour les

attributs de faible cardinalité. Pour réduire la taille d'un index binaire, plusieurs techniques de compression ont été proposées [4].



Client

RID^C	CID	Nom	Ville
6	616	Gilles	Poitiers
5	515	Yves	Paris
4	414	Patrick	Nantes
3	313	Didier	Nantes
2	212	Eric	Poitiers
1	111	Pascal	Poitiers

Temps

RID^T	TID	Mois	Année
6	11	Janvier	2003
5	22	Février	2003
4	33	Mars	2003
3	44	Avril	2003
2	55	Mai	2003
1	66	Juin	2003

(a)

Ventes

RID^S	CID	TID	Montant
1	616	11	25
2	616	66	28
3	616	33	50
4	515	11	10
5	414	66	14
6	212	55	14
7	111	44	20
8	111	33	27
9	212	11	100
10	313	11	200
11	414	11	102
12	414	55	103
13	515	66	100
14	515	55	17
15	212	44	45

IJB_Ville_Mois

	Ville			Mois					
RID	P	Pr	N	Ja	Fe	Ma	Av	Mai	Ju
1	1	0	0	1	0	0	0	0	0
2	1	0	0	0	0	0	0	0	1
3	1	0	0	0	0	1	0	0	0
4	0	1	0	1	0	0	0	0	0
5	0	0	1	0	0	0	0	0	1
6	1	0	0	0	0	0	0	1	0
7	1	0	0	0	0	0	1	0	0
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0
11	0	0	1	1	0	0	0	0	0
12	0	0	1	0	0	0	0	1	0
13	0	1	0	0	0	0	0	0	1
14	0	1	0	0	0	0	0	1	0
15	1	0	0	0	0	0	1	0	0

(b)

AND
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0

(c)

Figure.I.1 : *Exemple des index de jointure binaire* [5]

I.1.2 Stratégie d'exécution en présence des IJB

Une requête de jointure en étoile est caractérisée par un ensemble de sélections sur les tables de dimension, suivies de jointures avec la table des faits. Si tous les attributs objets des prédicats de sélection sont indexés, alors l'exécution d'une requête de jointure passe par les étapes suivantes :

1. Effectuer une réécriture de la requête qui consiste à séparer chaque jointure dans une sous-requête. Chaque sous-requête représente l'ensemble des sélections effectuées sur chaque table de dimension.
2. Pour chaque sous-requête, utiliser les IJB définis sur la table de dimension pour trouver un vecteur de bits représentant les n-uplets de la table des faits référencés par la sous-requête.
3. Effectuer une opération AND entre les vecteurs obtenus à partir des sous-requêtes pour trouver un seul vecteur référençant tous les n-uplets référencés par la requête.

4. Utiliser le vecteur résultat pour accéder à la table des faits et récupérer les n-uplets référencés par la requête globale.

Pour comprendre l'utilisation des IJB, nous supposons l'exemple suivant :

Exemple 1 : Soit l'entrepôt de données composé d'une table de faits ventes et deux tables de dimension Client et Temps représenté dans la figure I.1(a). Soit la requête *Q1* suivante :

*Select count(*)*
from Ventes V,Client C, Temps T
where V.CID=C.CID AND V.TID=T.TID
AND C.Ville='Poitiers' AND T.Mois='Juin'.

Pour réduire le coût de Q1 l'administrateur créé un IJB multi-attributs sur Ville et Mois comme suit :

CREATE BITMAP INDEX IJB_Ville_Mois
ON Ventes(Client.Ville, Temps.Mois)
FROM Ventes V, Client C, Temps T
WHERE V.CID= C.CID AND V.TID=T.TID

Cet index est représentée dans la figure I.1(b). Pour répondre à *Q1*, l'optimiseur lit les vecteurs de bits associés aux valeurs "Poitiers" et "Juin", effectue une opération AND et calcule le nombre de "1" dans le vecteur résultats (Voir figure I.1(c)).

I.1.3 Problème de sélection des index de jointure binaires

Nous présentons dans cette section une formalisation du problème de sélection des IJB, ensuite nous étudions sa complexité et nous présentons les principaux travaux effectués pour le résoudre.

I.1.3.1 Formalisation

La sélection d'une configuration d'IJB vise à optimiser la performance d'un ensemble de requêtes de jointure en étoile. Nous considérons l'espace de stockage réservé aux index comme une contrainte du problème de sélection des IJB. En conséquence, le problème de sélection d'une configuration d'IJB peut être formalisé comme suit :

Étant donné : (1) un entrepôt de données modélisé par un schéma en étoile ayant un ensemble de tables de dimension $D = \{D_1 , D_2 , ..., D_d \}$ et une table des faits F , (2) une charge de requêtes $Q = \{Q_1 , Q_2 , ..., Q_m \}$, où chaque requête Q_j a une fréquence d'accès f_j , et (3) une capacité de stockage S ;

Le problème de sélection des index de jointure consiste à trouver une configuration d'index *CI* minimisant le coût d'exécution de *Q* en respectant la contrainte de stockage ($Taille(CI) \leq S$).[5]

I.1.3.2 Complexité

La sélection d'une configuration d'IJB est généralement une tâche difficile comparée à d'autres types d'index. Cela est dû à plusieurs considérations :

– Un IJB est défini sur un ensemble d'attributs indexables qui représentent un sous-ensemble des attributs non clés des tables de dimension. Un attribut est indexable s'il est utilisé par un prédicat de sélection. Dans le contexte d'entrepôts de données, le nombre d'attributs indexables peut être important, vu le nombre de tables de dimension et le nombre d'attributs non clés de chaque table.

– Un IJB peut être défini sur un ou plusieurs attributs issus de différentes tables de dimension,

ce qui augmente le nombre d'IJB possibles.

– Un attribut indexable peut figurer dans plusieurs IJB car ces derniers peuvent être non disjoints.

– La taille d'un IJB dépend de la cardinalité des attributs indexés. Un attribut de forte cardinalité rend l'index volumineux, donc difficile à stocker et à maintenir.[5]

Soit $A = \{A_1 , A_2 , ..., A_K \}$ un ensemble d'attributs indexables candidats pour la sélection d'une configuration d'IJB. Chaque IJB dans cette configuration est constitué d'un sous ensemble d'attributs de *A*, par conséquent cette configuration constitue une partition de *A* en *n* ensemble de groupes. Chaque groupe d'attributs représente un IJB potentiel. Nous pouvons considérer deux scenarii :

1. Sélection d'un seul index de jointure : si on veut utiliser un seul IJB, le nombre de possibilité est donné par :

$$N = \binom{k}{1} + \binom{k}{2} + ... \binom{k}{k} = 2^k - 1 \qquad (1)$$

Si le nombre d'attributs indexables est égal à 5 (K = 5), alors le nombre d'index possible est égal à 31.

2. Sélection de plus d'un index de jointure : pour sélectionner plus d'un IJB, le nombre de possibilités est donné par :

$$N = \binom{2^k-1}{1} + \binom{2^k-1}{2} + \ldots + \binom{2^k-1}{2^k-1} = 2^{2^k-1} - 1 \qquad (2)$$

Par exemple, si le nombre d'attributs indexables est égal à 5, alors $N = 2^{31} > 1.2 \times 10^9$

I.1.4 Modèle de coût

Evaluer et comparer la qualité des différentes configurations d'index générées lors du processus de sélection nécessite l'utilisation d'un modèle de coût. Nous avons utilisé le modèle de coût proposé par Aouiche dans [6]. A partir d'une requête *Q* et d'une configuration d'index *CI*, ce modèle permet d'estimer la taille des index de *CI* ainsi que le coût d'exécution de *Q* en termes de nombre d'entrées-sorties en présence de *CI*. Il utilise un ensemble de paramètres dont les plus importants sont représentés sur le tableau I.1

I.1.4.1 Coût de stockage d'un IJB

Un IJB construit sur un ensemble d'attributs de dimension, stocke pour chaque n-uplet de la table des faits son identificateur de ligne (*RowID*) ainsi qu'un ensemble de vecteurs binaires représentant chacune une valeur des attributs indexés. L'espace de stockage d'un IJB dépend de deux paramètres : le nombre de n-uplets de la table des faits et la cardinalité des attributs indexables.

Soit un index *I* défini sur *n* attributs $A_1, A_2, \cdots, A_n$. L'espace de stockage en octets de *I* est calculé par la formule suivante :

$$Taille(I) = \frac{\left(\left|RowID\right| + \sum_{j=1}^{n}\left|A_j\right|\right) \times \left|F\right|}{8}$$

Paramètre	Signification
I	Un index de jointure binaire
RowID	Taille de l'identificateur de ligne en bits
$\|A_j\|$	Cardinalité de l'attribut A_j.
$\|F\|$	Cardinalité de la table des faits F
$\|\|F\|\|$	Nombre de pages nécessaires pour stocker F
Taille(I)	Taille d'un index I
PS	Taille de la page système

Tab I.1 *Paramètres utilisés dans le modèle de coût*

I.1.4.2 Coût d'exécution

Le coût d'exécution d'une requête Q en présence d'une configuration d'index *CI* est exprimé en nombre d'entrées-sorties nécessaires pour l'exécution de cette requête en utilisant *CI*. L'exécution de Q dans ce cas, passe par deux étapes importantes : le chargement des index, ensuite l'accès aux données. Par conséquent, deux coûts sont considérés : le coût de chargement des index et le coût d'accès aux données.

- Coût de chargement des index : le coût de chargement d'un IJB noté *CC(I)* correspond au nombre de pages lues pour le charger. Il est calculé par la formule suivante : $CC(I) = \frac{Taille(I)}{PS}$.
- Le coût de chargement de l'ensemble des index utilisés par une requête Q correspond à la somme des coûts de chargement de chacun de ces index.
- Coût d'accès aux n-uplets : soit N_t le nombre de n-uplets de la table de faits référencés par la requête Q. Le coût de lecture (*CL*) de ces n-uplets est donné par la formule suivante : $CL = \|F\| \left(1 - e^{\frac{N_t}{\|F\|}}\right)$

Où $||F||$ désigne le nombre de pages nécessaires pour stocker la table des faits F.

Nous présentons dans ce qui suit les principaux travaux sur la sélection d'index de jointure Binaire dans les entrepôts de données.

I.1. 5 Travaux de Aouiche et al:

Le travail d'Aouiche et al. [6] est parmi les rares travaux qui traitent le problème de sélection des IJB dans le contexte des entrepôts de données modélisés par un schéma en étoile. L'approche proposée se base sur une technique de datamining (recherche des motifs fréquents) pour élaguer l'espace de recherche des index de jointure. Un algorithme glouton est utilisé pour la sélection d'une configuration d'index. La figure I.2 représente les principales étapes de l'approche proposée

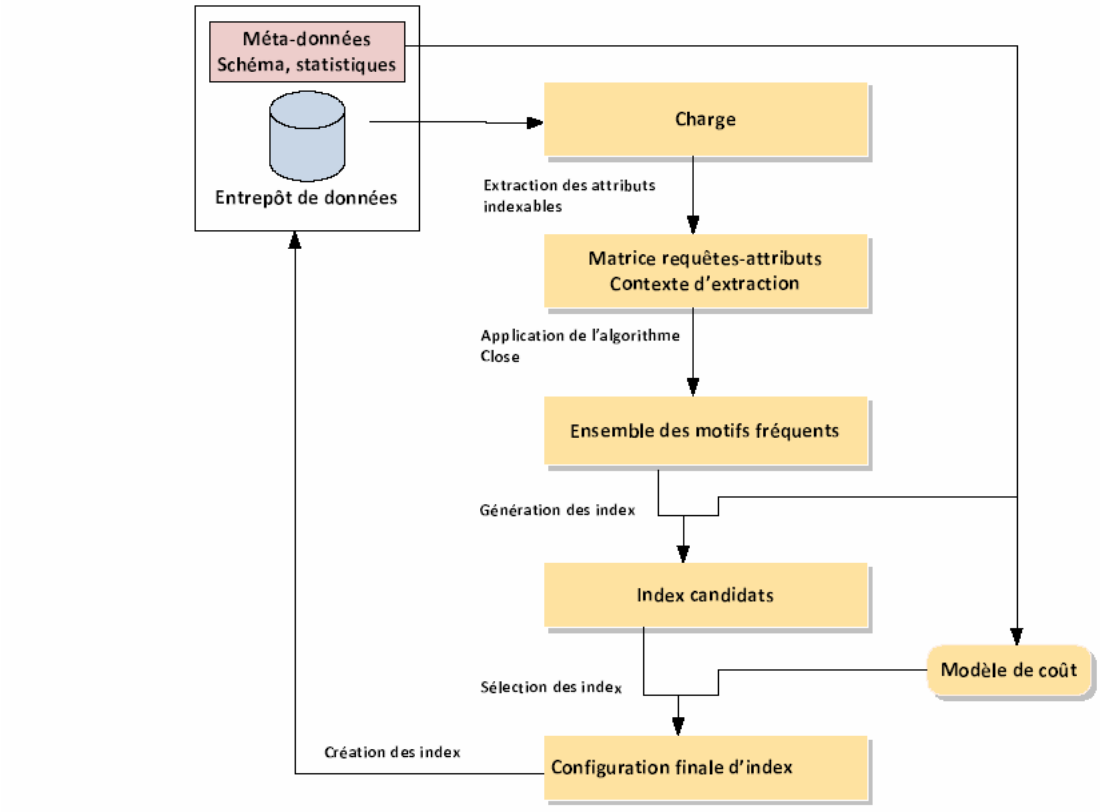


Figure.I.2 : *Architecture de fonctionnement de l'approche de Aouiche et al* [6]

L'approche de sélection proposée par les auteurs comporte six étapes :

1) ***Extraction de la charge de requêtes :*** la charge de requêtes est extraite à partir du journal des transactions sauvegardé et maintenu automatiquement par le SGBD.

2) ***Analyse de la charge :*** la charge de requête obtenue est analysée afin d'extraire l'ensemble des attributs indexables. Ces attributs sont ceux qui font l'objet de prédicats de sélection dans les clauses WHERE des requêtes.
3) ***Construction d'un contexte de recherche des motifs fréquents :*** le contexte d'extraction est représenté par une matrice requête-attributs construite à partir des requêtes et des attributs indexables. Les lignes dans cette matrice représentent les requêtes et les colonnes les attributs indexables utilisés par chaque requête. La j`eme case d'une ligne i dans cette matrice est mise à 1 si la requête Qi utilise l'attribut indexable Aj , elle est mise à 0 sinon.
4) ***Application de l'algorithme CLOSE sur ce contexte :*** l'algorithme CLOSE est appliqué sur le contexte d'extraction afin d'extraire l'ensemble des motifs fréquents.
5) ***Construction de l'ensemble des index candidats :*** l'ensemble des index candidats est construit à partir des motifs fréquents fermés résultant de l'application de l'algorithme CLOSE.
6) ***Construction de la configuration d'index finale :*** A partir de l'ensemble d'index générés dans l'étape précédente, un algorithme glouton est appliqué pour sélectionner une configuration d'index finale.

I.1.6 Travaux de Bellatreche et al :

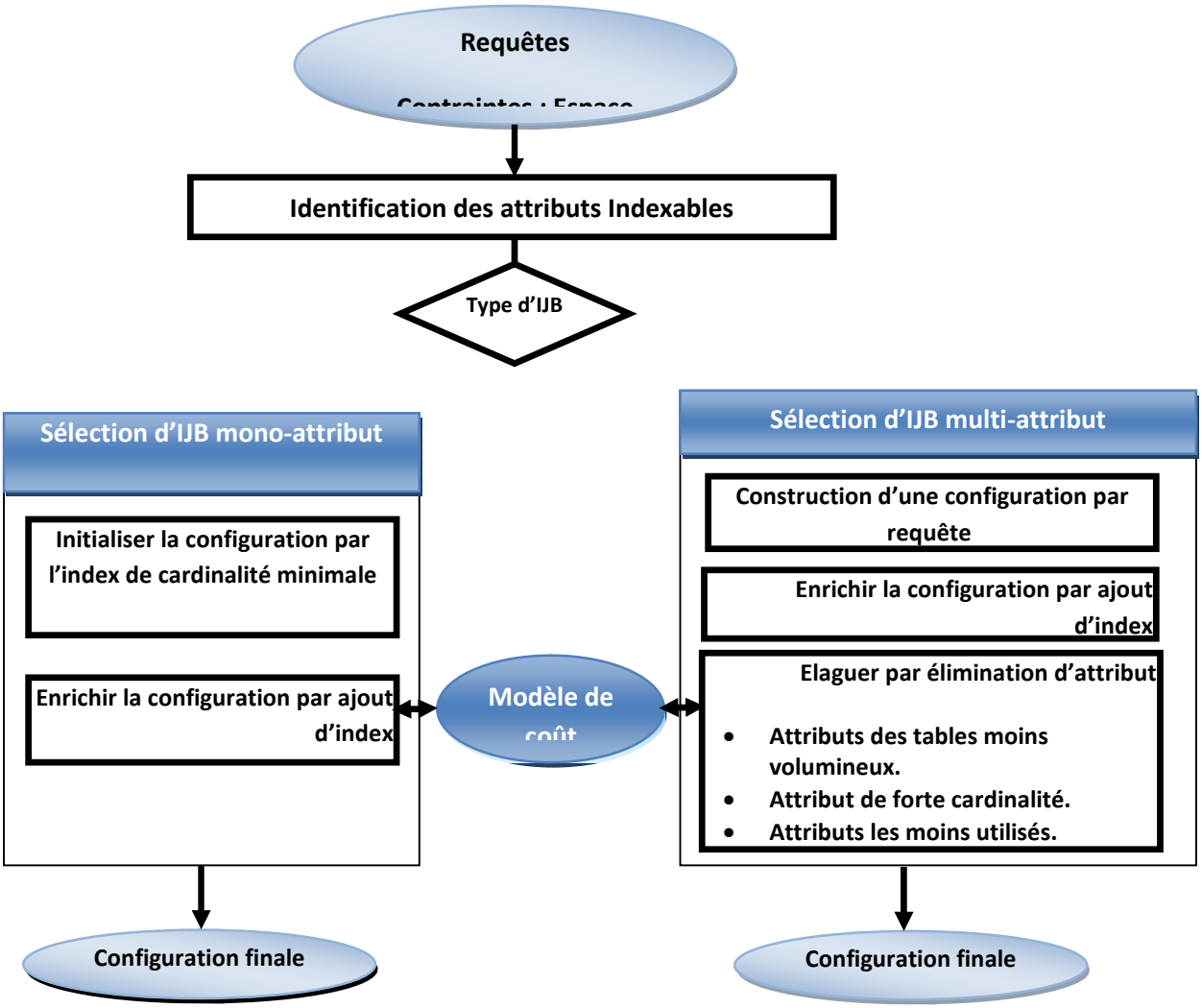
Les travaux de Bellatreche et al. [12, 13] présentent une amélioration des travaux de Aouiche et al [6]. Ces derniers considèrent seulement les fréquences d'accès des attributs comme critère de génération des motifs fréquents fermés.

Bellatreche et al. ont montré à travers un exemple que la fréquence d'accès seule ne permet pas de sélectionner un ensemble d'index efficace. En effet, les IJB sont créés pour optimiser des jointures entre la table des faits et les tables de dimension. En utilisant l'approche de Aouiche et al[6]., l'algorithme peut éliminer des index sur des attributs non fréquemment utilisés mais qui appartiennent à des tables de dimension volumineuses, ce qui ne permet pas d'optimiser une opération de jointure. Pour pallier ce problème, Bellatreche et al. proposent d'inclure d'autres paramètres dans la génération des motifs fréquents comme la taille des tables de dimension, la taille de la page système, etc.

Les auteurs proposent deux algorithmes DynaClose et DynaCharm qui sont utilisés dans la phase d'élagage de l'espace de recherche des index. Une fois les motifs fréquents générés, une étape de purification permet d'éliminer les motifs qui ne peuvent pas générer un index de jointure. Par exemple, un motif fréquent ne contenant aucun attribut non clé des tables de dimension sera supprimé. La purification permet de générer un ensemble d'attributs indexables candidats. Cet ensemble est défini par l'union des attributs non clés appartenant aux motifs fréquents générés. A partir de l'ensemble d'attributs candidats, un algorithme glouton proposé par les auteurs permet de sélectionner une configuration d'index finale sous une contrainte d'espace de stockage. L'algorithme glouton commence par l'index défini sur l'attribut ayant la cardinalité minimum, ajouter ensuite d'autres index itérativement jusqu'à ce que l'espace de stockage soit consommé ou tous les index sélectionnés.

I.1.7 Travaux de Boukhalfa et al:

Boukhalfa et al [5] proposent une approche de sélection d'une configuration d'IJB visant à réduire le coût d'exécution d'une charge de requêtes en prenant compte une contrainte de stockage. Ils proposent deux algorithmes de sélection, le premier sert à sélectionner une configuration d'index mono-attributs et le deuxième un ensemble d'index multi-attributs. L'architecture générale de leurs approches est représentée dans la figure I.3. Notons que les deux algorithmes utilisent un modèle de coût pour évaluer la qualité des solutions obtenues. Ce modèle est proposé dans Aouiche et al [6] et permet d'estimer la taille des IJB ainsi que le coût d'exécution des requêtes en présence d'une configuration d'IJB.



FigureI.3: *Architecture de fonctionnement de l'approche de Boukhalfa et al* [5]

Sélection d'une configuration d'index mono-attributs

La sélection d'une configuration d'index mono-attributs se déroule en trois étapes :

(1) l'identification des attributs indexables.

(2) l'initialisation de la configuration.

(3) l'enrichissement de la configuration actuelle par l'ajout de nouveaux index.

Dans la première étape, l'ensemble des requêtes est analysé afin d'extraire les attributs indexables. Ces attributs sont les attributs sur lesquels un prédicat de sélection est défini dans la charge de requêtes. Les attributs indexables candidats sont choisis parmi les attributs indexables de faible et de moyenne cardinalité. L'algorithme proposé démarre par une configuration initiale composée d'un index

mono attribut défini sur l'attribut ayant une cardinalité minimale noté BJImin (étape 2). La configuration initiale est améliorée itérativement par l'ajout d'index définis sur d'autres attributs non encore indexés (étape 3). L'algorithme s'arrête si les deux conditions sont satisfaites : aucune amélioration n'est possible et l'espace de stockage est consommé.

Sélection d'une configuration d'index multi-attributs

Un IJB multi-attributs peut être défini sur plusieurs attributs $\{A_1, A_2, \cdots, A_n\}$ où chaque attribut A_j peut appartenir à n'importe quelle table de dimension. La sélection des IJB commence par une configuration initiale qui permet d'optimiser toutes les requêtes, sans tenir compte de son coût de stockage. Par conséquent, ils procèdent à une réduction itérative de sa taille. Quatre étapes caractérisent leurs approche de sélection : (1) l'identification des attributs indexables, (2) la construction d'une configuration par requête, (3) la construction d'une configuration initiale et (4) la construction d'une configuration finale.

1) ***Identification des attributs indexables.***
2) ***Construction d'une configuration par requête.***
3) ***Construction d'une configuration initiale*** **:** dans cette étape ils construisent une configuration initiale à partir de l'ensemble des index générés lors de l'étape précédente. Cette configuration initiale est générée en effectuant l'union des index obtenus afin que chaque requête puisse être optimisée via l'IJB contenant tous les attributs indexables qu'elle utilise.
4) ***Construction d'une configuration finale*** **:** La configuration initiale ne prend pas en compte la contrainte d'espace qui peut être violée à cause du nombre important d'IJB sélectionnés pour satisfaire toutes les requêtes. Durant cette étape la configuration initiale est améliorée afin de réduire le coût d'exécution des requêtes et de respecter la contrainte de stockage.

La réduction de la taille d'un IJB peut être faite en éliminant certains attributs rentrant dans sa définition. Boukhalfa et al [5] ont considéré les quatre stratégies d'élimination suivantes :

- Elimination des attributs de forte cardinalité (AFC)
- Elimination des attributs appartenant aux tables moins volumineuses (TMV)
- Elimination des attributs les moins utilisés (AMU)
- Elimination des attributs apportant moins de réduction de coût(MC)

I.1.8 Travaux de Ziani et al:

Les travaux de Ziani et al. [15] présentent une amélioration des travaux de Bellatrache et al [12,13]. Ces derniers proposent un algorithme (Dyna*Close*) de recherche des motifs fréquents fermés qui s'appuie sur une fonction permettant de pénaliser des tables de petites tailles. Vu le nombre important des motifs générés par la technique des motifs fréquents fermés Ziani et al [15] proposent d'exploiter les motifs fréquent maximaux. Ces derniers sont moins nombreux que les motifs fréquents fermés ce qui va réduire l'espace de recherche des index candidats. Pour cela ils proposent l'utilisation de l'algorithme ***FPmax*** pour la recherche des motifs fréquents maximaux.

Leur approche est constituée des étapes suivantes :

1) sélection d'une charge de requêtes.
2) structuration des attributs indéxables contenus dans la charge sous forme d'une base transactionnelle ou les requêtes représentent les transactions et les attributs représentent les motifs. Ceci représente leur contexte d'extraction des motifs fréquents maximaux.
3) Génération des index candidats par l'algorithme ***FPmax*** implémenté en java.

Le nombre limité d'algorithmes de sélection des IBJ existants, l'importance des algorithmes génétiques et leurs performances dans les problèmes d'optimisation nous ont motivés pour en proposer de nouveau algorithme.

Dans la suite de ce mémoire on propose une adaptation des algorithmes génétiques au problème de sélection des IJB.

Chapitre II :

Les algorithmes génétiques

Introduction

Un algorithme génétique est, au premier lieu, un algorithme ; ce qui veut dire une méthode générale qui permet de résoudre un problème spécifique, en suivant un procédé bien précis ou un ensemble d'instructions, que l'on peut appliquer de la même façon, quelles que soient les données du problème.

Un nombre de biologistes ont utilisé les ordinateurs pour réaliser des simulations de systèmes génétiques dans le but de comprendre les systèmes naturels. Ces simulations ont mené à des résultats très importants.

Plusieurs expériences de ce genre et dans plusieurs domaines en vue de comprendre les phénomènes naturels ont vu le jour, mais sans soupçonner que l'algorithme d'exploration employé par la nature pourrait être utile pour les systèmes artificiels.

Alors, le vocabulaire employé dans les algorithmes génétiques est directement calqué sur celui de la théorie de l'évolution et de la génétique. C'est au début des années 1960 que john holland de l'université du Michigan, avec ses collègues et ses étudiants, a commencé à s'intéresser à ce qui allait devenir les algorithmes génétiques. Ses travaux ont trouvé un premier aboutissement en 1975 avec la publication de ***Adaptation in Natural an Artificial System***.

John holland a développé des idées et des théories qui ont conduit à la création des nouveaux principes d'un algorithme, fondés sur les mécanismes de la sélection naturelle et de la génétique.

Simultanément, Holland a réalisé le rôle fondamental la sélection naturelle – la survie naturelle du plus adapté - .

Donc : **Qu'est ce qu'un algorithme génétique ?** Un algorithme génétique est une méthode de recherche itératif, dont le but est d'optimiser une fonction de coût (ou *fitness*) [18]. Il utilise à la fois les principes de la survie des structures les mieux adaptées et les échanges d'informations pseudo aléatoires, pour former un algorithme d'exploration qui possède certaines caractéristiques de l'exploration humaine [15].

II.1 Les concepts importants des algorithmes génétiques

Par analogie à la science les algorithmes génétiques ont pris les concepts importants de cette dernière. Ces algorithmes ont pris :

- **Individu** : entité représentée par son code génétique.
- **Population** : groupe d'individus.
- **Evaluation** : mesure de l'aptitude d'un individu à son environnement.
- **Sélection** : choix des individus pour la reproduction.
- **Reproduction** : croisement (crossover) + mutation.
- **Croisement (crossover)** : création d'individu à partir de deux individus parents.
- **Mutation** : modification (aléatoire) du code génétique d'un individu.
- **Fitness (coût)** : fonction à optimiser (fonction d'adaptation).
- **Les chromosomes (individus)** : sont les éléments à partir des quels sont élaborés les solutions (mutation et croisement génétiques).
- **La population** (génération) : est l'ensemble des chromosomes

II.2 Applications

Les domaines d'applications sont multiples :

- optimisation de fonctions numériques difficiles (discontinues…)
- traitement d'image (alignement de photos satellites, reconnaissance de suspects…)
- optimisation d'emplois du temps.
- optimisation de design.
- contrôle de systèmes industriels.
- apprentissage des réseaux de neurones.

II.3 Principe

Il s'agit de simuler l'évolution d'une population d'individus divers (généralement tirés aléatoirement au départ) à laquelle on applique différents opérateurs (recombinaisons, mutation) que l'on soumet à une sélection, à chaque génération .Si la sélection s'opère à partir de la fonction d'adaptation, alors la population tend à s'améliorer (**figure II.1**) .

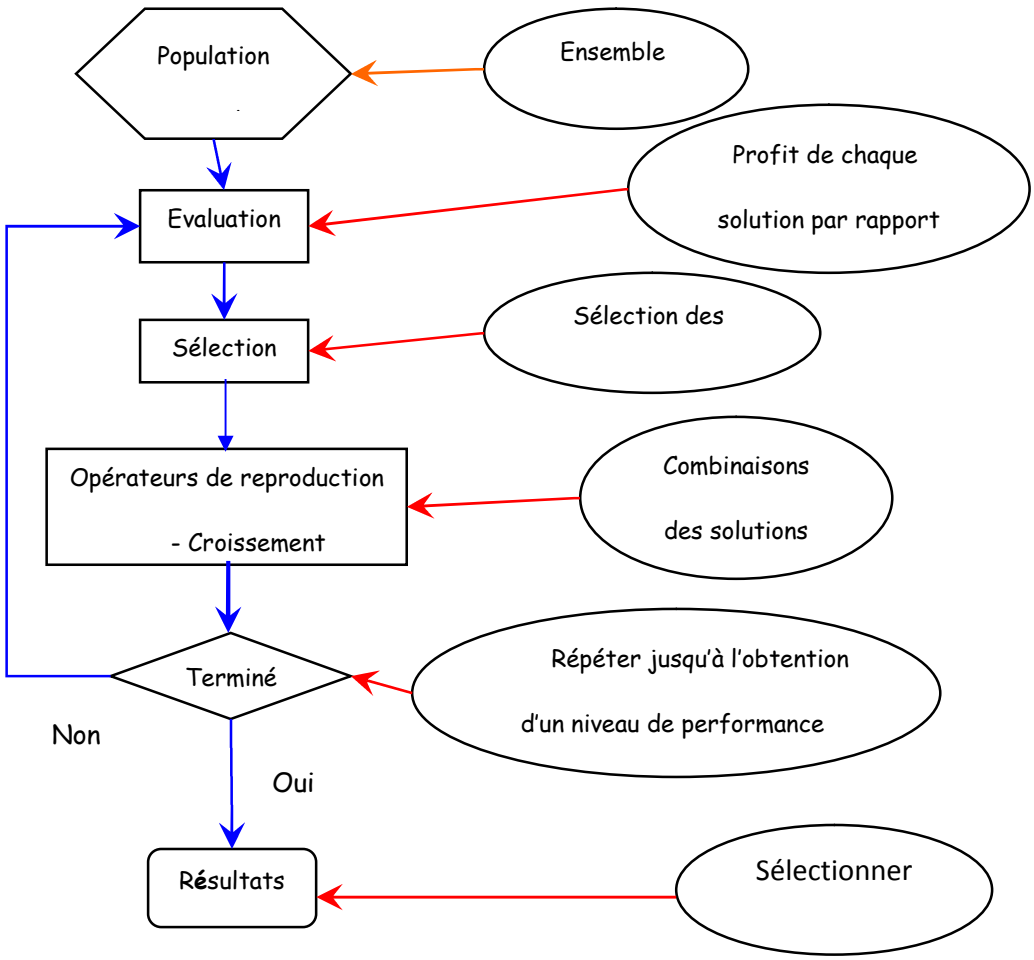


Figure II.1 : *Organigramme d'un algorithme génétique* [20]

Un algorithme génétique suit les étapes suivantes :

1. Générer aléatoirement une population de *n* chromosomes *x*;
2. Évaluer l'adaptabilité *f(x)* de chaque chromosome;
3. Créer une nouvelle population en :

    3.1 Sélectionner 2 parents chromosomes

    3.2 Croiser les deux parents avec une certaine probabilité T_c pour obtenir 2 enfants

    3.3 Sélectionner un chromosome, le muter avec une certaine probabilité T_m

    3.4 Placer les nouveaux chromosomes dans la population
4. Composer la nouvelle population
5. Si la nouvelle population n'est pas satisfaisante refaire les étapes à partir de 2.

II.4 Mécanisme de déroulement d'un algorithme génétique

Les mécanismes d'un algorithme génétique de base sont très simples, et ne mettent en jeu rien de plus compliqué que des copies de chaînes et des échanges de morceaux de chaînes.

Ainsi, un algorithme génétique commence avec une population de chaînes (individus) et génère par la suite des populations de chaînes successives. Et pour effectuer une recherche performante de structures de plus en plus intéressantes, il n'a besoin que des valeurs de la fonction à optimiser associées à chaque chaîne.

Il est maintenant nécessaire de définir un ensemble d'opérations simples qui permettent de produire, à partir d'une population initiale de la génération successive de plus en plus performante.

La génération d'une nouvelle population à partir de la précédente s'effectue suivant les étapes suivantes :

1. **Génération de la population initiale**

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Alors, il est préférable de démarrer avec une population initiale répartie de façon équitable sur l'espace de recherche ou avec des solutions approximatives assez bonnes qui ont été expérimentées auparavant.

2. **Evaluation**

A chaque itération ou génération, une nouvelle population est créée avec le même nombre d'individus, cette nouvelle génération contient généralement des individus plus adaptés à l'environnement tel qu'il est représenté par la fonction d'adaptation. Seuls les individus qui codent les meilleures solutions seront retenus. [17]

3. **La sélection**

La sélection permet d'identifier statiquement les meilleurs individus d'une population et d'éliminer les mauvais .Il y' a plusieurs méthodes de sélection, les plus utilisées sont :

a. *Roulette de casino :* C'est la sélection naturelle la plus employée pour l'AG binaire. Chaque chromosome occupe un secteur de roulette dont l'angle est proportionnel à son indice de qualité. Un chromosome est considéré comme bon aura un indice de qualité élevé, un large secteur de roulette et alors il aura plus de chance d'être sélectionné.

b. *" N/2 –élitisme " :* Les individus sont triés selon leur fonction d'adaptation, seule la moitié supérieure de la population correspondant aux meilleurs composants est sélectionnée. [16]

c. *"par tournoi " :* Choisir aléatoirement deux individus et on compare leur fonction d'adaptation (combattre) et on accepte le plus adapté pour accéder à la génération intermédiaire, et on répète cette opération jusqu'à remplir la génération intermédiaire (N/2 composants). Les individus qui gagnent à chaque fois on peut les copier plusieurs fois ce qui favorisera la pérennité de leurs gènes.

4. Opérateur de croisement

Le phénomène de croissement est une propriété naturelle de l 'ADN. C'est par analogie qu'ont été conçus les opérateurs de croissement dans les algorithmes génétiques.

Le croissement (crossover) est donc la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

Le croissement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes (individus). Pour effectuer des croissements sur des chromosomes constitués de *M* gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous chaînes terminales de chacun des deux chromosomes (*P1*, *P2*), ce qui produit deux enfants (*F1*, *F2*) (**figure II.2).**

.

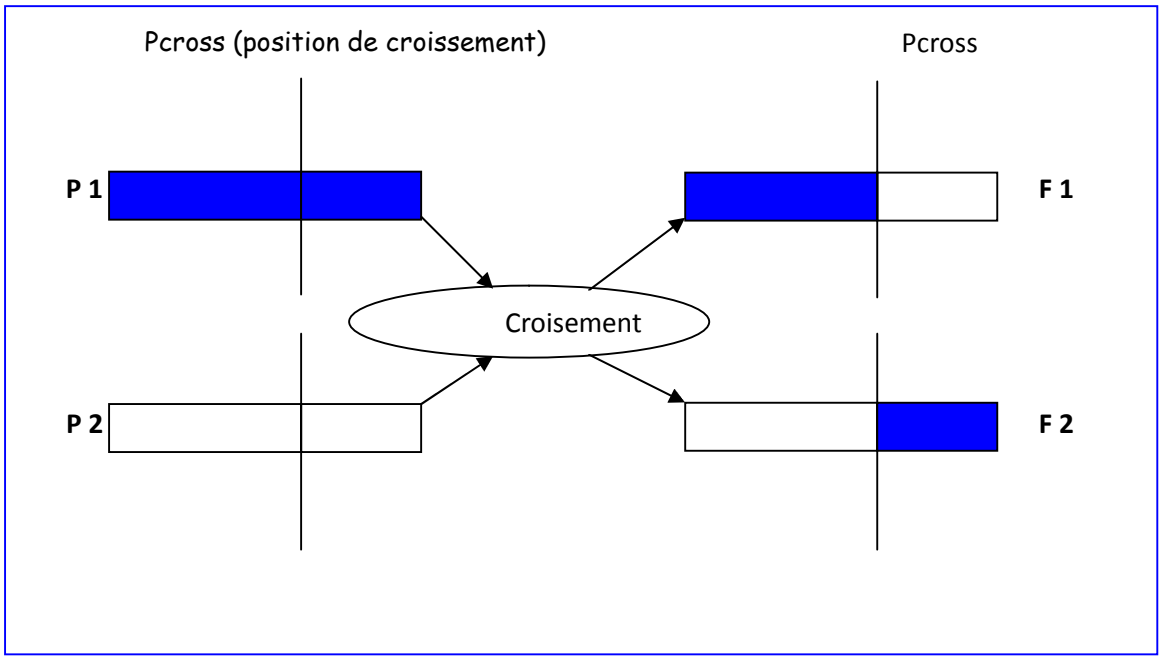


Figure II.2 : Principe de croisement

5. Opérateur de mutation

L'opérateur de mutation consiste, généralement, à tirer aléatoirement un gène dans le chromosome et le remplacer par une valeur aléatoire. Alors, la mutation est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un caractère

de la chaîne (**FigureII.3**). Son rôle est de protéger le système de la perte de la matière génétique potentiellement utile provenant de la reproduction et le croisement. [16]

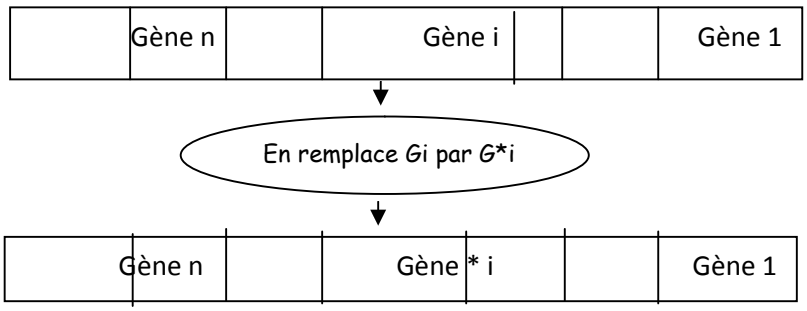


Figure II.3 : Principe de mutation

Chapitre III :

Adaptation des Algorithmes Génétiques au problème de sélection des IJB

Introduction

La conception physique des entrepôts de données relationnels est basée essentiellement sur la sélection d'un ensemble d'index afin de réduire le coût d'exécution des requêtes OLAP complexes. Ces entrepôts sont généralement modélisés par un schéma en étoile caractérisé par une table de faits volumineuse et un ensemble de tables de dimension liées à la table des faits par leurs clés étrangères. Les requêtes définies sur ce schéma (appelées requêtes de jointure en étoile) comportent plusieurs jointures entre la tables des faits et les tables de dimension ce qui rend leur coût d'exécution considérable. Les index de jointure binaires sont très adaptés pour réduire le coût d'exécution de ces jointures. Ils sont définis sur la table de faits en utilisant un ou plusieurs attributs de dimension. Sélectionner une configuration d'index pour réduire le coût d'exécution d'un ensemble de requêtes est reconnu comme un problème NP-Complet. Dans ce chapitre, nous présentons une approche d'adaptation des algorithmes génétiques au problème de sélection des index de jointure binaires.

III.1. Sélection des index de jointure binaires

Le problème de sélection d'une configuration d'IJB peut être formalisé comme suit :

- ✓ Un entrepôt de données modélisé par :
  - Un ensemble de tables de dimension $D = \{D_1, D_2, ..., D_d\}$
  - une table des faits F.
- ✓ une charge de requêtes $Q = \{Q_1, Q_2, ..., Q_n\}$
  - Où chaque requête Q_j a une fréquence d'accès f_j.
- ✓ une capacité de stockage S ;
- ✓ Trouvez une sélection des index CI :
  - Qui minimise le coût d'exécution de Q.
  - Taille(CI) ≤ S.

III.2. Adaptation des Algorithme génétiques au problème de sélection des index de jointure binaires

III.2.1 Génération de la population initiale

La population initiale est générée en effectuant l'union de certains nombre de configuration d'index, où la première configuration est générée avec l'union des index obtenus afin que chaque requête puisse être optimisée via l'IJB contenant tous les attributs indexables qu'elle utilise. Les autres configurations d'index sont générées aléatoirement.

Alors :

$$CI_1 = \bigcup_{i=1}^{n} IJB_i$$

Où IJB_i : est l'index créé pour la requête Q_i

$$CI_j = \bigcup_{i=1}^{n} \left(Generer\_aleatoirement\left(IJB_i \right) \right)$$

Où *CIj* est une configuration d'index générée aléatoirement sur les mêmes attributs indexables que la configuration CI_1.

III.2.2 Codage des individus

Le codage des individus est un codage binaire où chaque individu est composé des gènes (les index), comme il est montré dans le tableau Tab III.1, dans cet exemple l'individu (Configuration d'index) est composé de 6 index, chacun d'eux est indexé sur 4 attributs.

Index1	Index $_2$	Index $_3$	Index $_4$	Index $_5$	Index$_6$
1 0 1 0	1 0 0 0	0 0 1 0	1 1 1 0	1 0 0 1	0 0 1 1

Tab III.1 Codage des individus

III.2.3 Evaluation des individus (*Fitness*)

Evaluer et comparer la qualité des différentes configurations d'index générées nécessite l'utilisation d'un modèle de coût. Nous avons utilisé le modèle de coût d'exécution qui est expliqué dans le Chapitre I section I.1.4. Alors pour chaque configuration d'index en affectant une valeur d'adaptation (*fitness*) qui est le cout d'exécution, où a partir d'une requête Q et d'une configuration d'index CI, ce modèle permet d'estimer la taille des index de CI ainsi que le coût d'exécution de Q en termes de nombre d'entrées-sorties en présence de CI.

III.2.4 Sélection

L'opérateur de sélection permet de sélectionner les meilleurs individus dans la population pour participer dans la génération de la prochaine population. Nous avons utilisé la sélection par élitisme [23], où on sélectionne les parents qui ont le minimum de *Fitness*. Cette méthode de sélection permet de mettre en avant les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population.

III.2.5 Croisement

L'opérateur de croisement permet à deux individus d'échanger leurs gènes en vue de créer de nouveaux individus plus intéressants. Le croisement est appliqué sur deux individus pères choisis par l'opérateur de sélection. Dans notre approche on utilise un croisement en multi points selon le nombre des index constituant la configuration d'index. Dans l'implémentation réalisée on utilise un croisement en 4 points aléatoires. Cela permettra de donner la même chance à tous les attributs d'être choisis pour subir le croisement. Où on tire un nombre aléatoire pour fixer les emplacements de croisement. Aussi avec un taux de croisement variables.

Nous pouvons décrire le fonctionnement de l'opérateur de croisement par les étapes suivantes :

– Tirer un nombre aléatoire (entre 0 et 100)

– Si ce nombre est supérieur au taux de croisement Tc alors il n'y aura pas de croisement. Nous réinjectons les deux parents dans la population suivante.

– Si ce nombre est inférieur à Tc alors choisir aléatoirement quatre positions de croisement et échanger les gènes entre les deux individus.

–deux individus fils et les injecter dans la population suivante.

La figure III.2 montre un exemple de croisement en un seul point entre deux individus parents Le résultat du croisement de ces deux individus donne deux individus Enfant1 et Enfant2 représentés dans figure III.2.Ces deux enfants seront par la suite insérés dans la population suivante.

III.2.6 Mutation

L'opérateur de mutation permet de modifier occasionnellement des gènes d'un individu pour permettre d'explorer certaines zones dans la codification des individus où le croisement ne peut pas explorer. La mutation se fait avec un taux Tm généralement petit mais qui diffère d'une application à une autre. Si la mutation est trop fréquente, l'algorithme génétique est orienté vers une recherche aléatoire. L'opérateur de mutation que nous avons utilisé permet de modifier deux gènes de l'individu à muter. Nous pouvons décrire le fonctionnement de l'opérateur de mutation par les étapes suivantes :

– Choisir aléatoirement l'individu à muter.

– Tirer un nombre aléatoire (entre 0 et 100)

– Si ce nombre est supérieur à Tm alors injecter l'individu dans la population suivante sans mutation.

– Sinon, choisir un nombre aléatoire entre (1 et le nombre totales des gènes).

– affecter au gène sélectionné son complément (si le gène=0 alors gène :=1 sinon gène :=0)

- Injecter l'individu résultat dans la population suivante.

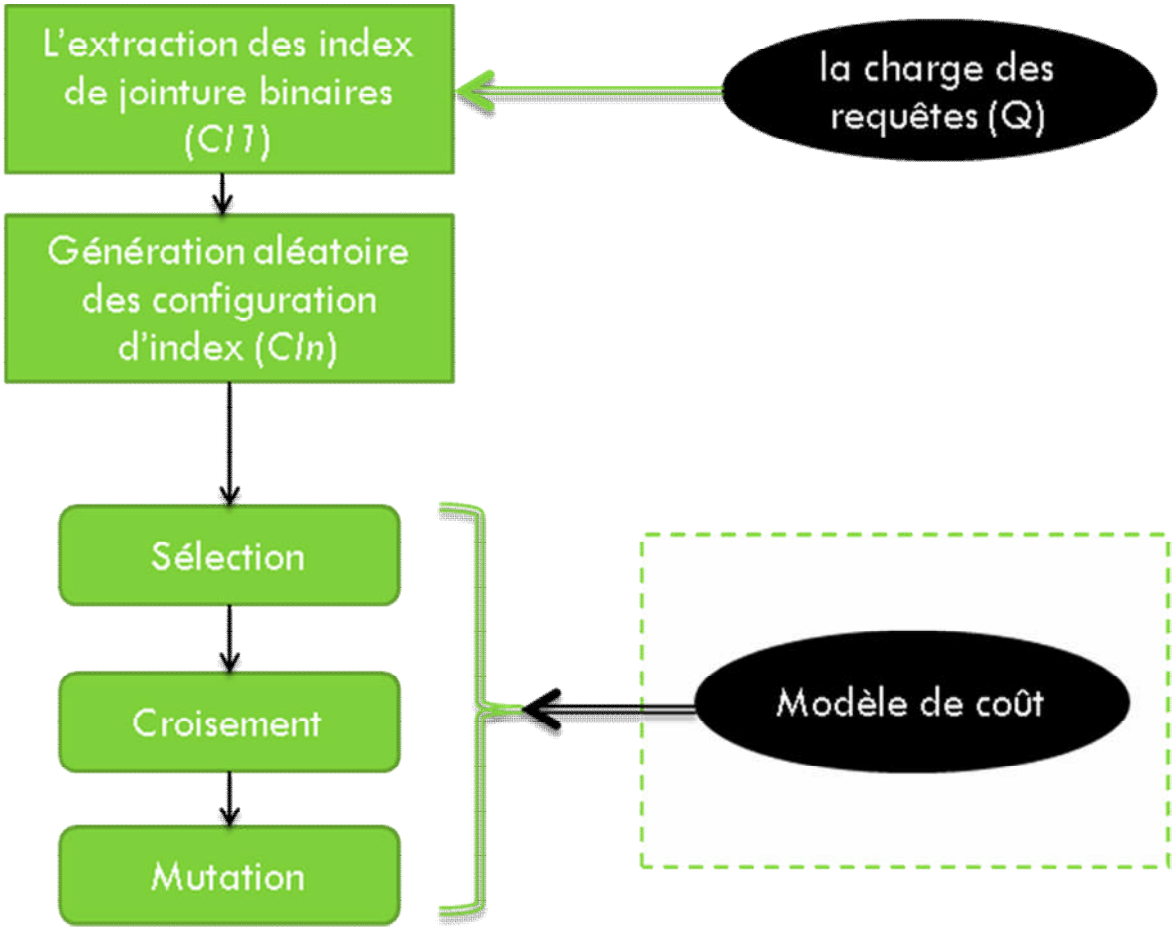


Figure III.1 Architecture de l'approche proposée

Algorithme génétique pour la sélection des IJB

Variable

-SelectUnIndex(Qj) fonction qui sélectionne un index multi-attributs pour chaque requête Qj

-SelectInd : fonction qui sélectionne le meilleur l'individu Ii.

-SelectMut : fonction qui sélectionne aléatoirement un individu.

-Injecter : Fonction qui remplace les individus croisés par leurs enfants ;

-Croiser : Fonction qui fait le croisement de deux Individus (parents).

-Muter : Fonction qui fait la mutation d'un Individus.

-Evaluer : Fonction qui calcul la Fitness de chaque configuration d'index.

-Select-Meill : Fonction qui sélectionne les individus qui ont la meilleure valeur de fitness.

Entrées : Charge Q, Taux croisement Tc, Taux de mutation Tm, nombre de configuration à généré M.

Sorties : Les Meilleurs Configuration d'index.

Debut

Pour j de 1 à n faire

IJB_j :=SelectUnIndex(Q_j) ;

Fin Pour

$CI_1 := \bigcup_{i=1}^{n} IJBi$;

Pour j :=2 to M faire

Créer_aleatoire CI_j ;

Fin Pour

PI:= $\bigcup_{i=1}^{n} CIi$

Pour S :=1 to N

Faire

Pour i :=1 to m faire Evaluer (CI_i) ; **Fait**

I_1 := SelectInd(PI) ;

I_2 := SelectInd(PI – I_1) ;

(E_1,E_2) :=Croiser (I_1, I_2) ;

I_3 :=SelectMut(PI) ;

E_3 :=Muter (I_3) ;

PF :=injecter (E_1,E_2,E_3 , PI) ;

PI :=PF

Fait

CI := select-Meill (PF) ;

Retourner CI ;

Fin

Algorithme1 - Algorithme génétique pour la sélection des IJB

III.3. Exemple d'application de l'approche proposée

Dans cet exemple on a une charge de requêtes qui est composé de 5 requêtes, et 3 attributs indexables (**A**, **B, C**). On aura donc pour chaque individu (Configuration d'index) 15 gènes. On prend 7 configurations d'index, la population générée est montré dans le tableau Tab III.2

III.3.1 Population initiale

	Index1	**Index $_2$**	**Index $_3$**	**Index $_4$**	**Index $_5$**
CI_1	0 0 1	0 1 0	0 1 1	1 0 1	1 1 1
CI_2	0 1 1	1 0 0	0 0 0	0 0 0	0 0 1
CI_3	0 0 1	0 1 0	0 0 0	1 0 0	0 0 0
CI_4	0 0 1	0 0 1	0 0 0	1 0 1	0 0 1
CI_5	0 0 0	0 0 0	1 0 0	0 0 1	0 1 1
CI_6	0 0 0	0 0 0	0 0 0	1 0 1	1 1 1
CI_7	1 0 1	0 0 1	0 0 0	0 1 0	0 0 1

Tab III.2 Population Initiale

III.3.2 Sélection, Croisement et Mutation

Les processus de sélection, croisement et mutation sont expliqués dans les figures III.2, figure III.3 où les individus CI_2, CI_4 sont sélectionnés, et ils ont subits un croisement en deux points au niveau de 4^{eme} gène et le 10^{eme} gène, avec cette opération on a eu 2 nouveaux individus NCI_2, NCI_4 .l'individu CI_1 a eu une mutation au niveau de 6^{eme} bit. À la fin de ces 2 opérations on a 3 nouveaux individus à les évaluer et les injecter dans notre nouvelle population (Tab III.3).

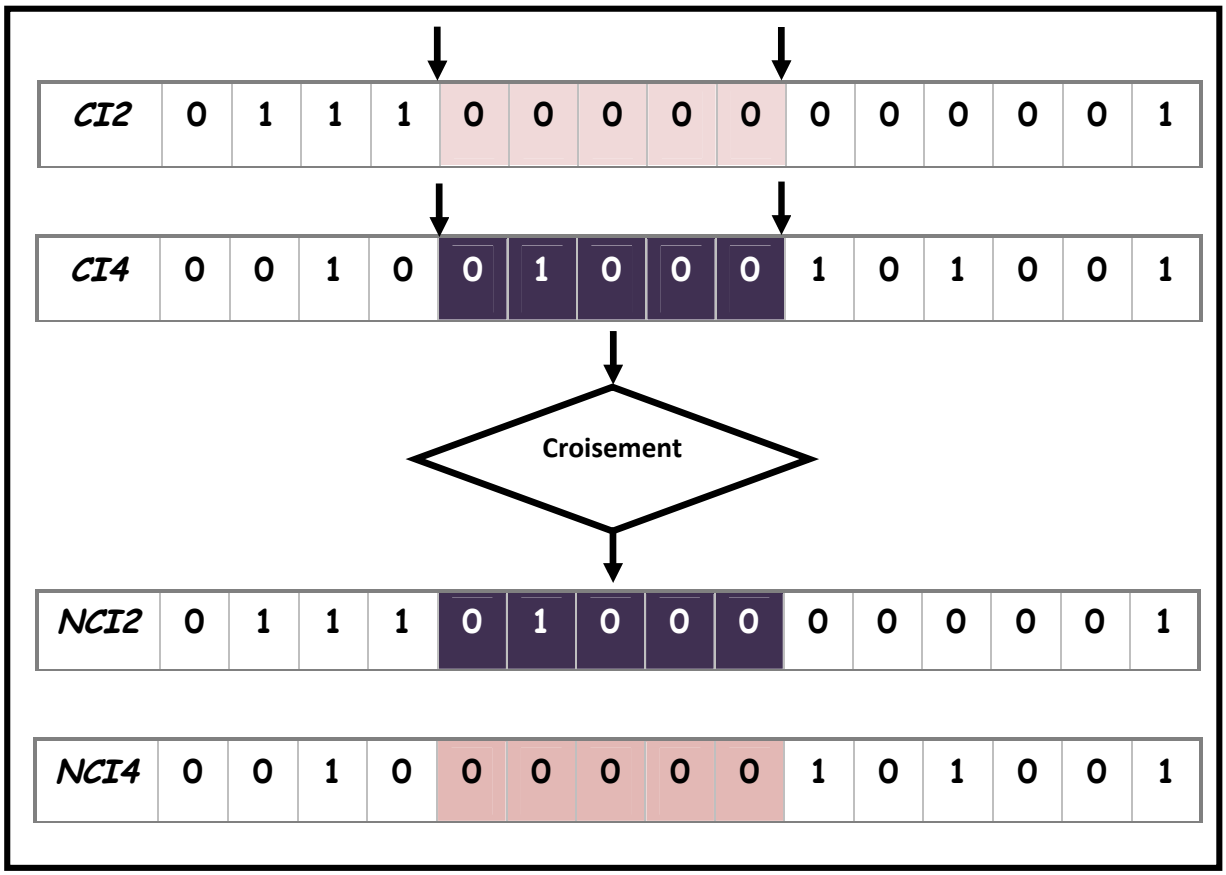


Figure III.2 Croisement

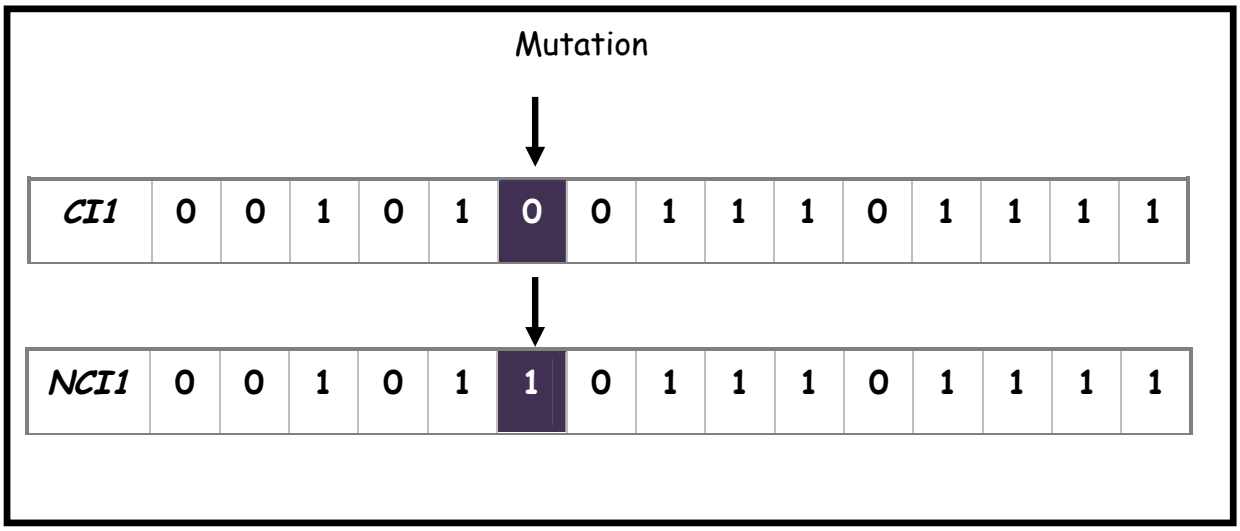


Figure III.3 Mutation

	Index1	Index $_2$	Index $_3$	Index $_4$	Index $_5$
CI_1	0 0 1	0 1 1	0 1 1	1 0 1	1 1 1
CI_2	0 1 1	1 0 1	0 0 0	0 0 0	0 0 1
CI_3	0 0 1	0 1 0	0 0 0	1 0 0	0 0 0
CI_4	0 0 1	0 0 0	0 0 0	1 0 1	0 0 1
CI_5	0 0 0	0 0 0	1 0 0	0 0 1	0 1 1
CI_6	0 0 0	0 0 0	0 0 0	1 0 1	1 1 1
CI_7	1 0 1	0 0 1	0 0 0	0 1 0	0 0 1

Tab III.3 Population Finale

III.3.3 Résultat obtenu après plusieurs itérations

Après plusieurs itérations de l'algorithme implémenté on obtient une population finale qui est l'ensemble des configurations d'index résultants, on prenant les meilleurs résultats qui sont les configurations d'index portant le minimum de cout d'exécution.

Dans l'exemple de la table Tab III.4 on a 4 configurations d'index après l'élimination des index redondants et nul de chaque configuration on aura le résultat montré dans la table III.5

	Index1	Index 2	Index 3	Index 4	Index 5
CI_1	1 0 1	0 0 1	0 1 1	1 0 0	1 1 1
CI_2	0 1 1	1 0 1	1 0 1	0 0 0	0 0 1
CI_3	0 0 1	0 1 0	0 0 0	1 0 0	0 0 0
CI_4	1 1 1	0 0 0	0 0 0	1 0 1	0 1 1

Tab III.4 les meilleures configurations

	CI_1	CI_2	CI_3	CI_4
IJB_1	{A}	{C}	{A}	{A, C}
IJB_2	{C}	{A, C}	{B}	{B, C}
IJB_3	{A, C}	{B, C}	{C}	{A, B, C}
IJB_4	{B, C}	-	-	-
IJB_5	{A, B, C}	-	-	-

Tab III.5 Résultat Final

Chapitre IV :

Etude Expérimentale

IV.1 Implémentation

Nous avons implémenté notre approche de sélection d'IJB sous Delphi. Cette approche est composée des étapes suivantes :

1-l'extraction de la configuration d'index CI1 : à partir de la charge de requête Q, on construit une configuration d'index où pour chaque requête on crée l'IJB apriori. Alors si on par exemple une charge de requête de 30 requêtes on aura une configuration d'index de 30 index.

2-Génération aléatoire d'autres configurations d'index : afin d'enrichir notre population manipulé avec notre algorithme génétique, on génère aléatoirement de nouvelles configurations d'index utilisant le même nombre d'index que la 1ère configuration (CI1).

3-lancement des processus de Sélection, Croisement, et Mutation : dans cette étape on sélectionne les meilleures configurations d'index (ayant le minimum de cout) et les croiser, les muter aléatoirement.

4- sélection de la meilleure configuration : à la fin de notre algorithme on sélectionne la meilleure configuration ayant le minimum de cout. Pour l'évaluation de nos configurations d'index on a utilisé le modèle de cout qui a été expliqué dans le chapitre I section I.

Entrepôt de données :

Notre étude expérimentale est effectuée sur l'entrepôt de données issu du banc d'essais APB-1 qui est présenté dans la figure IV.1. Le schéma en étoile que nous avons dégagé à partir de ce banc d'essais est constitué d'une table de faits Actvars et

de quatre tables de dimension, Prodlevel, Custlevel, Timelevel et Chanlevel. La table IV.4 résume les caractéristiques de chaque table.

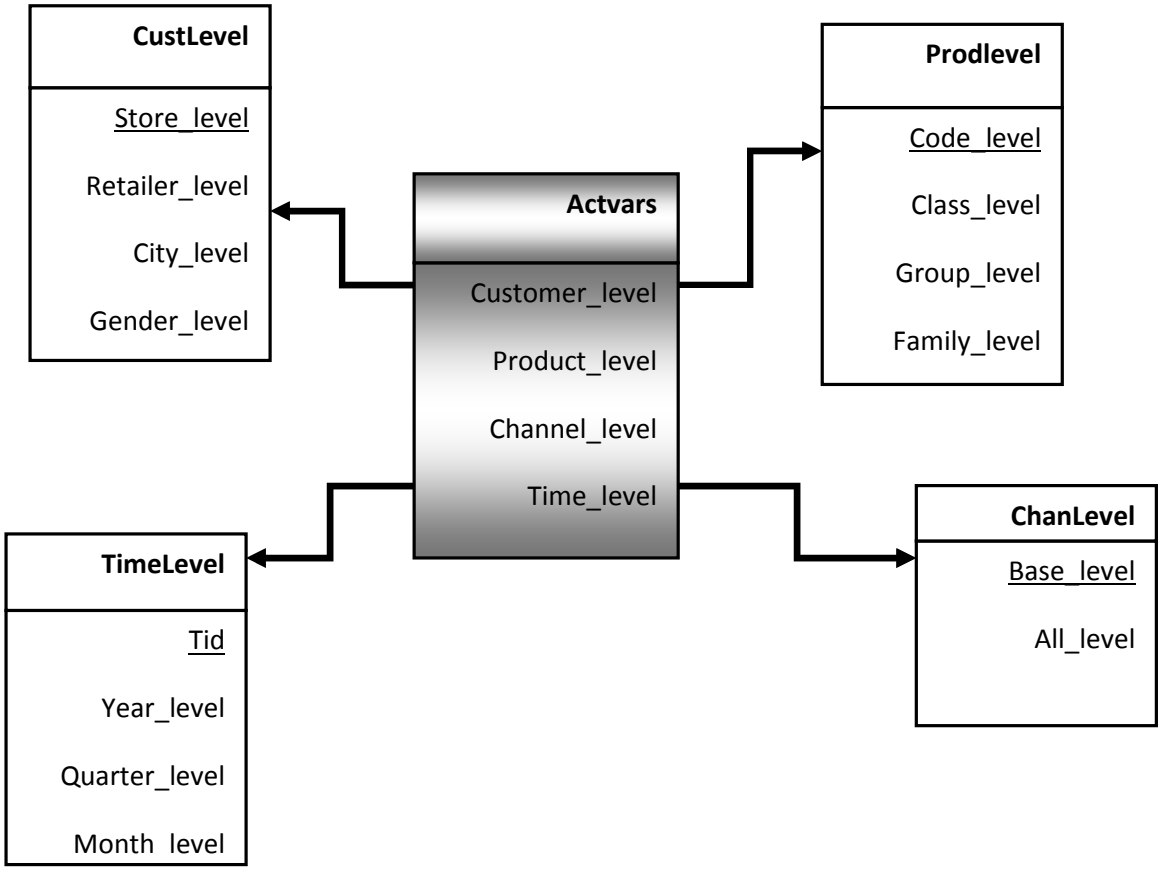


Figure IV.1 *Schéma en étoile de l'entrepôt expérimental*

.

Table	Nombre d'enregistrements	Taille d'un enregistrement
Actvars	24 786 000	74
Chanlevel	9	24
Custlevel	900	24
Prodlevel	9 000	72
Timelevel	24	36

Tab IV.1 Caractéristiques des tables de l'entrepôt

Charge de requêtes :

Sur l'entrepôt de données ci-dessus décrit, nous avons considéré 60 requêtes de recherche. L'ensemble des ces requêtes utilise 12 attributs de sélection. La table IV.2 donne pour chaque attribut sa cardinalité.

Code	Attribut	Cardinalité
A	ClassLevel	605
B	QuarterLevel	4
C	GroupLevel	300
D	FamilyLevel	75
E	LineLevel	15
F	DivisionLevel	4
G	YearLevel	2
H	MonthLevel	12
I	RetailerLevel	99
J	GenderLevel	2
K	AllLevel	5
L	CityLevel	4

Tab IV.2 Caractéristiques des attributs de sélection

IV.2 Expérimentation

Nous avons effectué plusieurs expériences en utilisant notre algorithme génétique. Ces expériences visent à comparer les performances (coût d'exécution (E/S) et taille).

Expérimentation1 : Variation de Taux de mutation et de Taux de croisement(Tm/Tc)

Nous avons fait plus d'une expérience en combinant différentes valeurs de taux de mutation et taux de croisement, les résultats obtenus sont montrés dans la figure IV.4, ces résultats montrent que le changement des taux de mutation et de taux de croisement n'as pas un grand effet sur le coût obtenue.

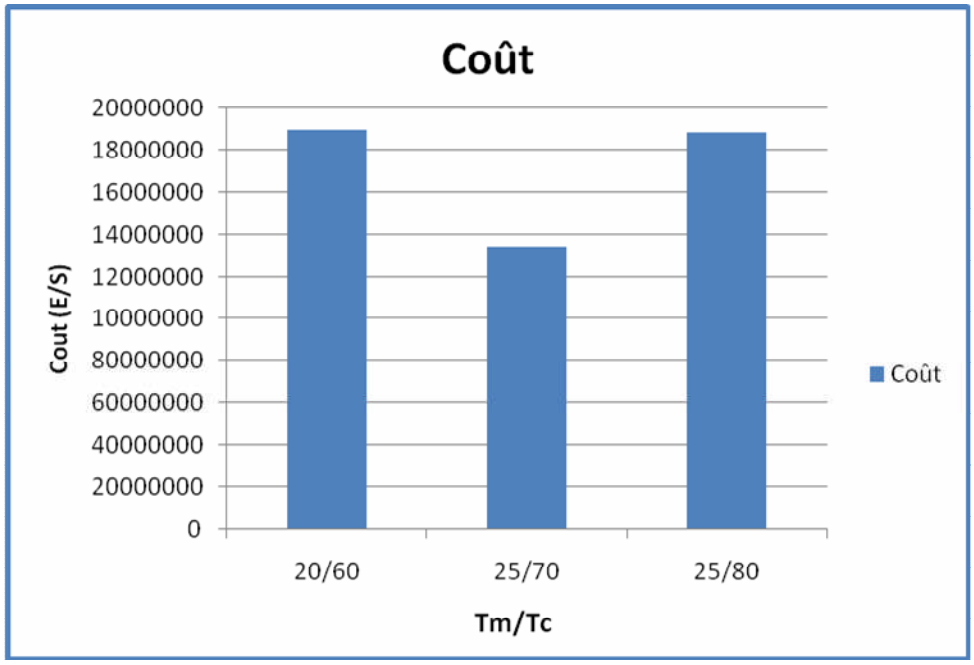


Figure IV.2 Effet de Tm et Tc sur le coût

Expérimentation2 : Comparaison de notre algorithme génétique avec d'autres techniques

Pour tester la performance de l'algorithme génétique implémenté nous avons comparé les résultats obtenus avec ce dernier avec les résultats obtenus avec la technique datamining de l'algorithme *FPmax* par Ziani et al [15].

L'étude expérimentale vise à montrer les points suivants :

- le nombre d'index générés avec chaque technique.
- La composition de la configuration d'index résultante de chaque technique.
- Espace(en Méga octet).
- Le cout d'exécution(E/S).

Nombre d'index :

Le tableau suivant montre le nombre des index générés pour les deux techniques comparés.

Algorithme *FPmax*	Algorithme génétique
14 index	18 index

Tab IV.3 Nombre des index pour chaque technique

Composition de la configuration d'index:

Le tableau en dessous montre les configurations d'index résultantes de chaque technique.

INDEX	**ALGORITHME GENETIQUE**	**ALGORITHME FPMAX**
IJB_1	**G, H**	**B**
IJB_2	**F, G, L**	**C**
IJB_3	**A, D, E**	**E**
IJB_4	**B, F, I, J**	**A, F**
IJB_5	**A, E, H, I**	**A, H**
IJB_6	**A, C, E, F, I**	**A, J, K**
IJB_7	**B, D, F, G, J**	**F, J, L**
IJB_8	**F, H, I, K, L**	**H, J, L**
IJB_9	**C, D, E, F, J**	**F, I, L**
IJB_{10}	**A, G, H, J, K**	**F, H, J, L**
IJB_{11}	**F, G, I, J, L**	**A, G, I, L**
IJB_{12}	**B, E, F, I, K, L**	**G, I, J, K**
IJB_{13}	**B, C, E, G, I, K**	**G, I, J, L**
IJB_{14}	**A, D, E, H, J, K, L**	**G, I, K, L**
IJB_{15}	**D, F, G, H, J, K, L**	
IJB_{16}	**A, B, D, F, H, J, L**	
IJB_{17}	**A, C, D, E, F, G, I**	
IJB_{18}	**B, D, E, F, G, H, J**	

Tab IV.4 Comparaison de configuration d'index des deux techniques

Espace et cout :

L'espace de chaque configuration d'index et le cout d'exécution de la charge des requêtes en utilisant les configurations d'index résultantes sont montrés dans les figures suivantes.

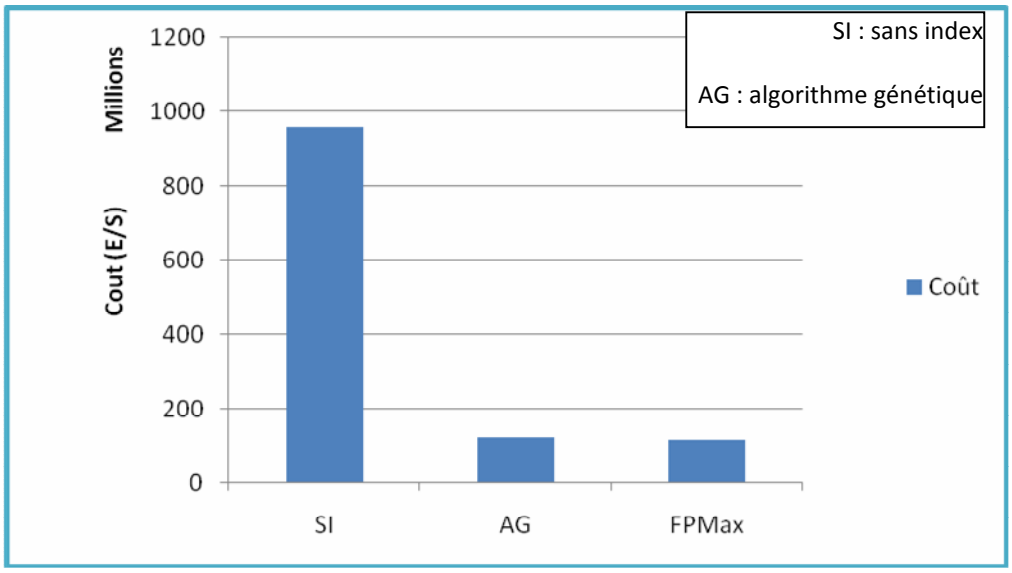


Figure IV.3 comparaison entre Algorithme génétique et l'algorithme FPmax (coût)

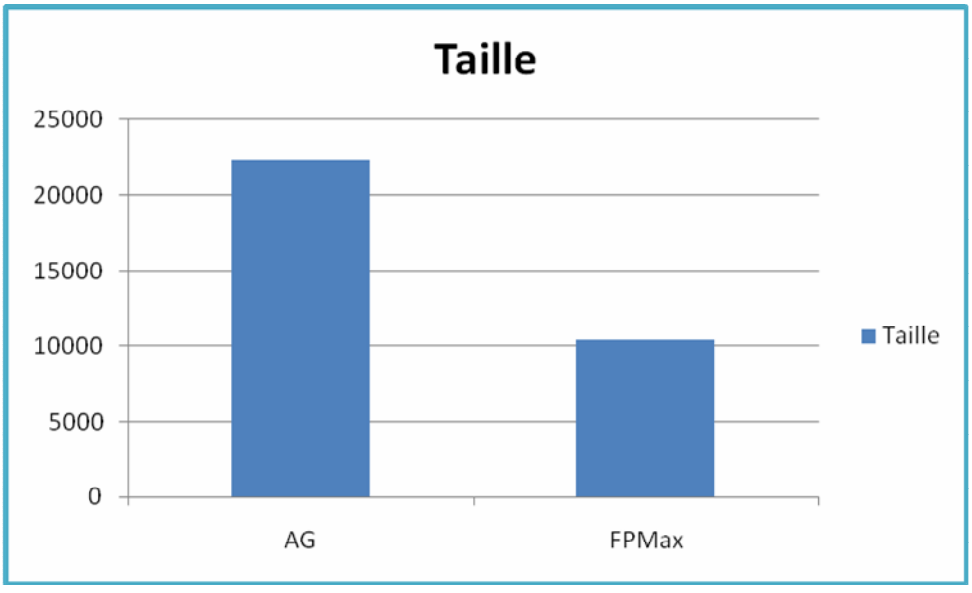


Figure IV.4 comparaison entre Algorithme génétique et l'algorithme FPmax (Espace)

Les résultats obtenus montrent que les deux approches génèrent des coûts d'exécution proches, cependant l'espace disque nécessaire pour le stockage des index est supérieur dans le cas de notre approche, ceci est surement dû au processus efficace d'élagage de l'approche datamining (algorithme *FPmax*), où il élimine les attributs non fréquents. Par contre avec notre algorithme génétique on utilise un croisement et mutation aléatoire ce qui implique à arriver de produire des index avec 7 attributs candidats ce qui implicite la taille élevée de la configuration de l'algorithme génétique.

Remarquons aussi qu'il y a aucun index commun aux deux configurations.

Conclusion et perspectives

Conclusion

Dans ce mémoire, nous nous sommes intéressés à l'optimisation des requêtes décisionnelles exécutées sur les entrepôts de données modélisées en étoile. Pour cela, nous avons proposé une approche de sélection des index de jointures binaires par un algorithme génétique. Nous avons implémenté notre approche en Pascal sous Delphi, et l'évaluée en utilisant le benchmark ABP-1. Les performances de l'approche proposée ont été comparées avec celle de l'approche utilisant la technique datamining (algorithme *FPmax*) de recherche des motifs fréquents max. les résultats montrent que les deux approches ont généré des coûts d'exécution proches, cependant l'espace disque nécessaire pour le stockage des index est supérieur dans le cas de notre approche, ceci est surement dû au processus efficace d'élagage de l'approche datamining.

Perspectives

Ce travail peut être amélioré si on arrive à reformuler le processus de croisement de notre algorithme génétique. Une autre amélioration peut être faite si on faisait une combinaison entre notre algorithme génétique et l'algorithme *FPmax* en prenant la première configuration de la population celle la résultante de l'algorithme *FPmax* au lieu de prendre toutes les index de jointure binaire associe à chaque requêtes de la charge des requêtes.

Références

[1] Kimball, R. et K. Strehlo (1995). Why decision support fails and how to fix it. *SIGMOD Record 24*(3), 92–97.

[2]O'Neil, P. et G. Graefe (1995). Multi-table joins through bitmapped join indices. *SIGMOD Record 24*(3), 8–11.

[3]O'Neil, P. et D. Quass (1997). Improved query performance with variant indexes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 38–49.

[4]Wu, K., A. Shoshani, et K. Stockinger (2010). Analyses of multi-level and multi-component compressed bitmap indexes. *ACM Transactions on Database Systems 35*(1).

[5] Kamel Boukhalfa, Ladjel Bellatreche, Benameur Ziani, Index de Jointure Binaires: Stratégies de Sélection & Étude de Performances, 2009.

[6]K. Aouiche, J. Darmont, O. Boussaid, and F. Bentayeb. Automatic Selection of Bitmap Join Indexes in Data Warehouses. 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK 05), August 2005.

[7]Aouiche, K., O. Boussaid, et F. Bentayeb (2005). Automatic Selection of Bitmap Join Indexes in Data Warehouses. pp. 64–73.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets. ICDT, pages 398–416, 1999.

[9] M. Zaki and C. Hsiao. Charm : An efficient algotithm for closed itemset mining.In proceeding of the 2nd SIAM International Conference on Data Mining (ICDM02),2002.

[10] H. J.Wang, J. and J. Pei. Closet+ : searching for the best strategies for mining frequent closed itemsets. in Proceedings of international conference on Knowledge discovery and datamining (ACM SIGKDD03), pages 236–245,2003.

[11] P. J. Han, J. and Y. Yin. Mining frequent patterns without candidate generation.In Proceedings of the ACM-SIGMOD 2000 Conference, Dallas, Texas, USA., pages 1–12, 2000.

[12] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. Selection and pruning algorithms for bitmap index selection problem using data mining. 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK '07), LNCS, pages 221–230,September 2007.

[13] L. Bellatreche, R. Missaoui, H. Necir, and H. Drias. A data mining approach for selecting bitmap join indices. Journal of Computing Science and Engineering, 2(1) :206–223,2008.

[14] M. Tomassini. A survey of genetic algorithms. Annual Reviews of Computational Physics, III(2) :87–118, 1995.

[15] B. ZIANI, Y. OUINTEN, « Sélection des index de jointure binaires dans les entrepôts de données: une approche basée sur la recherche des motifs fréquents maximaux » Technology (IIT09), 2009.

[16] http://khayyam.developpez.com/articles/algo/genetic/

[17]http://informatique.coursgratuits.net/methodes-numeriques/algorithmes-genetiques.php

[18] http://www.smsi.rnu.tn/html/manifes/mhosi/Mesghouni45.pdf

[19] http://www.webreview.dz/IMG/pdf/lasouaoui.pdf

[20] http://fr.wikipedia.org/wiki/Algorithme_génétique