REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : [Génie Informatique]

Par :

SABIH NOR EL HOUDA
HAMOUCH SARAH NOURELHOUDA

Sur le thème

Adaptation de l'algorithme Génétique au problème de transformation des modèles par l'exemple

Soutenu publiquement le 03 / 10 / 2021 à Tiaret devant le jury composé de :

Mr : BERBER EL MAHDI	Grade : MMA	Université de Tiaret	Président
Mr : SI ABD ELHADI AHMED	Grade : MMA	Université de Tiaret	Encadreur
Mr : TALBI OMAR	Grade : MMB	Université de Tiaret	Examinateur
Mr : OUARED ABDELKADER	Grade : MMB	Université de Tiaret	Co- encadreur

2020-2021

Remerciements

Premièrement nous remercions Dieu source de toute connaissance, et nous a donné la santé et la volonté d'entamer et de terminer ce mémoire.

Nous remercions le corps académique de département d'Informatique pour le cadre et la formation reçue tout au long de notre parcours universitaire Tous les enseignants, pour leurs enseignements de qualité et leurs conseils qui nous ont permis de poursuivre notre itinéraire académique jusqu'à présent.

Toute d'abord, ce travail ne serait pas aussi riche et n'aurait pas pu avoir le jour sans l'aide et l'encadrement de **Si abdelhadi Ahmed** et **Ouared Abdelkader**, pour ses multiples conseils et les efforts déployés et sa rigueur, sa disponibilité et ses qualités humaines m'ont profondément touché.

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Dédicaces

Arrivée à ce stade, n'est que le fruit du milieu familiale qui m'est propice, le fruit de
l'équilibre et du sacrifice de ***mes Parents*** *que Dieu les protègent.*
qu'il m'est agréable en ce moment de partager ce bonheur
avec eux. Ils ont été de tout temps, les plus proches,
et n'ont jamais ménagé leurs efforts,
leurs encouragements et leur soutien
avec abnégation et patience
- A Mon frère ***Mohamed elhadi Amine****, et ma petite sœur* ***Doaa****,*

Je tiens à témoigner toute ma gratitude.

- ***A mon fiancé*** *ne m'a procuré que confiance et stabilité. Tu as partagé avec moi les meilleurs moments de ma vie, aux moments les plus difficiles de ma vie, tu étais toujours à mes cotés, Je te remercie de ne m'avoir jamais déçu. Aucun mot ne pourrait exprimer ma gratitude..*
- *A mon Ami proche* ***Mohamed Ben Sadiaa*** *aux moments les plus difficiles de ma vie, tu étais toujours à coté de moi*

- A toute ma famille,
- A mes tantes et beaucoup plus ma tante ***Amina*** *et mes oncles,*

- *A mes cousines* ***Asma*** *et* ***Somia***
- *A ma très chère binôme* ***Sarah***

- A toutes mes amies, et mes collègues sur le ***DTN***

- *A tous ceux qui ont participé à ce travail de prés ou de loin.*

Norelhouda

Dédicace

Avec l'expression de ma reconnaissance, je dédie ce modeste travail a ce qui, quels que soient les termes embrassés, je n'arriverais jamais a leur exprimer mon amour sincère.

A l'homme, mon précieux offre du Allah, qui doit ma vie, ma réussite et tout mon respect : mon cher **père** Qu'ALLAH le tout puissant te préserve, t'accorde Santé, bonheur et te protège de tout mal.

A la femme qui a souffert sans me laisser souffrir, qui m'a jamais dit non à mes exigences et qui n'a épargné aucun effort pour me rendre heureuse mon adorable **mère** Qu'ALLAH te protégé et la santé, le bonheur et longue vie.

A mes chères sœur **Amel, Nour** et **Souhila** et mon frère **Mohamed** que j'aime tant Pour leur petit mot et leur soutien.

A **Soumia** une personne qui a une place spéciale dans mon cœur, une sœur, une amie Qui a été à mes côtés tous au long de cette année qui a partagé avec moi beaucoup de choses.

A ma tante **Sabrina** Que Dieu te protège et t'accorde santé et longue vie Je t'aime tellement

Sans oublier mon binôme **Nour elhouda** pour son soutien moral, sa patience et sa compréhension tout au long de ce projet.

Sarah

Résumé

La transformation des modèles consiste à transformer un modèle source (MS) conforme à un métamodel vers un modèle cible (MC) qui respecte les contraintes de métamodel cible pour différentes les exigences comme la simulation, la validation, le stockage etc . Nous pouvons citer l'exemple de transformation UML2 ER qui consiste à transformer un diagramme de classes UML vers un modèle ER relationnel pour un objectif de la persistance de ce modèle dans un SGBD relationnel (par ex. Oracle). Cette transformation doit respecter les besoins des utilisateurs et les contraintes imposées par des applications. Pour une seule instance d'un modèle source, plusieurs modèles cibles sont candidats à être sélectionnés. Devant l'explosion de l'espace de solution des MCs qui augmente en fonction des objets des modèles (classe, attributs, sous domaines des attributs etc.), le problème de transformation des modèles est assimilé comme un problème d'optimisation sous contrainte. Autrement dit, le problème de transformation est un problème NP complet. Notre objectif consiste à utiliser l'algorithme génétique pour résoudre le problème de transformation des modèles. Nous réalisons une implémentation sur un exemple d' UML2ER pour évaluer la faisabilité de notre approche.

Mots-clés : Transformation des modèles, Diagramme UML, Modèle relationnel, Algorithme Génétique

Abstract

Model transformation consists of transforming a source model (SM) conforms to a metamodel to a target model (TM) that respects the constraints of the target metamodel for different requirements like simulation, validation, storage etc. We can cite the example of UML2ER transformation which consists in transforming a UML class diagram to a relational ER model for the purpose of persisting this model in a relational DBMS (e.g. Oracle). This transformation must respect the needs of users and the constraints imposed by applications. For a single instance of a source model, multiple target models are candidates for the selection process. With the explosion of the solution space of MCs which increases according to the objects of the models (class, attributes, subdomains of attributes etc.), the model transformation problem is assimilated as a constraint optimization problem. In other words, the transformation problem is a complete NP problem. Our goal is to use the genetic algorithm to solve the model transformation problem. We carry out an implementation on an example of UML2ER to show the feasibility of our approach.

Keywords : Model Transformation by Example, Model Driven Engineering, Model Transformation, UML Diagram, Relational Model, Genetic Algorithm

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

L'automatisation des applications informatiques par le paradigme de l'ingénierie dirigée par les modèles (IDM) est un processus qui inclut la méta modélisation, la transformation et la génération de codes. La transformation des modèles transformés différente forme des modèles qui respectent les contraintes des métamodèles vers un modèle cible pour des besoins différents comme la simulation, le stockage des modèles, la validation formelle etc. Par exemple, transformer un diagramme de classes en modèle relationnel (c-à-d modèle SQL) pour un besoin de stockage des données de cette application.

Dans la pratique, le nombre d'objets (tables, classes, attributs, sous domaines des attributs) pour un modèle source peut être relativement grand, et chaque objet peut avoir un certain nombre d'attributs distincts qui sont inclus dans une classe séparée et dans les sous-classes (par exemple, les attributs du « produit » pourraient être sa « couleur », sa « taille », son « poids », etc.). Les requêtes d'utilisateur spécifient typiquement ces attributs, et la préparation d'une réponse à une requête peut impliquer une recherche étendue par un certain nombre de tables de dimension pour des valeurs d'attribut appropriées. En conséquence, elle peut être tout à fait longue pour répondre directement à des requêtes agrégat à partir de données stockées en fonction de ce modèle dans la base de données.

Afin de répondre au besoin de l'utilisateur, plusieurs transformations ont été proposées pour améliorer la performance de ces requêtes complexes d'analyse principalement : la fragmentation horizontale et verticale. Sélectionner un modèle cible optimal de chacune qui correspond un modèle source est un problème NP-Complet . Par conséquent, la sélection d'un modèle cible devient une tâche difficile. Pour satisfaire le maximum de requêtes ; il est indispensable d'opter pour une transformation, multiples actions sur le modèle source comme Split, Merge, la FH ce qui engendre une complexité énorme. Effectuer ces choix manuellement nécessite un temps de transformation non négligeable. Nous constatons un besoin crucial concernant le développement des outils de transformation des modèles pour aider les concepteurs et les développeurs.

Nous proposons une nouvelle démarche de transformation des modèles UML2ER dirigée par l'algorithme génétique (AG) pour aider les concepteurs dans leurs tâches de conception des modèles. Ce système nous montre l'aspect de converger vers une solution optimale en se basant sur les primitives génétiques : sélection croisement, mutation et évaluation.

Notre mémoire est organisé en quatre chapitres :

le premier chapitre , nous présentons une synthèse bibliographique sur les algorithmes d'optimisation : définition, concept général et classification des techniques.

Le second chapitre présente la notion de de l'ingénierie dirigée par les modèles (IDM) Nous y détaillerons aussi les principaux travaux proposés pour transformation des modèles.

Dans le troisième chapitre nous présenterons notre contribution. Nous modélisons notre système ainsi que les approches utilisées par une métaheuristique afin de résoudre les problèmes de trans-

formation des modèles . Nous décrivons la démarche de sélection des modèles cibles (MCs) et les concepts importants pour réaliser notre approche comme le modèle de coût ainsi que les algorithmes de sélection implémentés.

Le dernier chapitre sera consacré à la validation de notre approche de sélection des modèles cible .

Nous présenterons la description de l'outil que nous avons implémenté et nommé « Trans Gen ». C'est un outil assistant les concepteurs dans leur tâche de transformation des modèles. Cet outil doit fournir au concepteur un ensemble de recommandations concernant les actions sur les modèles source. L'étude de cas considère la transformation UML2 ER sur le banc d'essais TPC-H sous oracle 10g et sa charge de requêtes décisionnelles.

Chapitre 1

Algorithmes Génétiques

1.1 Introduction

Depuis le début de la vie l être humain cherche la perfection dans son travail pour ce là les scientifiques inventent les méthodes d'optimisation pour obtenir les bons résultats dans tous les domaines. Les problèmes d'optimisation occupent actuellement une place grandissante dans la communauté scientifique. Les méthodes d'optimisation recherchent une solution, ou un ensemble de solutions, dans l'espace de recherche, qui satisfont l'ensemble des contraintes et qui minimisent, ou maximisent, la fonction objective. Parmi ces méthodes, les méta- heuristiques sont des algorithmes génériques d'optimisation : leur but est de permettre la résolution d'une large gamme de problèmes différents (est un problème NP complet), sans nécessiter de changements profonds dans l'algorithme. Elles forment une famille d'algorithmes visant à résoudre des problèmes d'optimisation difficile.

1.2 L'optimisation combinatoire :

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation [38] et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire [26] . Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [13] . Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA). Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes) qui garantissent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

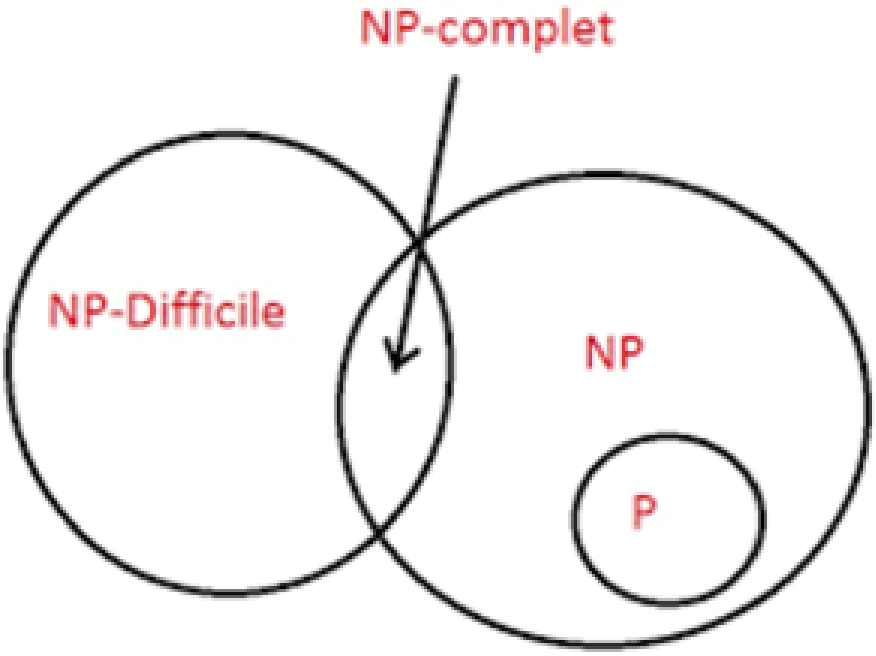

FIGURE 1.1 – complexité de problème

1.2.1 Méthode exacte vs méthodes approchées

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de séparation et évaluation progressive (SEP) ou les algorithmes avec retour arrière. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Malgré les progrès réalisés (notamment en matière de la programmation linéaire en nombres entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante. Les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation de grande taille si l'optimalité n'est pas primordiale. En effet, ces méthodes sont utilisées depuis longtemps par de nombreux praticiens. On peut citer les méthodes gloutonnes et l'amélioration itérative : par exemple, la méthode de Lin et Kernighan qui resta longtemps le champion des algorithmes pour le problème du voyageur de commerce [30].

1.2.2 Les méta-heuristiques

Depuis une dizaine d'années, des progrès importants ont été réalisés avec l'apparition d'une nouvelle génération de méthodes approchées puissantes et générales, souvent appelées métaheuristiques [16]. Une heuristique est une méthode qui cherche une bonne solution en un temps de réponse raisonnable sans garantir l'optimalité. Notons qu'une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème. Une méta-heuristique est définie de manière similaire, mais à un niveau d'abstraction plus élevé ,est constituée d'un ensemble de concepts fondamentaux (par exemple, la liste tabou et les mécanismes d'intensification et de diversification pour la métaheuristique tabou), qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation . Ainsi les métaheuristiques sont adaptables et applicables à une large classe de problèmes. Les métaheuristiques sont représentées essentiellement par les méthodes de voisinage comme le recuit simulé et la recherche tabou, et les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution. Grâce à ces métaheuristiques, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classiques de plus grande taille et pour de très nombreuses applications qu'il était impossible de traiter auparavant [26]

Méta-heuristiques pour la sélection des solutions optimale

Le problème de sélection d'une solution optimale dans un espace exponentiel est formalisé comme un problème d'optimisation combinatoire. Rappelons qu'un problème d'optimisation combinatoire est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution s X le nombre réel (ou entier) f(s). Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une solution s X optimisant la valeur de la fonction de coût f. Une telle solution s s'appelle une solution optimale ou un optimum global. Notons que dans notre cas, les ensembles S et X correspondent respectivement à l'ensemble de tous les solutions possibles de vérifiant la contrainte. La fonction objective correspond à la fonction de coût calculant le coût de chaque solution suivant la fonction objective [38].

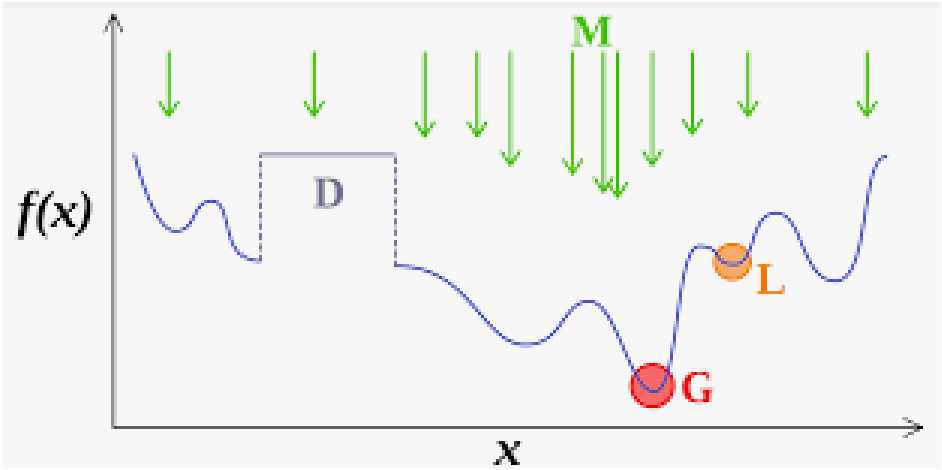

FIGURE 1.2 – Méta-heuristiques pour la sélection des solutions optimales

1.2.3 Classification des méta-heuristique

:

Nous pouvons classer les méta-heuristiques selon le nombre de solutions traitées simultanément en deux grandes catégories (voir figure 1.3) : (1) méthodes basées sur une population et (2) méthodes basées sur une seule solution. Les méthodes basées sur une population manipulent un ensemble de solutions de l'espace de recherche en même temps et exploitent l'évolution de cet ensemble pour trouver de meilleures solutions.

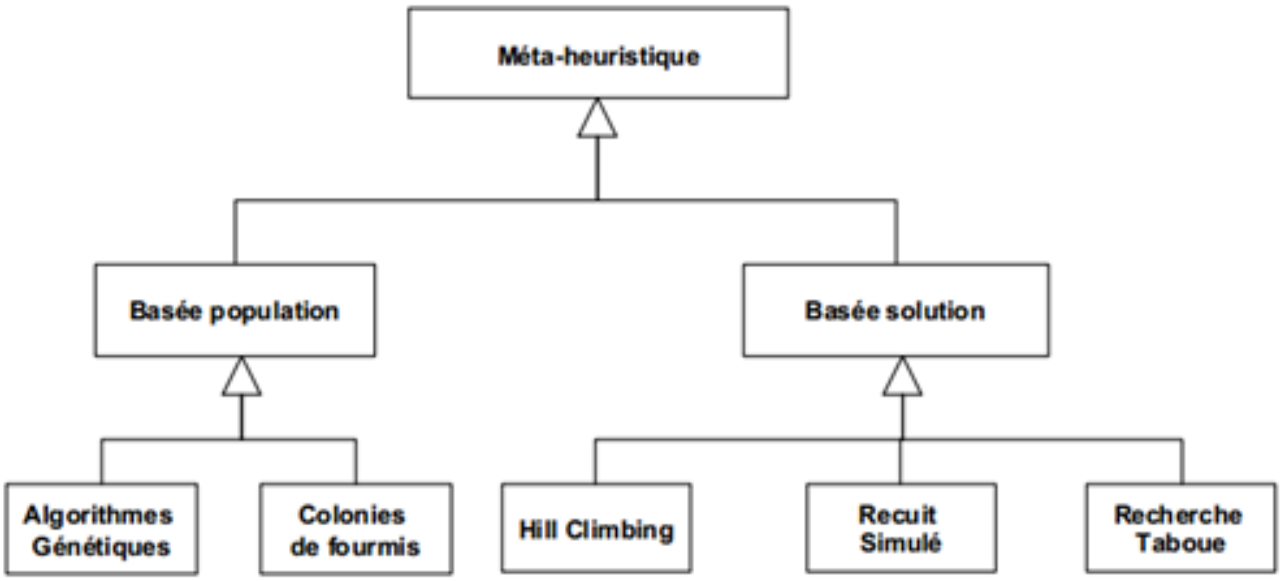

FIGURE 1.3 – Classification des principales méta-heuristiques

1.2.4 Les méta-heuristiques à solution unique :

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherché de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [Bac99].Nous présenterons ici les méthodes les plus utilisées et leur utilisation en extraction de connaissances : les méthodes de descente, le recuit simulé et la recherché tabou.

1.2.5 Les méthodes de descente (Hill Climbing)

Les méthodes de descente sont assez anciennes et doivent leur succès à leur rapidité et leur simplicité [37]. A chaque pas de la recherche, cette méthode progresse vers une solution voisine de meilleure qualité. La descente s'arrête quand tous les voisins candidats sont moins bons que la solution courante ; c'est-à-dire lorsqu'un optimum local est atteint. On distingue différents types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente stochastique et la descente vers le premier meilleur.

Le recuit simulé (Simulated Annealing) :) :

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralise à laquelle on introduit un paramètre de température qui sera ajusté pendant la recherché [45]. Elle inspire des méthodes de simulation de Metropolis (années50) en mécanique statistique. L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.

La recherché Tabou (Tabu search) : La recherché Tabou a été introduite par F. Glover[22] et a montré sa performance sur de nombreux problèmes d'optimisation. Les idées de bases de la recherché Tabou se retrouvent également dans le travail de P. Hansen [Han86]. Elle n'a aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local.

1.2.6 Les méta heuristiques à population de solutions :

Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

1.2.7 Les algorithmes génétiques (Genetic Algorithms) :

Les algorithmes génétiques sont des méthodes bases sur les mécanismes biologiques tels que les lois de Mendel et sur le principe fondamental (sélection) de Charles Darwin [36]. Holland exposa les principes de ces algorithmes pour permettre aux ordinateurs " d'imiter les êtres vivants en évoluant " pour rechercher la solution à un problème [27].

1.2.8 Les colonies de fourmis (Ants System)

Le système de fourmis (Ants System- AS) est une méthode d'optimisation base sur ces observations proposes par Dorigo [15]. Le système de fourmi sa été employé avec succès sur des nombreux problèmes (voyageur de commerce, affectation quadratique,...) mais les auteur sont remarqué que l'AS n'a pas un comportement très exploratoire ce qui a conduit les auteurs à utiliser des hybridations du système de fourmis avec des recherches locales. Les colonies de fourmi sont été utilisées en extraction de connaissances. On retrouve notamment leur utilisation pour effectuer des tâches de clustering.Ainsi, dans [25], Handl et Meyer proposent d'utiliser des colonies de fourmis pour regrouper des textes issus de moteur de recherché.

1.3 Les algorithme génétiques

Les algorithmes génétiques sont des méthodes stochastiques basées sur une analogie avec des systèmes biologiques. Ils reposent sur un codage de variables organisées sous forme de structures chromosomiques et prennent modèle sur les principes de l'évolution naturelle de Darwin pour déterminer une solution optimale au problème considéré. Ils ont été introduits par Holland (Holland, 1975) pour des problèmes d'optimisation complexe. Contrairement aux méthodes d'optimisation classique, ces algorithmes sont caractérisés par une grande robustesse et possèdent la capacité d'éviter les minimums locaux pour effectuer une recherche globale. De plus, ces algorithmes n'obéissent pas aux hypothèses de dérivabilité qui contraignent pas mal de méthodes classiques destinées à traiter des problèmes réels.

1.3.1 Présentation des algorithmes génétiques (AG) :

Les algorithmes génétiques manipulent un ensemble de points dans l'espace de recherche, appelé population d'individus. Chaque individu ou chromosome représente une solution possible du problème posé. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. Les algorithmes génétiques font évoluer cette population d'individus par générations successives, en utilisant des opérateurs inspirés de la théorie de l'évolution qui sont la sélection, le croisement et la mutation. Cinq éléments de base sont nécessaires pour l'utilisation des algorithmes génétiques [17] :

1- un principe de codage des éléments de la population, qui consiste à associer à chacun des points de l'espace d'état une structure de données, la qualité de ce codage des données conditionnant le succès des algorithmes génétiques ; bien que le codage binaire ait été très utilisé à l'origine, les codages réels sont désormais largement exploités, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles.

2- un mécanisme de génération de la population initiale qui doit être capable de produire une population d'individus non homogène servant de base pour les générations futures ; le choix de la population initiale est important, car il influence la rapidité de la convergence vers l'optimum global ;

dans le cas où l'on ne dispose que de peu d'informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

3- une fonction à optimiser, appelée fitness ou fonction d'évaluation de l'individu .

4- des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état ; l'opérateur de croisement recompose les gènes d'individus existant dans la population alors que l'opérateur de mutation garantit l'exploration de l'espace d'état.

5- des paramètres de dimensionnement, représentés par la taille de la population, le nombre total de générations, ou le critère d'arrêt, ainsi que les probabilités d'application des opérateurs de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure 1.4

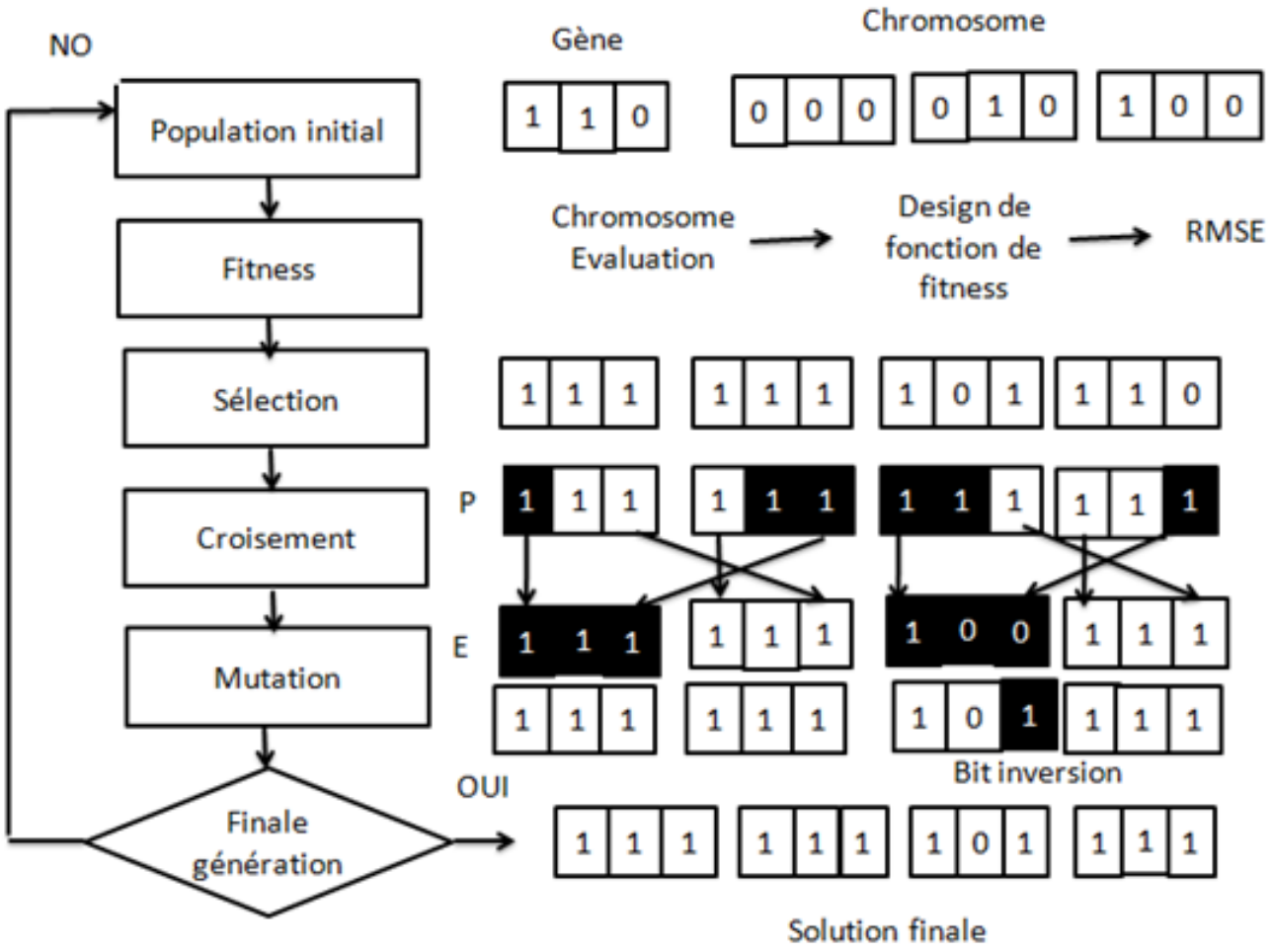

FIGURE 1.4 – Principe général des algorithmes génétiques

1.3.2 Description détaillée

codages des donnéés

Lorsqu'on aborde les algorithmes génétiques, la première tâche consiste à trouver une modélisation adéquate de l'individu qui facilite la description du problème et respecte ses contraintes. Cette modélisation, appelée codage, permet de représenter les solutions sous forme de chromosomes. Il existe principalement trois types de codage : le codage binaire, le codage réel et le codage de Gray :

Codage binaire :Ce codage a été le premier à être utilisé dans le domaine de l'AG, pour se dernier les opérateurs de croisement et de mutation sont souvent assez simples à utiliser [11]. Les premiers résultats de convergence théorique ont également été obtenus avec ce type de codage. Et Chaque gène dispose du même alphabet binaire 0, 1. Un gène est alors représenté par un entier, les chromosomes, qui sont des suites de gènes sont représentés par des tableaux de gènes et les individus de l'espace de recherche sont représentés par des tableaux de chromosomes [32]. Néanmoins ce type de codage présente quelques inconvénients :

1. Les performances de l'algorithme sont dégradées devant les problèmes d'optimisation de grande dimension à haute précision numérique. Pour de tels problèmes, les AG basés sur les chaînes binaires ont de faibles performances comme le montre michalewicz (Michalewicz, 1992).

2. La distance de Hamming entre deux nombres voisins (nombre de bits différents) peut être assez grande dans le codage binaire : l'entier 7 correspond à la chaîne 0111 et la chaîne 1000 correspond à l'entier 8. Or la distance de hamming entre ces deux chaînes est de 4, ce qui crée bien souvent une convergence, et non pas l'obtention de la valeur optimale.

Codage Décimale	Codage Binaire naturel
0	**0000**
1	**0001**
2	**0010**
3	**0011**

FIGURE 1.5 – exemple de codage binaire

• **Codage réel** : Contrairement au codage binaire, un gène est représenté par une suite de bits (un bit dans le code binaire) qui est associé à un réel. Ce type de codage peut être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle. Avec ce type de codage, la procédure d'évaluation des chromosomes est plus rapide vu l'absence de l'étape de transcodage (du binaire vers

le réel) . Les résultats donnés montrent que la représentation réelle aboutit souvent une meilleure précision et un gain important en termes de temps d'exécution [8].

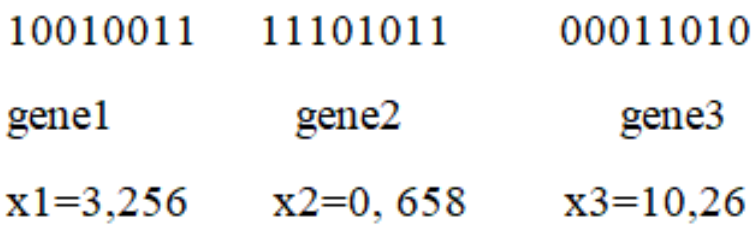

FIGURE 1.6 – exemple de codage reél

• **le codage de Gray** dans le cas d'un codage binaire on utilise souvent la "distance de Hamming" comme mesure de la distance de la similarité entre deux éléments de population, cette mesure compte les différences de bits de même rang de ces deux séquences. Et le codage binaire commence à montrer ses limites. En effet, deux éléments voisins en termes de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "codage de Gray" : le codage de Gray est un codage qui a comme propriété qu'entre un élément n et un élément n + 1, donc voisin dans l'espace de recherche, un seul bit diffère [23].

1.3.3 Génération aléatoire de la population initiale :

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel d'engendrer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes [33].

Si par contre, des informations a priori sur le problème sont disponibles, il paraît bien évidemment naturel d'engendrer les individus dans un sous-domaine particulier afin d'accélérer la convergence. Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités.[33].

1.4 Opérateurs des algorithmes génétiques

1.4.1 Opérateur de sélection

Pour faire évoluer et améliorer le patrimoine génétique d'une génération, les individus d'une population se reproduisent. La reproduction doit alors se faire entre les individus les mieux adaptés.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer partiellement les mauvais. Une opération de sélection est donc nécessaire pour pouvoir choisir les chromosomes qui garantissent l'amélioration de la qualité des solutions.

L'opération de sélection se base essentiellement sur l'évaluation des solutions qui consiste à donner une valeur à un chromosome en fonction de sa qualité. Après avoir évalué une population, ses individus passent par un processus permettant la sélection des parents de la population suivante.

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population [11].

Parmi les techniques de sélection, on cite :

La sélection pour la reproduction : On l'appelle tout simplement l'opération de sélection, et elle permet de choisir les individus qui participent à une reproduction (croisement ou mutation). Cette opération choisit, généralement, les individus les plus forts (meilleurs scores d'adaptation) pour produire l'enfant les plus performants.

La sélection pour de remplacement : l'opération de remplacement, et elle choisit les individus les plus faibles pour être remplacés par les nouveaux.

Il y a plusieurs techniques de sélection :

Sélection uniforme Cette méthode, simple, consiste à sélectionner aléatoirement un individu, d'une manière uniforme sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme (1=N) d'être sélectionné. La convergence de l'algorithme est en général lente [2].

Sélection par la roulette
(loterie biaisée) : C'est la méthode la plus connue des sélections stochastiques, propose par J. Holland [2]. Elle consiste à sélectionner les individus proportionnellement à leur performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés. La probabilité de sélection est calculée à partir de la valeur de "fitness" du chromosome dans la population [39].

Sélection par Tournoi : Cette méthode de sélection augmente les chances des individus de "mauvaise qualité" par rapport à leur fitness, de participer à l'amélioration de la population. En effet, c'est une compétition entre les individus d'une sous-population de taille M prise au hasard dans la population. Le paramètre M est fixé à priori par l'utilisateur. L'individu de meilleure qualité par rapport à la sous-population sera considère comme vainqueur et sera sélectionné pour l'application de l'opérateur de croisement [11].

1.4.2 Opération de croisement :

Le croisement permet l'enrichissement de la diversité dans la population en manipulant la structure des chromosomes. En effet, le croisement permet de créer de nouvelles séquences de gènes pour les chromosomes enfants à partir d'une base de configurations des séquences héritées des chromosomes parents. Cette opération se produit selon une probabilité P_c,

fixé par l'utilisateur selon le problème à optimiser.

Son rôle fondamental est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population. Dans la littérature, il existe plusieurs opérateurs de croisement qui dépendent essentiellement du type du codage et de la nature du problème à traiter [31].

• **Croisement à un point** Une position k est choisie aléatoirement dans les chromosomes parents. Les chromosomes enfants sont le fruit d'un échange de deux parties des parents. La figure 1.7 illustre un exemple de croisement à un point.

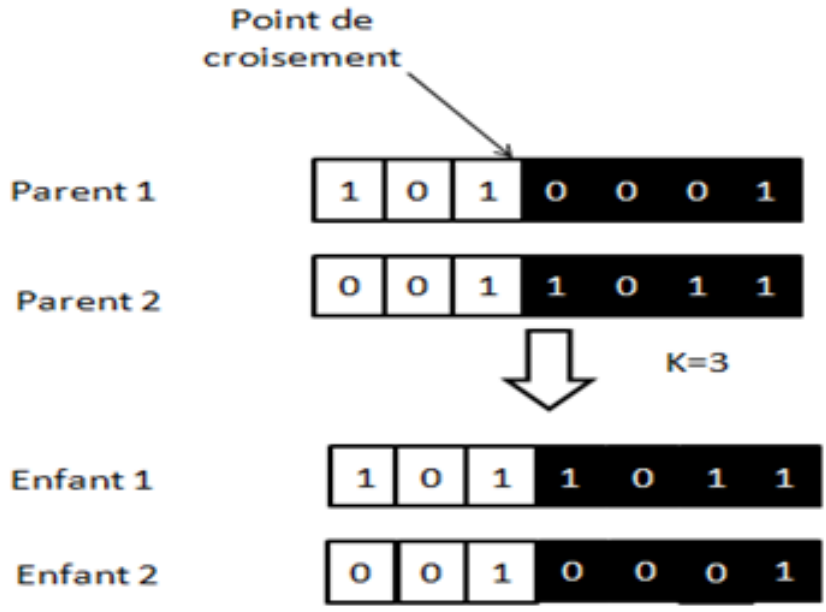

FIGURE 1.7 – Exemple du croisement avec un point

• **Croisement multi-points (deux points)**

Ce croisement est similaire au précédent sauf que plusieurs points de croisement y sont impliqués. La figure 1.8 illustre un cas du croisement multi-points qui est le croisement bi-points. Il consiste à choisir deux points de croisement k1 et k2 et à échanger les segments des deux parents.

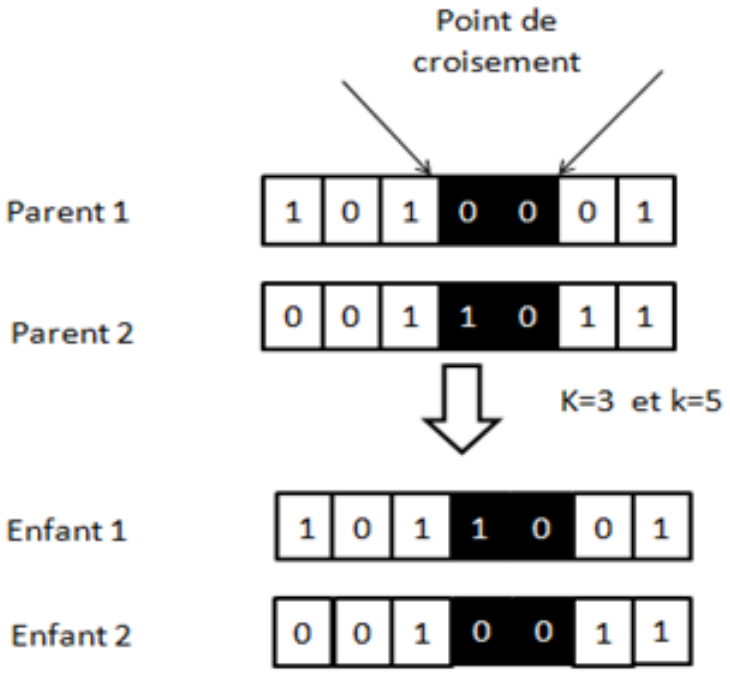

FIGURE 1.8 – Exemple d'un croisement avec multi-points(deux point)

• **Croisement uniforme** Parmi les approches les plus connues du croisement uniforme, on cite l'utilisation du vecteur masque, le masque étant une suite aléatoire de 0 et 1 de la taille du chromosome. Il sert à déterminer de quel parent sont repris les gènes. La figure 8 présente un exemple du croisement uniforme utilisant le masque : en présence du 0, l'enfant hérite du parent 1 et en présence de 1, l'enfant hérite du parent 2.

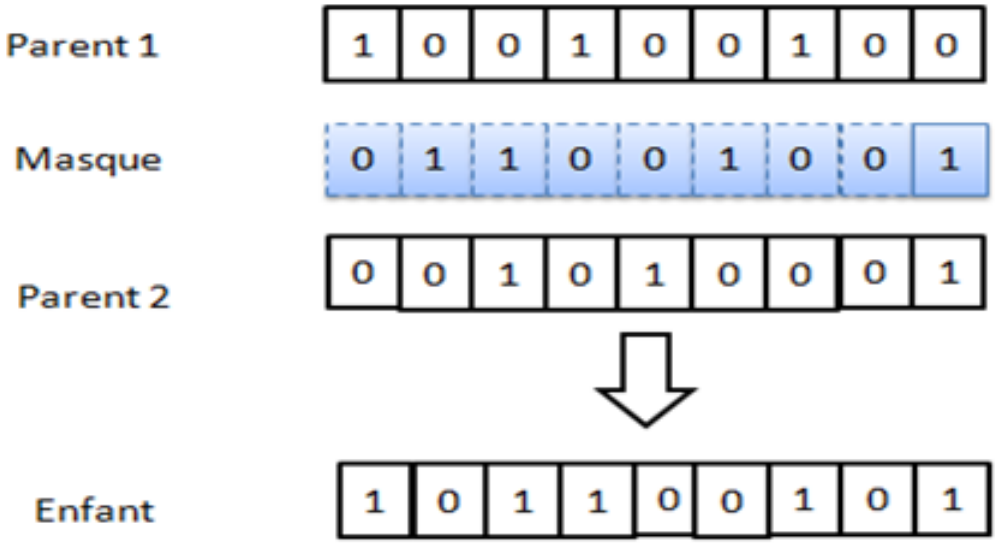

FIGURE 1.9 – Exemple du croisement avec uniforme

1.4.3 Opération de mutation :

La mutation est un changement aléatoire occasionnel avec une faible probabilité Pm (fixée par l'utilisateur) de la valeur d'un ou plusieurs gènes d'un chromosome. En général, la mutation permet de garder une diversité dans l'évolution des individus et d'éviter les optimums locaux.

• **Mutation uni-point** Cette mutation consiste en la transformation aléatoire d'une seule valeur d'un chromosome. La figure 1.9 présente un exemple d'une mutation uni-point.

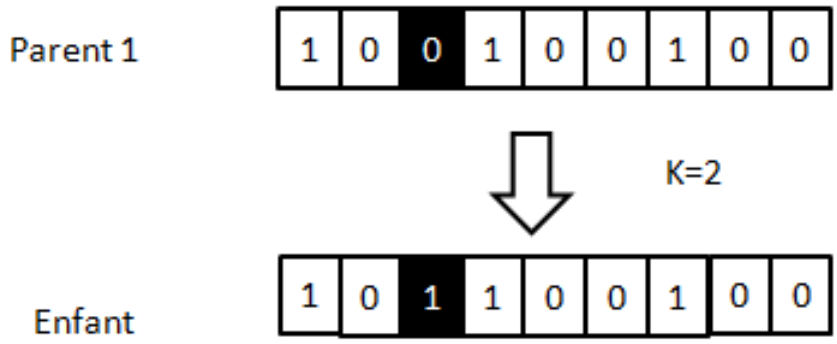

FIGURE 1.10 – Exemple d'une mutation uni-point

Mutation multi-points

Elle consiste en la transformation aléatoire de deux ou plusieurs valeurs d'un chromosome. La figure 1.11 donne un exemple d'une mutation bi-points.

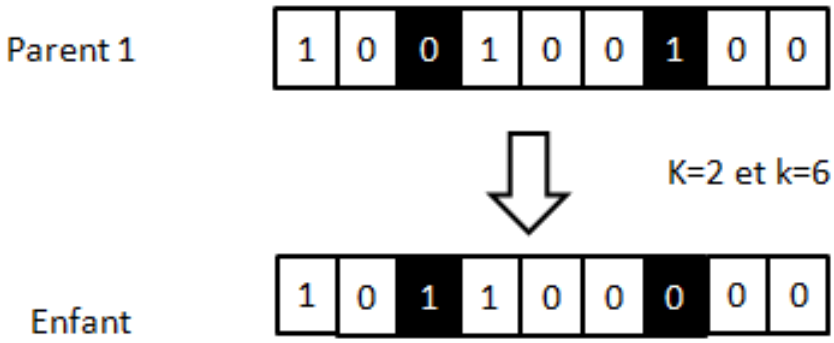

FIGURE 1.11 – Exemple d'une mutation bi-points

1.5 Exemple du voyageur de commerce :

Nous allons maintenant nous intéresser à une problème plus concrète : le problème du voyageur de commerce. Cet exemple est un classique appartenant à la classe des problèmes NP-complets. Il consiste à visiter un nombre N de villes en un minimum de distance sans passer une fois par la même ville .

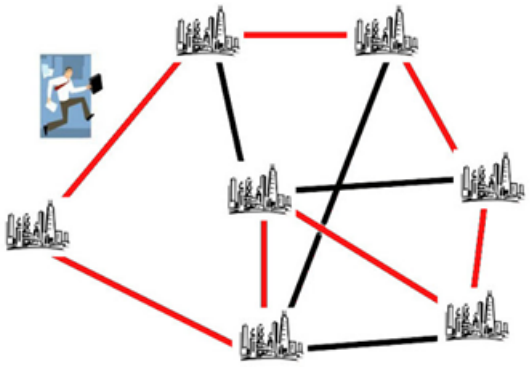

FIGURE 1.12 – Une carte montre les villes

1.5.1 Codage des points de l'espace de recherche :

Une première idée serait de coder chaque permutation (i.e : chaque point de l'espace de recherche) par une chaîne de bits. Par exemple, pour n = 10 : 0011 0111 0000 0100 0001 0010 0101 0110

1.5.2 Représentation d'une solution :

Comme nous l'avons déjà dit le voyageur de commerce doit revenir à son point de départ et passer par toutes les villes une fois et une seule. Nous avons donc codé une solution par une structure de données comptant autant d'éléments qu'il y a de villes, c'est à dire une permutation. Chaque ville y apparait une et une seule fois. Il est alors évident que selon la ville de départ que l'on choisit on peut avoir plusieurs représentations différentes du même parcours.

TourManager : il s'agit toutes les villes

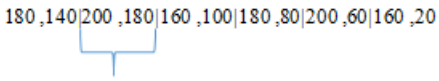

FIGURE 1.13 – Tour de manager montre les villes

City Initialisé population : Nous tournons 50 fois à chaque fois que nous lisons Dans chaque tour, nous lisons, changer l'ordre, par exemple : A B C D E F ils deviennent C F A C B D Et nous stockons dans le Tour :

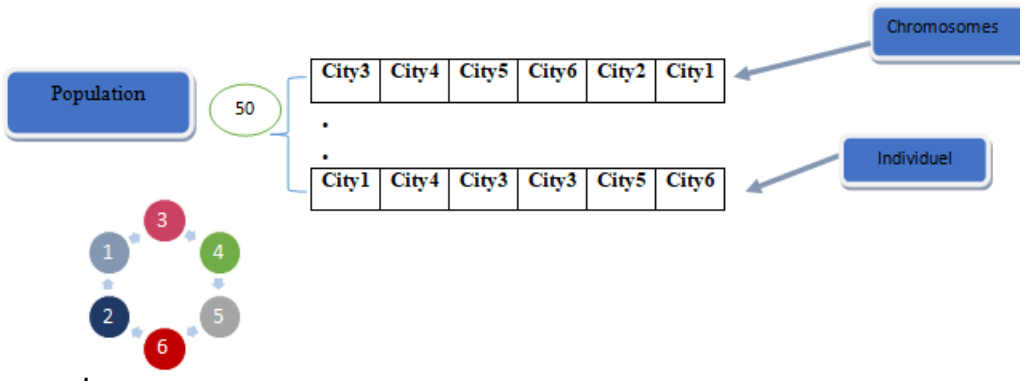

FIGURE 1.14 – codage d'un solution (ensemble de villes) dans un tableau

La sélection :

Nous utilisons ici la méthode de sélection par roulette.

Algorithme Génétique (GA) contient une fonction évolue population Ce qui permet Créé population vide

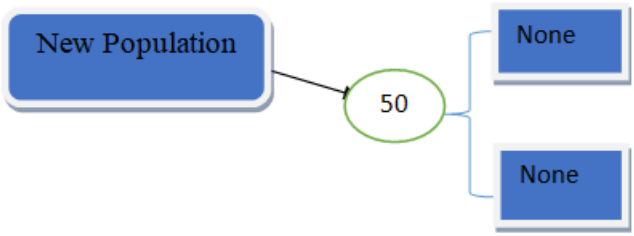

FIGURE 1.15 – creé une population vide

Et puis sélectionnez 5 chromosomes au hasard et prendre le meilleur le met en parent1 Et le même avec parent 2 Ce processus est répété 50 fois et sauvegarder crossover parent1 et parent2 à new population puis la mutate new population

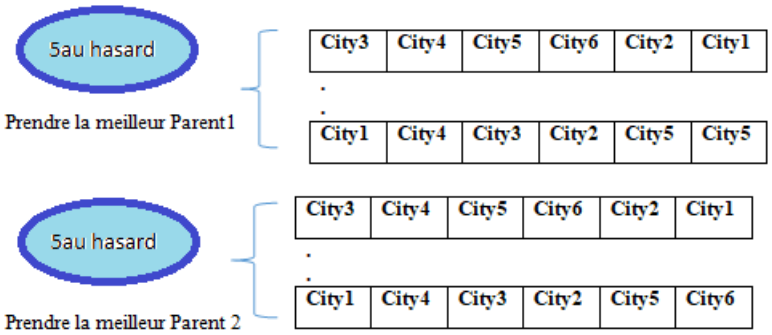

FIGURE 1.16 – selection les meillieur chromosomes

Croisement :

Nous utilisons ici crossover a deux point Nous prenons des valeursaléatoires star pos et end po Pour Prendre des valeurs de àstar pos et end pos dans parent1 Et compléter le reste parent2

star pos=2 end pos=5 start Pos<end Pos and i >Start Pos and i <end Pos

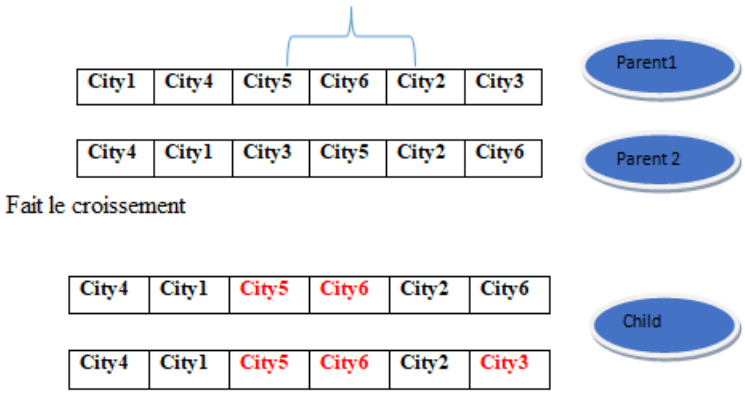

FIGURE 1.17 – croisment des chromosomes

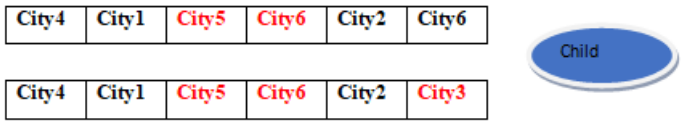

FIGURE 1.18 – mutation des chromosomes

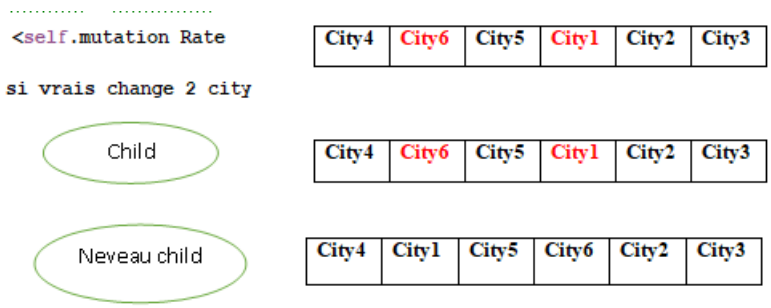

FIGURE 1.19 – nouveau enfants

1.6 Conclusion

Dans cette partie, nous avons présente les algorithmes génétiques de manière générale et donné les éléments de base nécessaires à la compréhension de la méthode . nous avons montré que les algorithmes génétiques fournissent de très bonnes solutions dans un temps raisonnable.

Chapitre 2

Transformation des Modèles

2.1 Introduction

L'ingénierie Dirigée par les Modèles (IDM), ou Model Driven Engineering (MDE) en anglais, est une branche de l'ingénierie du logiciel. Selon [4] En génie logiciel, l'ingénierie pilotée par modèle (IDM) est un ensemble de pratiques basées sur le concept d'un modèle de domaine. Ces pratiques visent à automatiser la production, la maintenance ou l'utilisation de systèmes logiciels. L'objectif de cette approche est de concentrer les efforts sur le domaine d'application du logiciel plutôt que sur sa mise en œuvre. Émergeant dans les années 2000, l'ingénierie dirigée par les modèles est un sujet de recherche actif qui utilise largement les métamodèles et les transformations de modèles.

2.2 Les principes généraux de l'IDM

L'ingénierie dirigée par les modèles (IDM) est basée sur le principe du "tout est modèle" [5]. L'idée centrale est d'utiliser autant de modèles, à défirent niveaux d'abstraction, que la description d'un système le nécessite. Le code réel de l'application n'est plus considéré comme artefact de première importance car il est obtenu par transformation d'un modèle plus abstrait représentant le processus métier de l'application vers un autre plus concret représentant le système selon une technologie donnée.

2.2.1 Modèle

Généralement, les modèles sont utilisés pour analyser le système spécifié ou pour générer l'implémentation du système requis. Il est presque impossible d'avoir une définition universellement acceptée [5] D'après l'OMG : "Le modèle d'un système est une description ou une spécification de ce système et de son environnement pour un objectif donné. Un modèle est souvent présenté comme une combinaison de diagrammes et de texte"[24] . Cette définition présente l'idée générale de ce qu'est un modèle et comment il est représenté.

2.2.2 Méta-modèle

Un méta modèle est un modèle qui définit le langage utilisé pour décrire des modèles [12]. Un méta modèle représente les concepts du langage de méta-modélisation utilisé et la sémantique qui leur est associée. En d'autres termes, le méta modèle décrit le lien existant entre un modèle et le système qu'il représente. On dit qu'un modèle est conforme à un méta modèle si l'ensemble des éléments du modèle sont définis par le méta modèle.

Un méta modèle devrait faire partie d'une architecture de méta-modélisation, qui permet à un méta modèle à être considérée comme un modèle, et d'une manière similaire, il est lui-même décrit par un autre méta modèle. On désigne ce méta modèle particulier par le terme méta-méta modèle [12].

le meta Object Facility (MOF)[9] qui a été déni par l'OMG comme standard pour la définition de méta modèles et son implémentation par l'Eclipse Modelling Framework (EMF) [10].

2.2.3 Meta Object Facility

Le Meta Object Facility (MOF) est la technologie proposée par l'OMG pour définir des méta-données. Nous pouvons définir une méta-donnée comme étant une donnée sur les données. Le but du MOF est de fournir un cadre de travail (framework) supportant tout type de méta-donnée et permettant la définition de nouveaux types au fur et à mesure de l'évolution des besoins. Le MOF spécie la structure et la syntaxe de nombreux métamodèles comme UML (Unied Modeling Language), CWM (Common Ware house Metamodel) et SPEM (Software Process Engineering Metamodel). Le MOF spécie aussi des mécanismes d'interopérabilité entre ces méta modèles[9]

2.2.4 Eclipse Modeling Framework (EMF)

est un framework de modélisation et une infrastructure de génération de code pour la construction d'outils et des applications basées sur des modèles de données structurées [35]. Depuis un modèle décrit généralement en XMI (XML Meta data Interchange) [10], EMF fournit des outils permettant de produire des classes Java représentant le modèle avec un ensemble de classes pour adapter les éléments du modèle an de pouvoir les visualiser, les éditer avec un système de commandes et les manipuler dans un éditeur.

2.2.5 Méta-méta-modèle

Un méta-méta-modèle est le modèle d'un langage permettant d'écrire des langages de modélisation. Il est conforme à lui-même, devenant ainsi le sommet de la pile de méta-modélisation . Ainsi, chaque plateforme de modélisation est basée sur un méta-méta-modèle. On peut citer par exemple MOF et sa version simplifiée EMOF définis par l'OMG [34], Ecore qui est le méta-méta-modèle d'Eclipse [42] ou encore FM3 qui est le méta-méta-modèle du projet FAME [7].

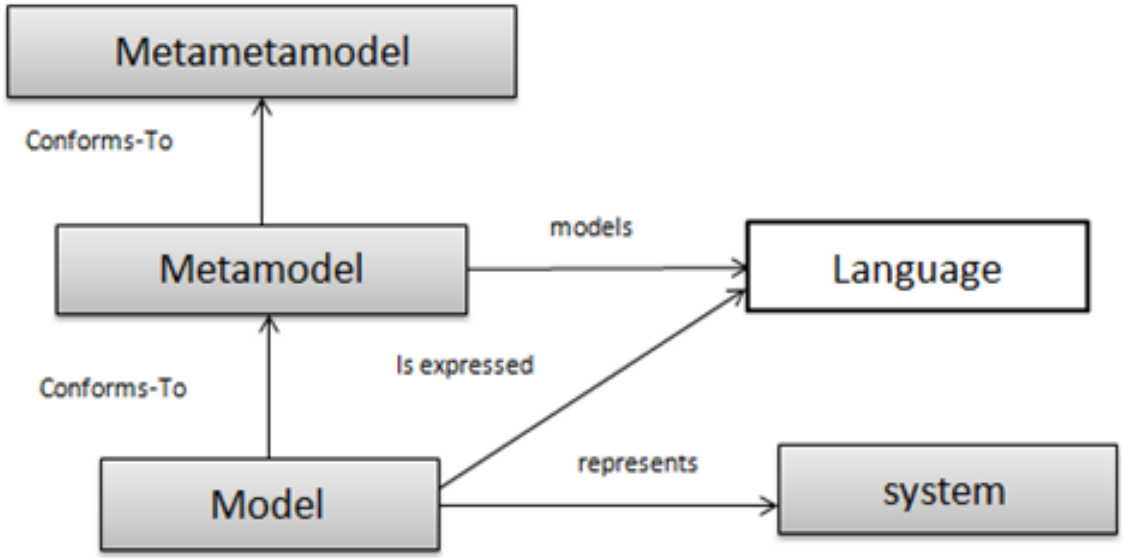

FIGURE 2.1 – Notions de base en ingénierie des modèles

2.2.6 Les langages de l'ingénierie des modèles

De nombreux langages ont été développés pour l'IDM. Ils peuvent être classés en quatre catégories :

Les langages de méta–données : Ecore d'IBM, EMOF de l'OMG ou les schémas XML du W3C.

Les langages de transformation : QVT de l'OMG, ATL de l'INRIA, GreaT de l'Université de Vanderbilt, XSLT du W3C ; Les langages de requêtes : OCL de l'OMG ou XQUERY du W3C.

Les langages d'actions : Action Semantics de l'OMG, Xion de Objection. On peut ajouter à cette classification Kermeta qui est à la fois un langage de méta^{2}modélisation et un langage d'actions.

Dans une approche IDM, si le modèle est la donnée principale, l'opération principale est alors la transformation de modèle. La formalisation des modèles et surtout de leur méta-modèle permet d'envisager la création de programmes manipulant les modèles pour créer d'autres modèles, rendant ainsi les modèles productifs, c'est-à-dire que la création de modèles ne vient pas s'ajouter au processus de développement pour aider les développeurs à maîtriser leur projet mais en est une étape. Les informations contenues dans les domaines pourront être répercutées dans d'autres modèles par le biais d'une transformation.

2.3 Transformations de modèles :

La définition de transformations a pour objectif de rendre les modèles opérationnels dans une approche IDM, ce qui augmente considérablement la productivité des applications. La transformation de modèles est une opération très importante dans toute approche orientée modèle. En effet, les transformations assurent les opérations de passage d'un ou plusieurs modèles d'un niveau d'abstraction donné vers un ou plusieurs autres modèles du même niveau (transformation horizontale) ou d'un niveau différent (transformation verticale). On peut citer comme exemple de transformation verticale, la transformation PIM vers PSM dans l'approche MDA. Le modèle transformé est appelé modèle source et le modèle résultant de la transformation est appelé modèle cible.

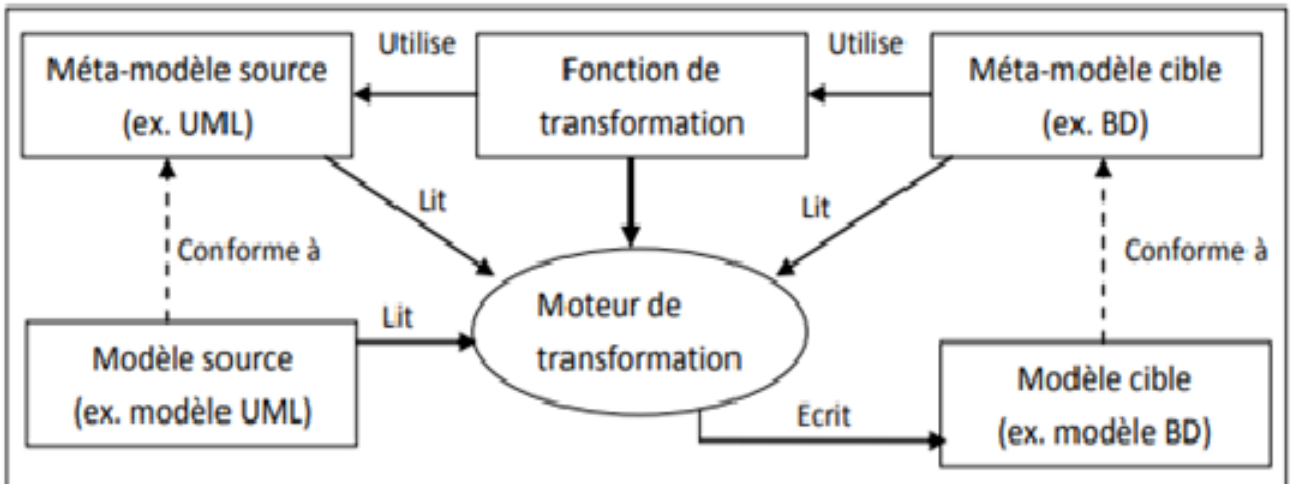

FIGURE 2.2 – Schéma de base d'une transformation de modèles

2.3.1 Cas d'utilisation

La transformation de modèles est cruciale au succès de l'IDM, dans la mesure où elle permet d'automatiser les activités d'évolution, de synchronisation et de génération de modèles. Elle peut opérer verticalement (des modèles à différents niveaux d'abstraction) ou horizontalement (modèles au même niveau d'abstraction décrivant différents aspects du système). Amrani et al. Proposent dans [1] un catalogue de transformations qui regroupe les cas d'utilisation suivants :

• **Manipulation** : elle regroupe les opérations de base telles que l'ajout, la suppression et la modification d'éléments.

• **Interrogation** : cette transformation consiste à extraire certaines informations à partir du modèle manipulé.

• **Raffinement** : elle permet de se déplacer vers un modèle d'un niveau de spécification plus bas que le modèle manipulé.

• **Abstraction** : à l'inverse du raffinement, cette transformation permet d'augmenter le degré d'abstraction.

• **Synthèse** : ce type d'opération permet de produire un langage bien défini qui peut être stocké, un exemple de synthèse est la génération de code.

• **Rétro-ingénierie** : contrairement à la synthèse, elle permet d'extraire un modèle à partir d'artéfacts de bas niveau.

• **Approximation** : cette opération est un raffinement dans lequel le modèle source est une idéalisation du modèle cible.

• **Translation de la sémantique** : elle permet de traduire la sémantique d'un langage dans un autre formalisme.

• **Analyse** : cette transformation fait correspondre le modèle à un formalisme qui peut être analysé plus facilement.

• **Simulation** : elle définit la sémantique opérationnelle du langage de modélisation en mettant à jour l'état du modèle.

• **Normalisation** : cette opération réduit la complexité syntactique du modèle en transformant ce dernier dans une forme canonique.

• **Rendu** : elle permet de faire correspondre une ou plusieurs représentations concrètes d'une même syntaxe abstraite.

• **Génération d'instance** : la génération produit automatiquement des instances (modèle) d'un méta-modèle en particulier.

• **Migration** : transforme les modèles vers un autre langage de modélisation en maintenant le niveau d'abstraction de chacun.

• **Optimisation** : cette catégorie regroupe les transformations qui visent à améliorer les qualités opérationnelles du modèle.

• **Restructuration** : cette opération restructure le modèle afin d'améliorer certaines de ses caractéristiques sans changer pour autant son comportement observable.

2.3.2 Principales approches de transformation de modèles

Dans [6] trois approches de transformations de modèles sont identifiées : l'approche par programmation, l'approche par Template et l'approche par modélisation.

Approche par programmation

consiste à définir des canevas des modèles cibles souhaités. Ces canevas sont des modèles cibles paramétrés ou des modèles Template. L'exécution d'une transformation consiste à prendre un modèle Template et à remplacer ses paramètres par les valeurs d'un modèle source

Approche par modélisation

consiste quant à elle à appliquer les concepts de l'ingénierie des modèles aux transformations des modèles elles-mêmes. L'objectif est de modéliser les transformations de modèles et de rendre les modèles de transformation pérennes et productifs, en exprimant leur indépendance vis-à-vis des plates-formes d'exécution.

Le standard MOF 2.0 QVT de l'OMG a été élaboré dans ce cadre et a pour but de définir un méta-modèle permettant l'élaboration des modèles de transformation de modèles. A titre d'exemple, cette approche a été choisie par le groupe ATLAS à travers son langage de transformation de modèles ATL.

2.3.3 Types de transformation

Une transformation de modèles met en correspondance des éléments des modèles cible et source. Dans [44], on distingue les types de transformation suivants :

•**Une transformation simple** (1 vers 1) qui associe à tout élément du modèle source au plus un élément du modèle cible. Un exemple typique de cette situation est la transformation d'une classe UML munie de ses opérations et de ses attributs en une classe homonyme en Java .

•**Une transformation multiple** (M vers N) prend en entrée un ensemble d'éléments du modèle source et produit un ensemble d'éléments du modèle cible. Les transformations de décomposition de modèles (1 vers N) et de fusion de modèles (N vers 1) sont des cas particuliers de transformations multiples .

•**Une transformation de mise à jour** encore appelée transformation sur place consiste à modifier un modèle par ajout, modification ou suppression d'une partie de ses éléments. Dans ce type de transformation, les modèles source et cible sont confondus. Une telle transformation agit directement sur le modèle source sans créer de modèle cible. Un exemple typique d'une telle transformation est la restructuration de modèles (Model Refactoring) qui consiste à réorganiser les éléments du modèle source afin d'en améliorer sa structure ou sa lisibilité.

2.3.4 Axes de transformation

Selon la classification de [7] la transformation de modèles est caractérisée par trois axes principaux : l'axe processus, l'axe méta modèle et l'axe paramétrage.

• **L'axe processus** :permet de positionner fonctionnellement les transformations par rapport au processus global d'ingénierie. Il est composé de deux sous-axes : le sous-axe vertical qui consiste à faire une transformation de modèles en changeant de niveau d'abstraction (par exemple : raffinement d'un modèle, passage de PIM vers PSM), et le sous-axe horizontal qui consiste à faire une transformation de modèles en restant au même niveau d'abstraction (par exemple : Restructuration de modèle ou Model Refactoring) .

•**L'axe méta-modèle** : permet de caractériser l'importance des méta-modèles mis en jeu dans une

transformation. Par analogie avec la programmation classique, un algorithme vérifie la conformité des types des variables mis en jeu dans une opération. Nous pouvons faire le parallélisme avec un algorithme de transformation de modèles qui permet de faire des opérations entre modèles (différence, composition, fusion, etc.). Dans le cas de la transformation, les méta-classes des méta-modèles correspondent aux types des variables, d'où l'influence des méta-modèles dans le code d'une transformation de modèles. Exemple de transformation utilisant l'axe méta-modèle : UML to SysML[20].

•**L'axe paramétrage** : permet de caractériser le degré d'automatisation des transformations avec la présence de données pour paramétrer la transformation. Le passage des informations à une transformation de modèles peut être soit interne (par exemple, des valeurs ou des relations fixées dans des règles), soit transmis à la transformation (par exemple, le modèle source).

2.3.5 Taxonomie des transformations

Partant de la nature des métamodèles source et cible, on distingue selon la classification adoptée dans [12] les transformations dites endogènes et exogènes combinées à des transformations dites verticales et horizontales

•**Transformations endogènes/exogènes** : une transformation est dite endogène si les modèles source et cible sont des instances du même méta-modèle. Inversement, elle est dite exogène si les deux modèles sont conformes à deux méta-modèles différents.

•**Transformations in-place/out-place** : une transformation est dite in-place, lorsque la manipulation permettant d'obtenir le modèle cible est effectuée directement sur le modèle source. La transformation est out-place si le modèle cible est construit à partir des propriétés d'un autre modèle.

•**Transformations horizontales/verticales** : une transformation est dite horizontale lorsque le modèle produit par cette dernière se situe au même niveau d'abstraction que le modèle source. Elle est dite verticale quand les modèles source et cible sont à des niveaux d'abstraction différents.

•**Transformations syntactiques/sémantiques** : les transformations qui transforment la syntaxe uniquement sont dites syntactiques. Les transformations, plus sophistiquées, qui prennent en considération la sémantique du modèle source lors de la manipulation sont dites sémantiques.

La FIGURE 2.3 résume les combinaisons possibles entre transformations de modèles.

2.3.6 Propriétés des transformations :

Les principales propriétés qui caractérisent les transformations de modèles sont : la réversibilité, la traçabilité, la réutilisabilité, l'ordonnancement et la modularité

•**Réversibilité** : une transformation est dite réversible si elle se fait dans les deux sens. Exemple : Model to Text et Text to Model.

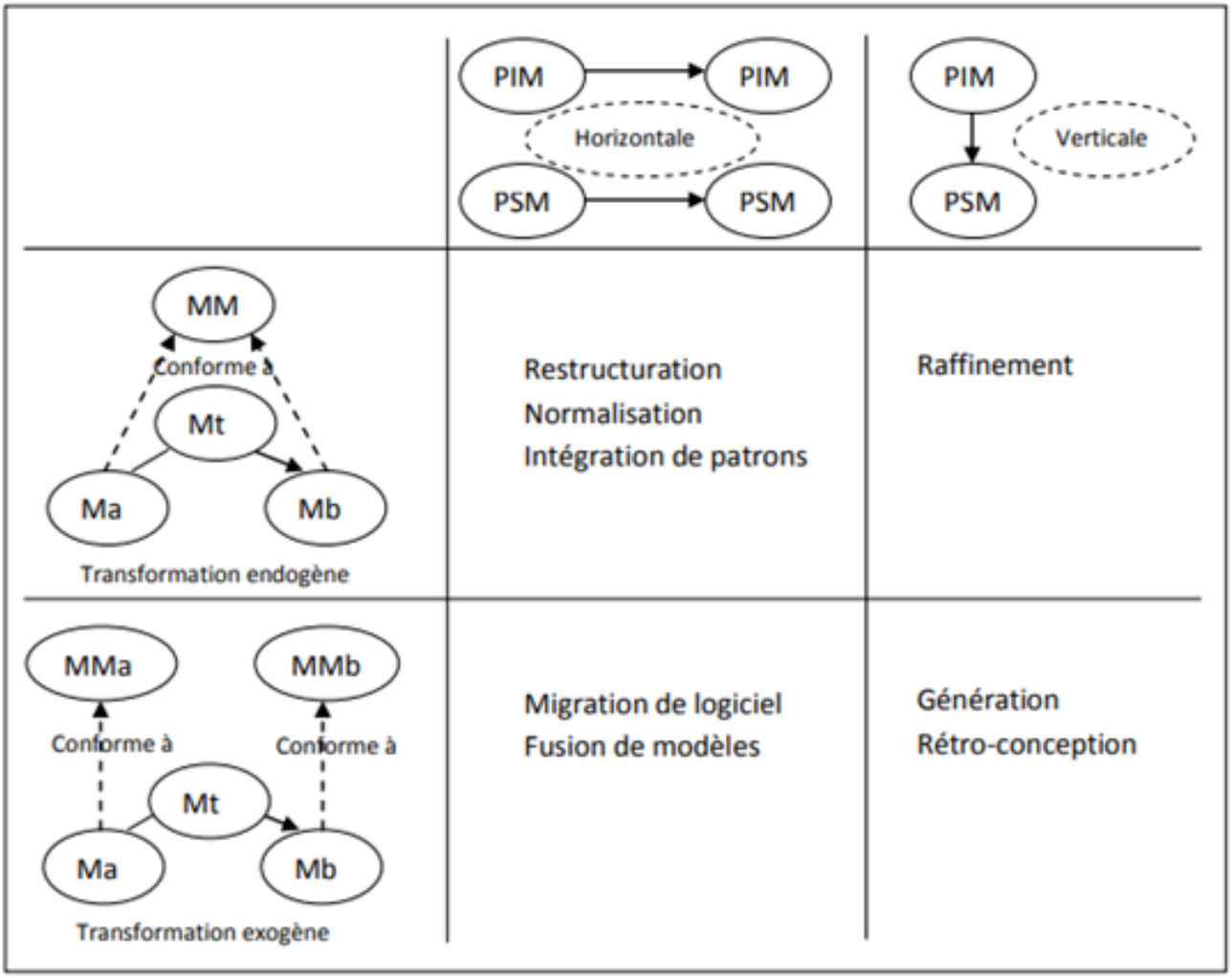

FIGURE 2.3 – Taxonomie des transformations de modèles

•**Traçabilité** : la traçabilité permet de garder des informations sur le devenir des éléments des modèles au cours des différentes transformations qu'ils subissent. Dans un contexte d'ingénierie dirigée par les modèles, il est normal que l'information relative à la traçabilité soit considérée comme un modèle. Un modèleest donc associé à chaque exécution d'une transformation tracée. La définition d'un méta modèle de traces permet de structurer les traces qui seront générées par la plate-forme de traçabilité et ainsi de mieux les manipuler.

•**Réutilisabilité** : la réutilisabilité permet de réutiliser des règles de transformation dans d'autres transformations de modèles.

•**Ordonnancement** : l'ordonnancement consiste à représenter les niveaux d'imbrication des règles de transformation. En effet, les règles de transformations peuvent déclencher d'autres règles.

•**Modularité** : une transformation modulaire permet de mieux modéliser les règles de transformation en faisant un découpage du problème. Un langage de transformation de modèles supportant la modularité facilite la réutilisation des règles de transformation.

2.4 Transformation de modèles par l'exemple :

Les transformations de modèles sont au cœur de l'ingénierie dirigée par les modèles. Elles sont habituellement développées par des programmeurs spécialisés et leur code doit être remis à jour lors de toute variation des besoins ou des méta-modèles impliqués. Pour faciliter le développement de ces transformations, une approche possible consiste à générer des règles de transformation à partir d'exemples de transformation . L'avantage des exemples est qu'ils peuvent être d'une part écrits dans une syntaxe concrète, plus accessible pour l'utilisateur qu'un langage de transformation et d'autre part plus faciles à définir que les règles elles-mêmes. Beaucoup de ces transformations sont assez simples à appréhender, puisqu'elles consistent à associer un motif du modèle source à un motif du modèle cible, notamment dans le cas d'un changement de méta-modèle pour la représentation d'un même domaine (par exemple une transformation d'UML vers un modèle Entité-Relation, ou d'UML vers Java)[14].

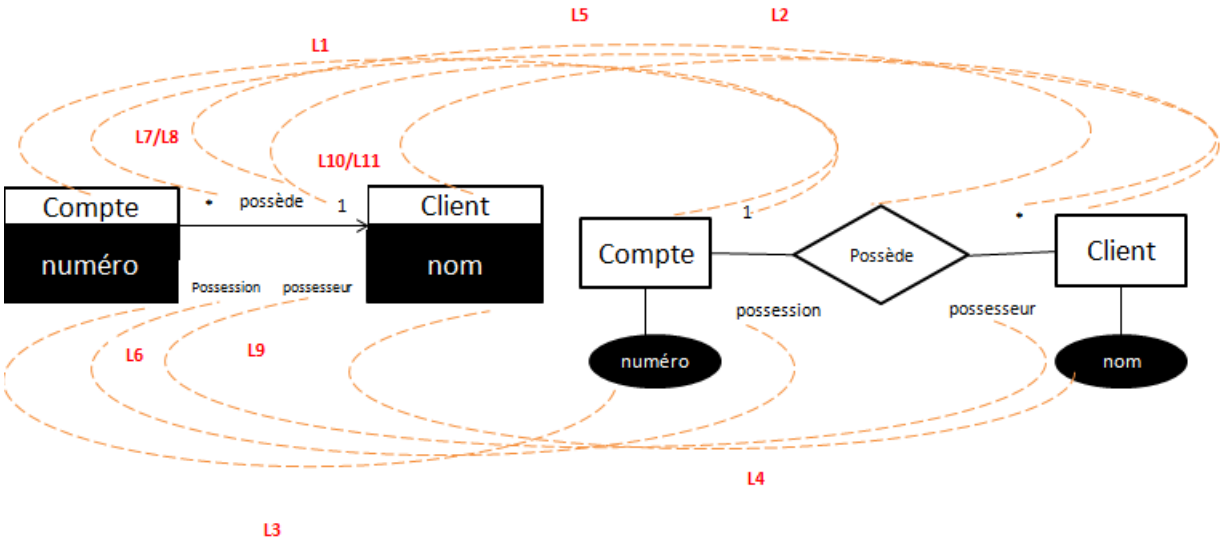

FIGURE 2.4 – Exemples de modèles dans leur syntaxe concrète, UML à gauche et Entité-Relation à droite

[14]

2.4.1 Approches existantes

Bien que les travaux sur la programmation par l'exemple ou les requêtes par l'exemple soient assez anciens, nous donnons pour origine de la transformation de modèles par l'exemple les travaux de Varró, tout d'abord dans [43] . L'auteur part du constat que les langages de transformation ont

tendance à monter graduellement leur niveau d'abstraction pour se conformer à la spécification QVT (Query, Views and Transformations) de l'OMG [46] qui encourage les langages déclaratifs basés sur les règles de transformation. Il propose alors une approche permettant la spécification semi-automatique d'une transformation, c'est-à-dire une ébauche de la transformation. Cette approche est itérative et chaque itération se découpe en quatre étapes :

– **étape 1** : création manuelle d'un alignement de modèles prototypes. Il s'agit de créer un ensemble de modèles dans le langage source de la transformation, ainsi que les modèles transformés correspondant dans le langage cible, et couvrant les différentes situations possibles de la transformation. Puis on relie les éléments sources et cibles par des liens d'appariement, typés en fonction de la règle de transformation dont ils sont l'illustration. Par exemple dans une transformation allant d'UML vers un modèle Entité-Association, un type de lien d'appariement ClassToEntity illustrera la règle transformant une Class vers une Entity.

– **étape 2** : dérivation automatique des règles de transformation en se basant sur l'alignement de l'étape 1. Ces règles devraient permettre d'obtenir à partir des modèles sources donnés en exemple les modèles transformés correspondant, eux aussi donnés en exemple.

– **étape 3** : raffinement manuel des règles. Les règles obtenues portent sur la structure, et ne prennent pas en compte d'éventuelles modifications de valeurs, qui sont à faire par le développeur.

– **étape 4** : validation des règles obtenues sur de nouveaux cas de tests qui deviennent, lorsque les règles ne sont pas satisfaisantes, de nouveaux prototypes pour une nouvelle itération du processus. Durant la même période Wimmer et al. [44] propose une approche similaire pour dériver des transformations sous la forme de règles de type 11 exprimées en ATL. La contribution en question est également itérative et incrémentale, elle diffère cependant des deux contributions précédentes sur deux aspects. D'abord, les traces de transformation exploitées sont spécifiées dans une syntaxe concrète plutôt qu'abstraite. Ensuite, le processus de dérivation se fait selon une approche orientée objet, contrairement à celui de Varró où des graphes sont utilisés. La contribution de Wimmer est également améliorée dans des contributions subséquentes, par Strommer et al. [40, 41],Où un langage pour la définition de correspondances de type nm est présenté, jumelé à un algorithme de raisonnement pour la dérivation de règles (nm également) à partir des correspondances exprimées. L'usage d'opérateurs de chaînes tel que la concaténation est également brièvement mentionné dans l'une des deux contributions.

Un autre travail qui s'inscrit dans le contexte des transformations de modèles par l'exemple est celui de Garcia-Magarino et al.[21] qui propose un algorithme permettant de générer des règles de transformation de type n m, vraisemblablement dans plusieurs langages de transformation, à partir d'un ensemble de paires de modèles sources et cibles interconnectés (agrémentés par des traces). À l'instar de l'approche utilise des graphes. Afin de pallier le fait que certains langages de transformations ne permettent pas d'écrire des règles de plusieurs à plusieurs, les règles sont d'abord dérivées dans un langage de transformation générique avant d'être transformées dans le langage de transformation souhaité (ATL dans l'article).

Kessentini et al. [28] Utilise des analogies pour effectuer une transformation. Contrairement aux

contributions citées plus haut, cette approche ne produit pas de règles de transformation, mais dérive plutôt le modèle cible directement à partir du modèle source en considérant la transformation de modèles comme un problème d'optimisation. Le problème tel que posé est abordé en utilisant l'optimisation par essaims particulaires (OEP) dans la première contribution et une combinaison de OEP et du recuit simulé (RS) dans la deuxième. L'approche d'apprentissage est ensuite améliorée dans [29] où des règles de transformation sont produites à partir de modèles sources et cibles qui ne sont pas accompagnés de traces de transformation. L'approche est validée sur une transformation de modèles comportementaux (diagramme de séquence vers réseau de Petri colorés). Une autre contribution de TMPE qui ne produisait pas de règles de transformation initialement est celle de Dolques et al. Ce travail se base sur l'analyse relationnelle de concepts (ARC), une variante de l'analyse formelle de concepts, pour classifier les éléments sources et cibles ainsi que les correspondances fournies en entrée. Les patrons identifiés sont organisés dans des treillis partiellement ordonnés et sont ensuite analysés pour filtrer les plus pertinents. Les travaux les plus récents dans le contexte de la TMPE sont ceux de Faunes et al. [19, 18]Dans lesquels la programmation génétique (PG) est utilisée pour faire évoluer une population de transformations sur plusieurs générations jusqu'à produire la transformation attendue. Le processus de dérivation prend en entrée des paires d'exemples de modèles sources et cibles uniquement (sans traces de transformation) et produit en sortie des règles exécutables de type nm. L'approche est ensuite améliorée par la contribution de Baki et al.[3] où le programme génétique tente d'apprendre simultanément les règles de transformation ainsi que le contrôle d'exécution qui doit être exercé sur celles-ci pour former un programme de transformation correct. Cette seconde version permet également de dériver des règles de transformations plus complexes en incluant la négation de conditions, l'usage de primitives de navigation ainsi que la considération des types et des domaines de définition lors de la construction du modèle cible.

2.5 Conclusion :

Dans ce chapitre nous avons présentés les concepts importants liés à l'ingénierie dirigée par les modèles (IDM) qui propose de travailler à un niveau d'abstraction élevé via des modèles pour faciliter la conception des systèmes complexes. Nous avons ensuite présenté les transformations de modèles qui fournissent un moyen efficace pour manipuler automatiquement les modèles et passer d'un formalisme à un autre, l'appariement de méta-modèlesest une approche intéressante dans les cas les plus simples mais les cas les plus complexes requièrent un processus différent tel que les approches parl'exemple.

Chapitre 3

Adaptation de l'algorithme Génétique pour la transformation des modèles

3.1 Introduction

Dans ce chapitre, Nous allons présenter la démarche à suivre afin d'adapter l'algorithme génétique pour le problème de transformation de modèles, en détaillant chaque composante de cet algorithme et les choix des stratégies tout en justifiant chaque décision de conception.

3.2 Formalisation du problème

Dans le cas d'un problème de transformation de modèle, le meilleur scénario pour transformer un modèle MSi source en un modèle cible MCj idéal répondant aux besoins de l'utilisateur et/ou de l'application et répondant aux objectives de la transformation, le problème est l'explosion du nombre des instance de modèles cibles qui dépend du nombre de fragments générés à partir de modèle source ($\mathcal{MS}$). Par exemple dans le cas de transformation d'un modèle relationnel $\mathcal{MR}$ vers un modèle cible fragmenté qui répond du problème de l'optimisation des requêtes SQL, Si le nombre de fragment de table T_i est m_i, alors le nombre de fragments de la table qui contient des clés étrangères est $N = \prod_{m_i}$.

Le problème de sélection du schéma de fragmentation qui est le modèle cible $calMC$, en considérant une contrainte sur le nombre de fragments généré, est formalisé comme suit :

Etant donné :

- $\mathcal{Q}$= $\{\mathcal{Q}_1,...,\mathcal{Q}_m\}$ un ensemble de requêtes.
- $\mathcal{R}$ une table qui contient les clés étrangères et $\mathcal{T}$ =$\{t_1,...,t_n\}$, t tables de modèle $\mathcal{MS}$.
- W le nombre de fragments de la table de relation $\mathcal{R}$ imposé par le concepteur
  problème de sélection d'un $calMC$ en se basant les opérations de transformations : fragmenter et fusionner consiste à fragmenter et/ou fusionner la table de $calR$ en N fragments selon un schéma de fragmentation qui correspond au MS des tables $calT$ tel que :

— Le coût d'exécution des requêtes soit minimal
— Le nombre de fragment de F soit minimisé : $N \leq W$

3.3 La démarche de transformation

La sélection d'un modèle cible optimal basé sur l'opérateur split/merge (Split pour fragmenter et Merge pour fusionner) quatre étapes importantes : d'abord les informations qui permettent de la spécification d'un schéma de transformation sont extraites. Ensuite, un codage d'un schéma de transformation est spécifie. Puis un ensemble de schéma candidats est générer. Enfin, parmi cet ensemble de schémas possibles, le schéma de fragmentation optimal est sélectionnée.

3.3.1 Préparer la transformation :

A partir de l'ensemble des requêtes, les prédicats de sélection sont extraits. Ces prédicats permettent de spécifier les attributs de sélection sur lesquels portera la transformation des tables T et dont les domaines sont découpés en sous domaines. Ces attributs permettent, à leur tour, d'identifier les dimensions candidates à la transformation Target /fusionnement. Puis, pour chaque table de $calMS$ T, un ensemble de prédicats complet et minimal est générer.

3.3.2 Coder un modèle source à transformé :

Chaque attribut de transformation AR_i de la table T_k, extrait au niveau de l'étape précédente, possède un domaine de valeurs ti. Le découpage du domaine t_i en n sous domaine $st_1, ..., st_{ni}$ permet de spécifier n_i fragments de la table T_k. Pour t attributs AF_1,..., AF_t de la table T_k le nombre de fragments total est égale à $\prod_{n_i}$.

Ce nombre représente le nombre de fragments maximum pour une table, il se peut que certains domaines soient regrouper dans le même domaine ce qui permet de générer un nombre des fragments T inférieur à $\prod_{n_i}$. Afin d'exprimer cela, nous proposons un codage du schéma cible de transformation qui attribue à chaque sous domaine st_j de chaque AR_i un numéro. Les sous domaines du même AF_i ayant le même numéro définissent le même fragment.

Exemple

Soit un modèle source dont le schéma est donné par la figure 3.1. Soient les attributs de fragmentation suivants : "'Genre"' avec les valeurs {F, M}, "'Classe"' avec les valeurs {A, B, C} et "' Age"' dont les valeurs sont représentées par l'intervalle [0, 60]. Le découpage en sous domaines des domaines de ce s 'attributs est : Genre : {F} et {M}, Classe : {A}, {B} et {C}, enfin Age = [0, 20[, [20, 40[et [40, 60].

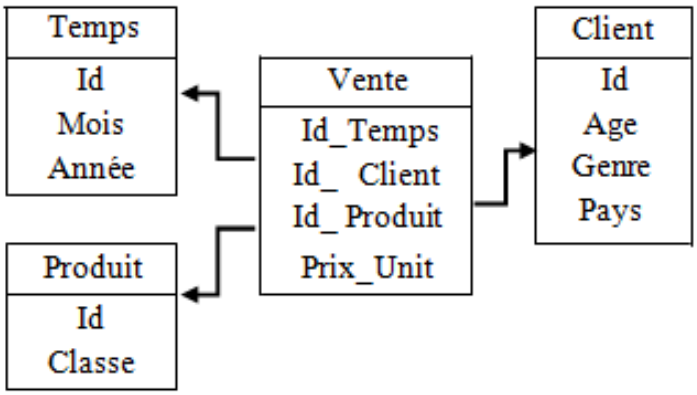

FIGURE 3.1 – Schéma d'un modèle source

Un codage du schéma de transformation est représenté codage d'un schéma de transformation..

Age			Genre		Classe		
[0, 20[	[20, 40[	[40, 60]	F	M	A	B	C
0	1	2	0	0	0	0	1

FIGURE 3.2 – Codage d'un schéma de transformation

Le codage présenté sur la figure 3.2 montre les points suivants :

01 - Les tables concernées par la fragmentation sont la table *Client* sur les attributs *Âge* et *Genre* et la table *produit* sur l'attribut Classe.

02 - Pour la table Client, chaque sous-domaine de l'attribut possède un numéro différent. Par contre, les domaines de l'attribut Genre possèdent le même numéro. Cet attribut ne participe donc pas au processus de fragmentation. Par conséquent la table Client est fragmentée en trois fragments selon l'attribut Âge.

03 - Pour la dimension Produit, les domaines {A} et {B} de classe sont regroupés en un seul domaine (ils possèdent le même numéro 0), ce qui signifie que les tuples Produits dont la classe est égale à "'A"'ou "'B"'appartiennent au même fragment.Cela donne deux fragments de la table Produit.

04 - Générer une solution initiale : nous proposons d'utiliser l'algorithme d'affinités adapté à la FH dans les entrepôts. Une affinité entre deux sous-domaines est la somme des fréquences d'accès des requêtes accédant simultanément aux deux sous-domaines. Appliquer une heuristique : la solution initiale est améliorée grâce à l'application des heuristiques.

a) Réécriture des requêtes

nous proposons une nouvelle démarche de fragmentation non supportée par les SGBD commerciaux. pour les bénéficiers, il faut réécrire les requêtes sur le nouveau schéma de données. Cette réécriture a pour but de charger uniquement les fragments nécessaires pour l'exécution d'une requête donnée. Une fois l'entrepôt de données fragmenté, il peut être représenté sous forme de plusieurs sous-schémas en étoile. Ainsi, pour calculer le coût d'exécution d'une requête sur un schéma fragmenté, il faut calculer ce coût sur chaque sous-schéma. Avant cela, il faut identifier les sous-schémas valides et leur correspondance pour une requête.

1. identifier les sous schémas de MC valides pour la requête comme suit :
2. Identifier les fragments de dimensions valides pour la requête (comparer les prédicats de la requête et ceux formant la clause définissant chaque fragment de dimension)
3. Utiliser ces fragments pour déterminer les fragments de faits valides.
4. Exécuter la requête sur chaque sous-schéma. Pour ce faire il faut identifier la correspondance entre la requête et le fragment de faits pour déterminer les jointures à exécuter en plus des jointures spécifiées dans la clause WHERE. Cette correspondance peut être :
   — **Totale :** la réponse à la requête est le fragment de faits, aucune jointure n'est requis
   — **Partielle :** le fragment de faits contient des tuples non pertinents pour la requête, il faut effectuer des jointures avec les tables de dimensions pour les éliminer.

3.4 Modèle de coût

La transformation de modèle est effectuée par heuristique à savoir l'algorithme génétique. Cet algorithme est guidé par un modèle de coût qui permet d'évaluer chaque schéma de transformation cible généré par génétique. Cela permet de choisir la configuration qui optimise le mieux la charge de requête. En prenant l'exemple de la fragmentation, les approches qui se basent sur la complétude et minimalité des prédicats ou ceux qui utilisent les affinités ne donnent aucun mécanisme permettant d'estimer la qualité du schéma de fragmentation obtenue. Pour les algorithmes à base d'affinités par exemple, le paramètre essentiel est la fréquence d'accès des requêtes aux données. Ce paramètre n'est pas suffisant car certaines requêtes non fréquentes peuvent engendrer un coût d'exécution important. Il faut utiliser d'autres paramètres comme la taille des tables, la taille des tuples, les facteurs de sélectivité des prédicats de sélection et la taille d'une page système. De plus, le de choix d'un schéma de fragmentation est un problème NP-Complet. Ils nécessitent l'utilisation d'heuristiques qui se basent sur une fonction objective. Cette fonction est spécifiée à partir du modèle de coût.

Le modèle de coût Permets d'estimer le coût d'exécution des requêtes. Il est composé de deux grandeurs :

(1) le nombre d'opérations d'entrée sortie (E S) nécessaire pour charger les données exploitées par une requête.

(2) le coût de calcul CPU. Généralement, le coût CPU est négligeable devant le coût d'E/S. Ainsi, seul ce dernier est pris en compte pour estimer le coût d'exécution.

Dans notre travail, nous allons effectuer la transformation de modèle à base d'algorithme génétique. Il faut donc établir un modèle de coût mathématique qui va permet de spécifier la fonction objective et guider la sélection des structures d'optimisation. Afin d'établir ce modèle de coût, nous présentons la formulation du coût d'exécution des jointures ainsi que le coût d'exécution d'une requitte sur un schéma non optimisé.

Formule de coût sur un schéma source non optimisé Soit un schéma source d'un modèle de données modélisé en étoile avec une table de fait R et d tables de dimensions T1, ...Tn. Soit une charge de requête Q1, ... Qn.le coût d'exécution d'une requête sur un entrepôt de données représente le coût de chargement des tables pour effectuer les jointures, est donné par la formule suivante (on suppose que les jointures intermédiaires ne nécessitent pas une écriture sur disque) :

$Cost(Qi,\ ) \ = \ 3 \ \times \ (|R| \ + \Sigma d \ 1|Ti|)$

Nous allons présenter, dans les sections qui suivent, la formulation du modèle de coût pour la fragmentation horizontale est les index de jointures binaires.

3.5 Modèle de coût pour la transformation

Le modèle de coût, que nous allons utiliser pour évaluer un schéma source et le schéma de transformation cible (fragmenté), s'inspire du modèle avancé dans . Ce modèle permet de calculer le nombre d'E /S nécessaires pour exécuter une charge de requêtes sur un schéma fragmenté par fragmentation horizontale. Plus exactement une fragmentation horizontale primaire sur les dimensions et une dérivée sur la table de fait. Chaque requête comporte des prédicats de sélection et des jointures en étoile entre fait et dimension.

Les paramètres utilisés dans ce modèle de coût sont classés en trois catégories :

Les paramètres sur l'entrepôt de données comme la taille des tables, la taille d'un tuple, la taille des tables en pages systèmes nécessaires pour leur stockage.

Les paramètres sur le système physique à savoir la taille de la page système.

Les paramètres sur la charge de requêtes comme la sélectivité des prédicats.

Nous considérons un entrepôt de données modélisé en étoile avec une table de fait R et d table de dimensions T1, ...Td. La fragmentation de l'entrepôt donne N sous schéma en étoiles S1, ... SN. Soit une requête Q a exécuté sur l'entrepôt.

Le coût d'exécution d'une requête sur un sous schéma si estime, le coût d'exécution d'une requête sur un sous schéma si estime le coût de chargement du fragment fait puis la jointure entre ce fragment et les fragments dimensions.

$$Cost(Q, MS_i) = valide(Q, S_i \ \times \left[3 \times \left[\prod_{j=1}^{mi} Sel(PR_j) \times |R| \sum_{k=1}^{d} \prod_{j=1}^{L_i} sel(PM_j\ ) \times |T_k|\right] \ \right] \tag{3.1}$$

Le coût total d'exécution de la requête sur tout l'entrepôt fragmenté est
$Cost\ (Q, SR) = \sum_{i=1}^{N} Cost(Q, S_i)$

3.6 Fonction objective pour l'algorithme génétique

Afin de sélectionner les MC nous avons utilisé un algorithme génétique qui exploite une fonction objective. Cette fonction évalue, pour chaque itération de l'algorithme génétique, le taux d'optimisation du coût d'exécution, en matière d'E S, d'un ensemble de requêtes sur un modèle cible. Cette évaluation est effectuée en comparant ce coût par rapport au coût d'exécution des requêtes sur un entrepôt non optimisé. Ainsi, le but de l'algorithme génétique est de trouver la solution (schéma de fragmentation) qui minimise la fonction objective. Nous présentons, dans ce qui suit, la formulation générale d'une fonction objective ainsi que les différents types de fonctions. Nous exposons ensuite la fonction objective pour la sélection d'un schéma de fragmentation et celle utilisée pour sélection des index

3.6.1 Formulation d'une fonction objective

Soit $F(x)$ comme une fonction objective utilisée dans une heuristique. On s'intéresse généralement à Maximiser F(x) sous la contrainte $C(x)$. C'est un problème d'optimisation d'une fonction F sous une contrainte Soit P en (x) une fonction pénalité que pénalise une solution x qui viole une contrainte. L'utilisation de la fonction de pénalité pour la fonction F(x) permet de transformer le problème en un problème d'optimisation sans contrainte d'une fonction F(x) définie à partir des fonctions F(x) et $Pen(x)$. en utilisant la mode division.

Mode division :

$$F'(x) = \begin{cases} \frac{f(x)}{Pen(x)}\ si\ Pen(x) \sum 1 \\ F(x)\ si\ non \end{cases} \tag{3.2}$$

Nous avons choisi le mode division. Mais la formulation de la fonction objective nécessite d'être adaptée à notre problème de sélection. En effet, le problème présenté ainsi est un problème de maximisation. Or, notre sélection de structure d'optimisation se base sur la minimisation du coût d'exécution des requêtes. L'adaptation du problème nécessité une reformulation de la fonction pénalité. Cette fonction doit pénaliser les solutions qui violent la contrainte en augmentant leur fonction objective. Nous présentons dans ce qui suit , la fonction objective choisie pour la sélection d'index et celle adoptée pour la sélection d'un schéma de fragmentation.

On distingue trois types de fonction objective permettant d'évaluer le coût d'exécution d'une charge de requête sur un MS non optimisé

— La première fonction objective profit » privilégie les structures d'optimisation qui apporte les plus de profit lors de l'exécution de la charge de requête.

— La seconde « fonction objective ratio profit/contraint » favorise les structures d'optimisation qui apportent le plus de profit mais qui ne violent pas une contrainte C.
— Enfin la troisième fonction « fonction objectif hybride » représente une hybridation des deux fonctions annoncée. Elle permet de sélectionner, en premier lieu, les structures qui apportent le plus de profit. Ensuite, si la contrainte risquée d'être violée, la fonction objective sélectionne les structures qui apportent le plus de profit mais qui ne violent pas la contrainte.

Pour notre approche de sélection nous avons retenu la fonction objective avec ratio profit/contrainte. Elle permet de sélection des modèles cibles optimales sous la contrainte liée à la technique, cela en spécifiant la fonction F et la fonction pénalité P en à partir de la contrainte, puis déterminer la fonction générale F'.

3.6.2 Fonction objective pour la fragmentation

Nous allons présenter, dans ce qui suit, la formulation de la fonction objective ainsi que la fonction pénalité pour la sélection, par l'algorithme génétique, d'un schéma de fragmentation.

Soit un entrepôt de données modélisé en étoile avec une table de fait F et d tables de dimensions D1, ...Dn. nous allons considérer ce qui suit :

— Une charge de t requite $\mathcal{Q}$= $\mathcal{Q}_1$,...,$\mathcal{Q}_t$ à executer sur un modèle source $\mathcal{MS}$.
— L'espace de m solutions ou schémas de fragmentations possibles générés par l'algorithme génétique $SF = SF_1,\ldots,SF_m$.
— Nsf_i sous schéma en étoiles relatifs à un schéma $SF_i = S_1,...,SN_i$. Rappelons le coût d'exécution d'une requête Q_k sur un sous schéma Sj :

$$Cost(Q_k, S_j) = Valide\ (Q_k, S_j) \times \left[3\times \quad \left[\prod_{p=1}^{M_j} Sel(P_{F_p})\times|F|+\sum \prod_{\substack{p=1 \\ S=1}}^{\substack{L_j \\ d}} Sel(P_{M_p})\times|D_s|\right]\right] \tag{3.3}$$

Avec M_J

le nombre de prédicats de sélection qui définit un fragment fait LJ^S le nombre de prédicat qui difinissent un fragmenté est le suivant ;

$$Cost(Q_k, SF_i) = \sum_{j=1}^{Ns\ f_i} Cost\ (Q_{t,}SF_j) \tag{3.4}$$

On peut donc exprimer le cout d'exécution de toute la charge de requête par :

$$Pen(SF_i) = \frac{N_s f_i}{W} \tag{3.5}$$

Afin de formuler la fonction objective, il faut définir la fonction pénalité Étant donné une contrainte sur le nombre de fragments W

l'algorithme génétique peut générer des solutions génitique SF_i dont le nombre de fragmentation est égal à :

$$Pen(SF_i) = \frac{N_s f_i}{W} \tag{3.6}$$

Enfin l'algorithme génétique doit sélectionner un schéma de fragmentation qui minimise la fonction objective :

$$F'(SF_i) = \begin{cases} Cost(Q, SF_i) \times Pen(SF_i) \; si \; Pen\,(SF_i) > 1 \\ Cost(Q, SF_i) \; si \; non \end{cases} \tag{3.7}$$

3.7 Mécanisme de codage d'un schéma de transformation

Nous présentons dans cette section le mécanisme de codage que nous avons adopté pour représenter un schéma de transformation de modèle.

3.7.1 Génération d'un modèle de transformation à partir d'un codage

Tout algorithme exploitant notre mécanisme de codage, doit nécessairement pouvoir le décoder, c'est-à-dire générer le schéma de fragmentation correspondant à un code donné. Ce décodage se fait en deux étapes :

1. génération des schémas de fragmentation des tables de dimension,
2. Génération du schéma de fragmentation de la table des faits

3.7.2 Fonction Fitness

Un individu dans notre algorithme génétique représente une solution possible du problème donc un schéma de transformation (TM). Pour évaluer les différentes solutions, nous avons défini une fonction fitness basée sur le modèle de coup vu précédemment. Nous avons formalisé le problème de fragmentation horizontale comme un problème de minimisation, or les algorithmes génétiques traitent des problèmes de maximisation. Pour traiter un problème de minimisation avec les algorithmes génétiques, une transformation de la fonction à minimiser est souvent effectuée pour le ramener à un problème de maximisation. Par conséquent, nous proposons la maximisation du gain obtenu entre le coût d'exécution des requêtes sur l'entrepôt initial et le coût total des requêtes de la charge sur un schéma de fragmentation. Le gain peut être calculé comme suit : Gain = Cout Init – Cout Sch, où : - cout Init : coût d'exécution des requêtes sur l'entrepôt initiale non fragmentée. - Cout Sch : coût d'exécution des requêtes sur un schéma de fragmentation. Notre AG peut générer des solutions violant la contrainte de maintenance (solutions inadmissibles). Pour ne pas être confronté à une population totalement inadmissible, nous nous assurant qu'au moins la moitié de la population est admissible, ceci est réalisé en réparant les solutions inadmissibles. Cette réparation

consiste à fusionner des sous-domaines appartenant à des fragments différents d'un attribut choisi aléatoirement afin d'obtenir un nombre de sous schémas qui satisfait la contrainte W fixée par le concepteur. Rappelons que notre algorithme génétique garde trace de la meilleure solution de chaque génération, de ce fait, nous nous assurons que le meilleur individu sélectionné d'une génération soit admissible, c. -à-d que si ce dernier est inadmissible, nous procédons à sa réparation puis à sa ré-évaluation, ensuite nous le comparons avec le meilleur individu admissible de sa génération, nous sélectionnons enfin le meilleur des deux comme étend le Best de cette génération. Afin de pénaliser ces schémas, une fonction de pénalité Pen(MS)est introduite et fait partie de la fonction objective, elle est définie comme suit :

Pen(MS) = W / nbfragtr, où : - W : contrainte du nombre de fragments maximum à ne pas dépasser. - NbFragSCH : nombre de fragments générés par un schéma de fragmentation.

La valeur de Pen est < 0 si le nombre de fragments d'un schéma sont supérieurs à W, ce qui diminue la valeur de la fonction objective F(SM) et défavorise la probabilité de sélection.

La valeur de Pen est > 0 dans le cas contraire ce qui permet d'augmenter la valeur de F(SM) ainsi que la probabilité de sélection.

F(SF)= Gain*Pen(SM)

3.7.3 Génération de la population initiale

Pour procéder à la création de la population initiale, nous générons aléatoirement N solutions pour former les individus de cette première génération où N'est un nombre constant fixé par l'administrateur et représente le nombre d'individus par génération. La génération aléatoire des solutions est effectuée en remplissant pour chaque individu, d'une manière aléatoire, les cellules dans le tableau multidimensionnel.

La numérotation aléatoire des cellules présente un problème de multi-instanciation. Pour résoudre ce problème, nous appliquons la fonction fitness que nous avons défini auparavant. Cette fonction sera appliquée sur toutes les lignes de tous les individus de la population initiale. L'algorithme génétique améliore ensuite la population initiale en lui appliquant les opérateurs génétiques de sélection, croisement et mutation que nous décrivons dans les sections suivantes.

3.7.4 Sélection

L'opérateur de sélection permet de sélectionner les meilleurs individus d'une population pour participer dans la génération de la population suivante. Dans le cadre de notre travail, nous avons utilisé la sélection par roulette. Cette méthode associe chaque individu à sa qualité (valeur de la fonction d'évaluation).

La roulette est partagée entre tous les individus, la surface allouée à chaque individu est en fonction de la qualité de l'individu. Par conséquent, les meilleurs individus ont plus de chances d'être sélectionnés.

L'implémentation de cette méthode se fait de la manière suivante :

• Calculer la valeur de la fonction d'évaluation de chaque individu et la somme totale de ces valeurs.

• Chaque individu est associé à sa probabilité de sélection (segment de longueur égale à sa valeur de fitness divisée par la somme des fitness déjà calculée).

• Tirer un nombre aléatoire et choisir l'individu dont le segment en globe ce nombre.

3.7.5 Croisement

Le croisement est un opérateur de recombinaison qui 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. Nous utilisons le croisement multipoints entre individus qui peut être considéré comme un croisement monopoint sur chaque ligne (attribut) de l'individu. Après le choix de la position de croisement sur chaque attribut, les deux individus échangent leurs gènes. Cela est effectué en échangeant les numéros des cellules qui se trouvent après le point de croisement entre les deux individus. Cet échange peut rendre la numérotation non conforme, ce qui nous amène à appliquer la fonction Renuméroter sur tous les attributs des individus fils avant l'injection dans la population suivante (voir Figure 3.3).

Le croisement est effectué avec un taux de croisement Tc, généralement grand pour pouvoir combiner des solutions et former des générations de meilleure qualité. Nous décrivons le fonctionnement de l'opérateur de croisement par les étapes suivantes :

• Tirer un nombre aléatoire N (entre 0 et 100)

• Si N > Tc alors il n'y aura pas de croisement. Nous réinjectons les deux parents dans la population suivante.

• Si N < Tc alors sur chaque ligne choisissait aléatoirement une position de croisement et échangeait les gènes entre les deux individus.

• Renuméroter les deux individus et les injecter dans la population suivante.

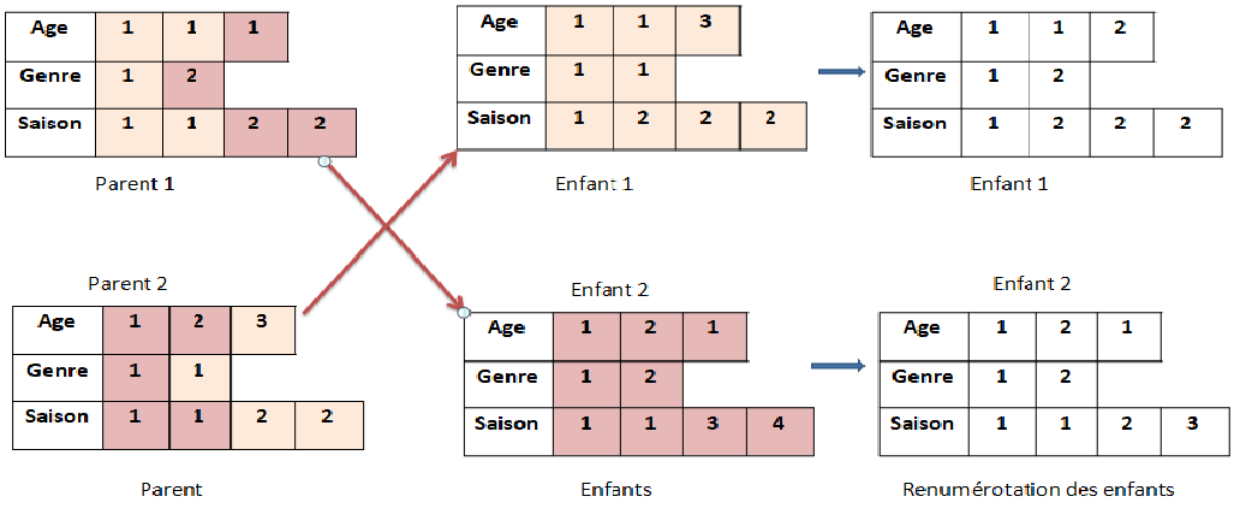

FIGURE 3.3 – Exemple de croisement

3.7.6 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. L'opérateur de mutation que nous avons utilisé permet de modifier une case du tableau multidimensionnel représentant l'individu choisi. Notons que la modification d'une case dans une ligne peut engendrer un problème de multi-instanciation. Pour résoudre ce problème, chaque ligne sera renumérotée (voir Figure3.4).

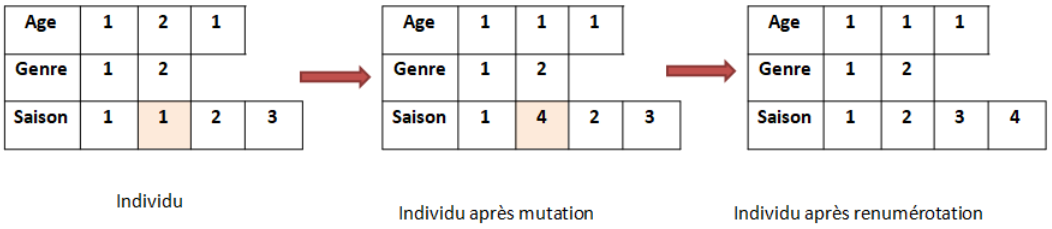

FIGURE 3.4 – Exemple de mutation

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

- Déterminer aléatoirement le nombre d'individus à muter.
- Choisir aléatoirement ces individus.
- Tirer un nombre N aléatoire (entre 0 et 100)
- Si N > Tm alors injecter l'individu dans la population suivante sans mutation.
- Sinon, pour chaque individu choisi, choisir aléatoirement une ligne .

- Choisir aléatoirement sur cette ligne une case à muter.
- Donner une valeur aléatoire à cette case (entre 1 et le nombre de cases).
- Renuméroter la ligne modifiée.
- Injecter l'individu résultat dans la population suivante.

Exemple d'instaciation

La figure suivante illustre le déroulement d'un exemple d'instanciation de notre approche proposée qui consiste à injecter l'individu résultat dans la population suivante :

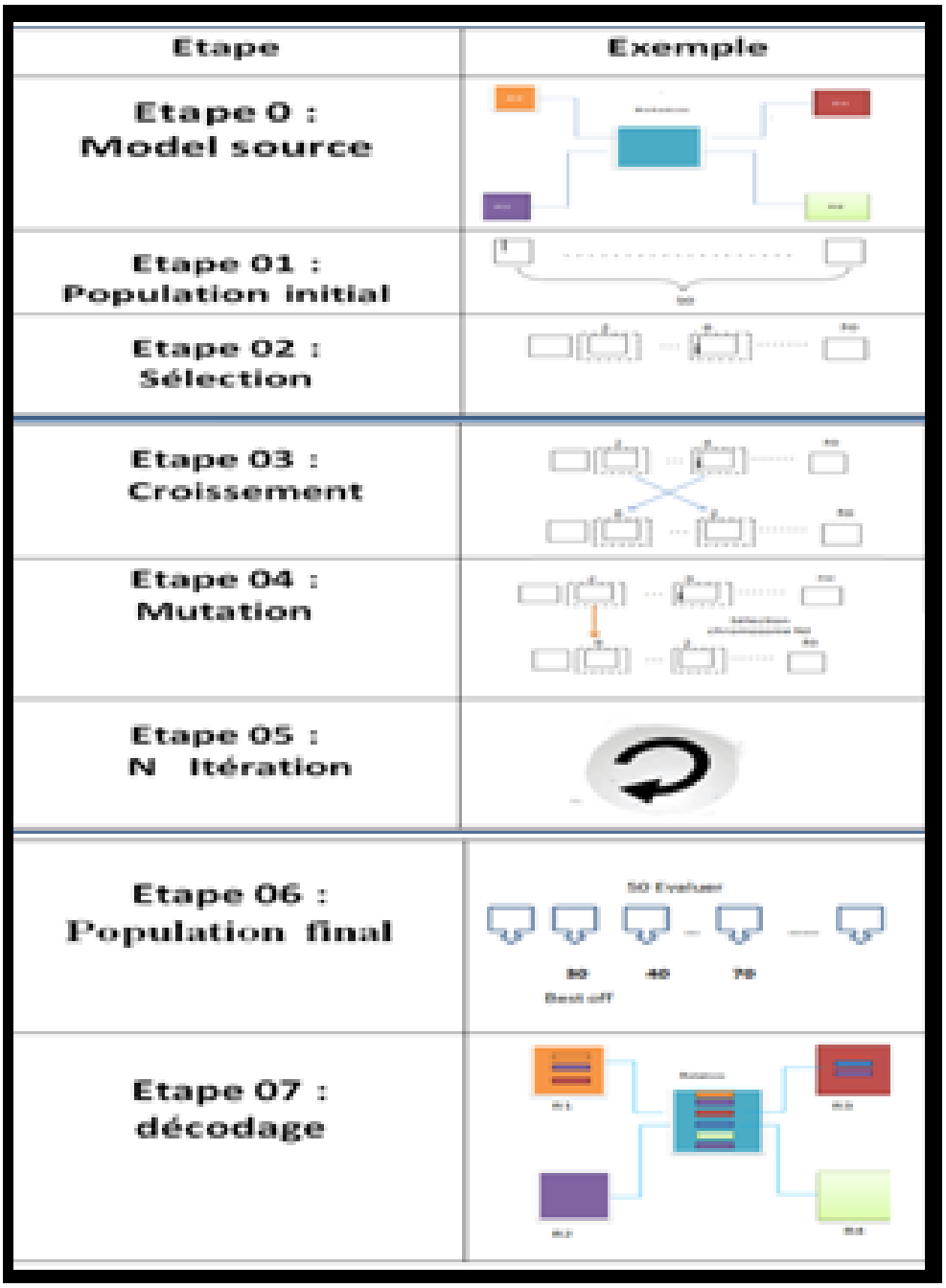

FIGURE 3.5 – Les opérateurs génétiques aux transformations des modèles

3.8 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle démarche de transformation des modèles en se basant sur le schéma de fragmentation. Cette démarche se base sur l'algorithme génétique. Ainsi, nous avons décrit l'architecture de notre démarche de transformation, nous avons présenter les algorithmes de transformation et le modèle de coup utilisé pour guider la sélection des schémas cibles.

Chapitre 4

Développement de notre outil et étude expérimentale

4.1 Introduction

La performance en matière de temps d'exécution des requêtes est un problème clé dans les basess de données. Le choix de type des transformations utilisé impacte le temps d'exécution des requêtes définies sur ce modèle source. Afin de valider nos contributions, nous avons développé un outil prototype, présentée dans le chapitre 3. Notre outil sélectionne un schéma optimal de transformation et également évalué les performances de requetés avec et sans transformation des modèles. Afin de motiver nos travaux de recherche d'un point de vue pratique, nous avons développé un outil qui assiste le concepteur dans ses tâches de transformation des modèles.Nous présentons dans ce chapitre l'environnement d'implémentation, l'outil implémentant notre application ainsi qu'une série d'expériences en utilisant ce dernier

4.2 Environnement d'implémentation

Nous présentons ici brièvement l'environnement d'implémentation et outils utilisés durant la phase de réalisation.

Langage de programmation

Nous avons implémenté notre plateforme avec le langage de programmation JAVA vu que ce dernier est un langage simple, intuitif, orienté objet, performant et dynamique. C'est aussi un langage multiplate-forme disposant d'une JVM (Java virtual Machine) lui permettant de s'exécuter dans des environnements hétérogènes en permettant une indépendance envers les réseaux et les systèmes d'exploitation.

4.2.1 Environnement de Développement Intégré (IDE)

Le choix de l'IDE est souvent dicté par des préférences personnelles. Néanmoins, il est conseillé d'utiliser Éclipse ou Netbeans dans notre cas. Ces deux IDEs facilitent le développement en Java SE et ME . Les développements de ce projet ont été effectués en utilisant Netbeans.

4.2.2 Serveur de bases de données

Nous utilisons les données du banc d'essai Star Schéma benchmark (SSB), qui est basé sur le schéma de données du benchmark TPC-H. implémenté sur oracle 11 g. ce benchmark utilise un schéma en étoile illustré dans la figure 4.1 . Il contient quatre tables de dimension et une table des faits. La taille de chaque table en matière d'instances est décrite dans figure 4.2.

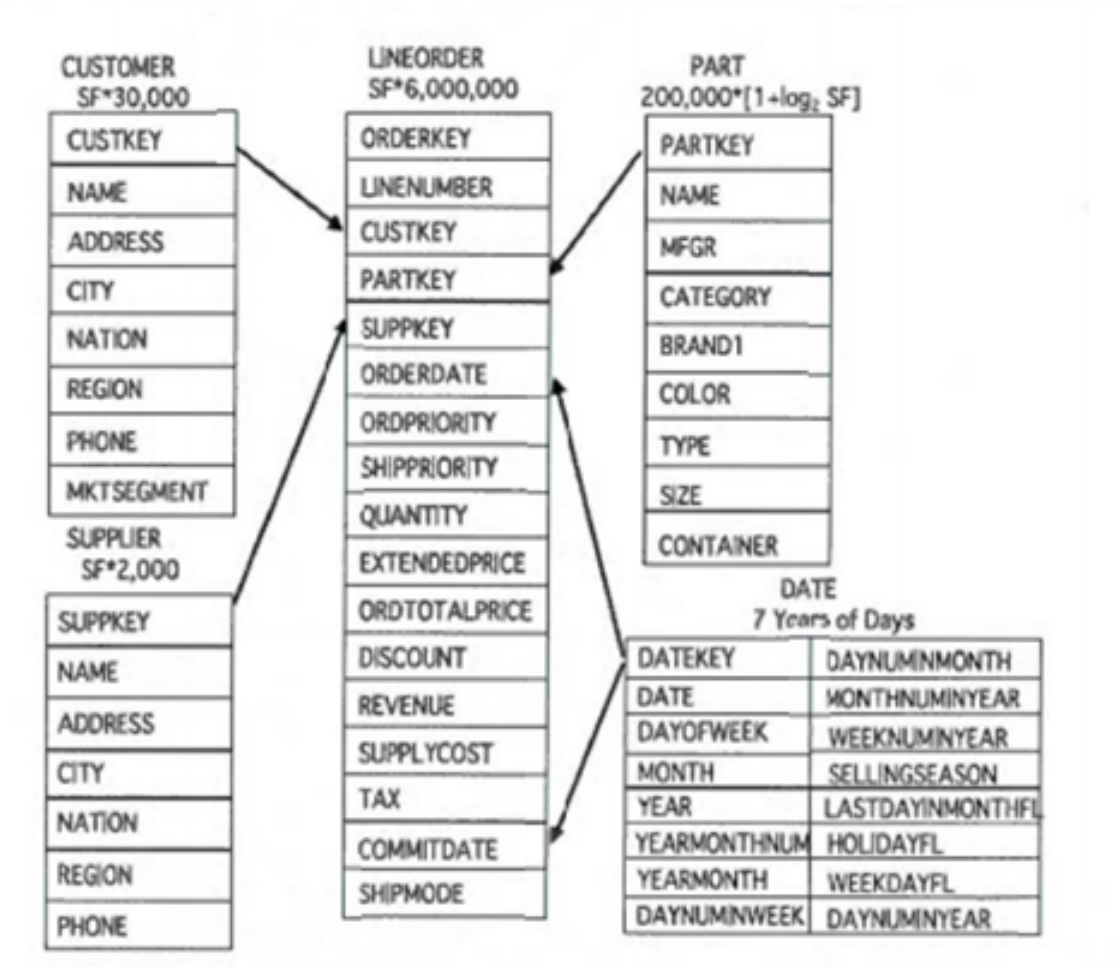

FIGURE 4.1 – Schéma en étoile de l'entrepôt expérimental

table	Type	Cardinalité
Lineorder	Faits	6.000.000*SF
Supplier	Dimension	2.000*SF
Custumer	Dimension	30.000*SF
Part	Dimension	200.000*(1+log2SF)
Dates	Dimension	2556

FIGURE 4.2 – La taille des tables SSB en termes d'instancesl

Processus chargement Afin de charger le benchmark SSB, nous avons suivi les étapes décrites dans la figure ci-dessous :

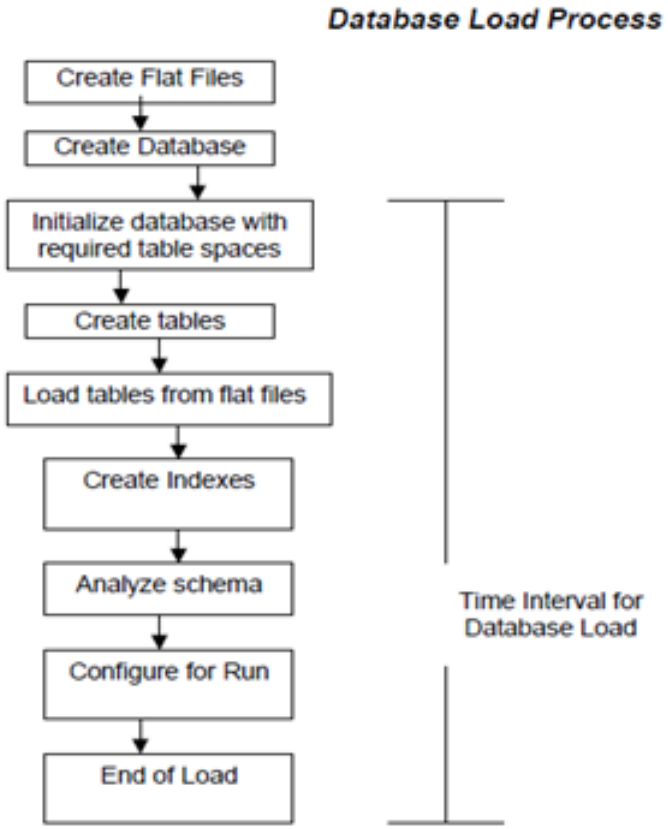

FIGURE 4.3 – processus de chargement du Benchmark SSB

4.2.3 Présentation de notre application

Conception

Nous présentons dans cette section la conception de notre outil. Nous commençons par présenter les objectifs attendus du développement de cet outil. Nous évoquons ensuite l'utilisation de modèle de tâches que nous avons effectuées pour concevoir l'outil et son interface. Nous terminons par présenter l'architecture fonctionnelle de notre outil.

Objectifs

Notre outil gère principalement deux techniques de transformation ; la fragmentation et le fusionnement.Les objectifs principaux sont : – Permettre la visualisation de l'état courant de ces données de ce modèle sources : la structure de MS (schéma, attributs, taille de chaque table, définition de chaque attribut, etc.) et la charge définie sur l'entrepôt (le nombre de sélections, le nombre de jointures, attributs de sélection, les facteurs de sélectivité de chaque prédicat, etc.) – Offrir deux modes d'évaluation possible : personnalisée et non personnalisée. Dans la sélection non personnalisée, nous supposons que le concepteur n'a pas assez de connaissances sur les actions effectuées. L'outil fait alors le choix (par défaut) et propose la solution générée par cet outil. Tandis que la sélection personnalisée donne plus de liberté à l'administrateur pour le choix de l'algorithme, des paramètres, des attributs et des tables sur lesquels il souhaite sélectionner la technique d'optimisation.

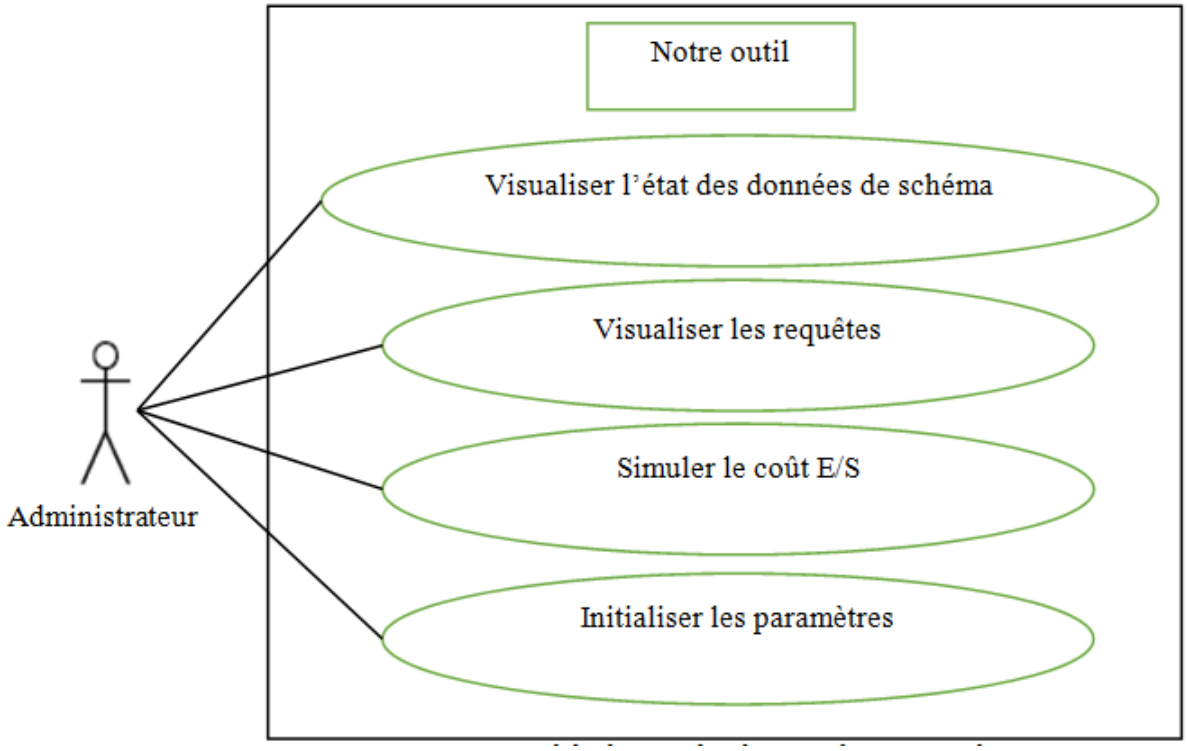

FIGURE 4.4 – Le modèle des cas d'utilisation de notre outil

Conception de l'outil

Après avoir exprimé et analysé les besoins de l'administrateur dans les sections précédentes, nous présentons dans cette partie l'étude conceptuelle de l'outil qui consiste à mettre en place un diagramme de classes d'UML, qui servira de fondation pour le développement et la réalisation de l'outil.

Diagramme de Classes Notre modèle de conception schématisé par le digramme de classe d'UML, qui servira de fondation pour le développement et la réalisation de l'outil. La figure 4 illustre notre modèle générique que nous avons proposé.

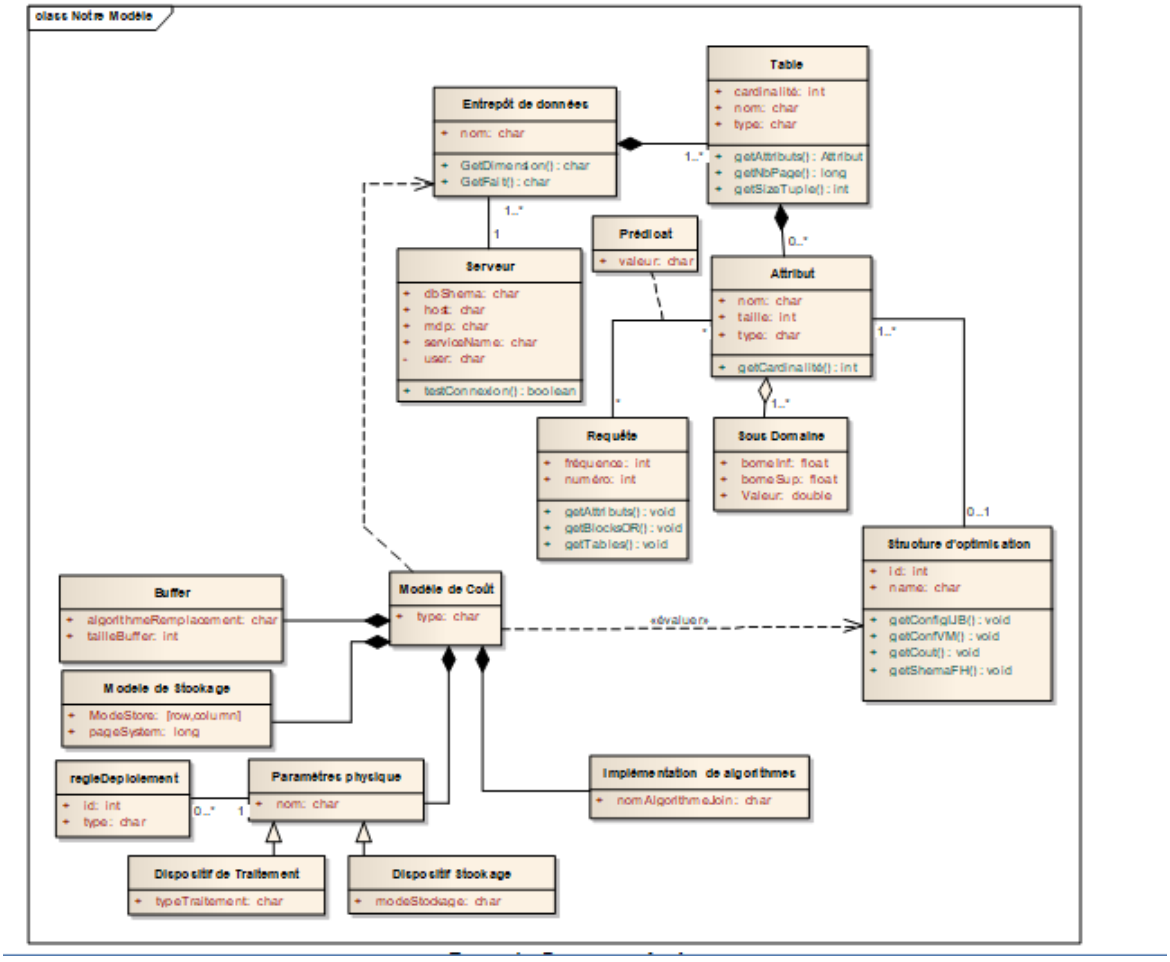

FIGURE 4.5 – Digramme de classes

Description des classes : Le tableau ci-dessous décrit les différentes classes composant le volet conceptuel de notre outil.

Nom de la classe	Description
SGBD	Contient les informations concernant les SGBD : noms, mot de passe, URL et le driver de connexion ; elle contient des opérations comme la possibilité d'ajout ou de suppression d'un SGBD.
User	L'utilisateur exploitant notre système est l'administrateur ayant comme privilèges de manipuler la base ou l'entrepôt de données, l'utilité de l'identification de cette classe se voie lors de l'authentification de l'administrateur et la connexion.
Entrepôt de données	Représente l'entité l'entrepôt de données en étoile.
Table	Représente l'entité table d'un entrepôt de données, cette classe regroupe les propriétés et les traitements d'une table, chaque instance correspond à une dimension ou un fait de l'entrepôt de données.
Attribut	Représente l'entité attribut d'une table d'un entrepôt de données.
Sous Domaine	Sous ensemble de valeurs ou un intervalle d'un attribut.
Prédicat	Un prédicat est constitué d'un attribut et un sous ensemble de valeurs ou un intervalle.
Requête	Représente l'entité requête de type SELECT sur un entrepôt de données. Elle contient les informations telles que le numéro de requête ou la fréquence.
Charge de requêtes	C'est une abstraction de l'ensemble de requêtes que soumet l'administrateur. Cette classe contient des méthodes permettant l'extraction et l'enregistrement de toutes les informations concernant les requêtes.
Algorithme jointure	Décrit l'implémentation des politique des jointures.
Buffer	Représente l'abstraction de la mémoire tampon.
Structure optimisation	décrit la configuration physique des trois techniques d'optimisation sur l'entrepôt de données : abstraction de l'entité ensemble des fragments abstraction de l'entité configuration d'Index de jointure binaire
Dispositif stockage	Représente l'abstraction des mémoires HDD, SSD et RAM.
Dispositif traitement	Représente l'abstraction de CPU.
Règles déploiement	Représente les scénarii d'implémentation du support de stockge

TABLE 4.1 – Description des classes

4.3 Présentation de l'outil réalisé

Nous présentons dans ce qui suit la présentation de notre outil développé et un scénario d'utilisation.

4.3.1 Interface d'accueil

Visualiser un entrepôt de données

Après le lancement de l'outil le concepteur se connecte au SGBD Oracle et récupère certaines informations telles que : le schéma de modèle source, par ex. les tables, les attributs des tables.

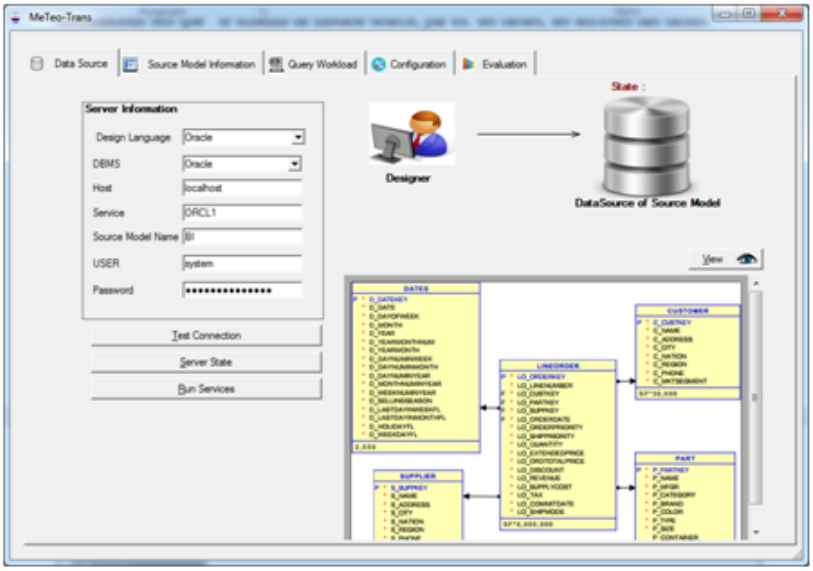

FIGURE 4.6 – Visualiser les données du modèle source

Chargement de requêtes

Le concepteur permet également de spécifier la charge de requêtes à estimer, à partir d'un emplacement sur disque.

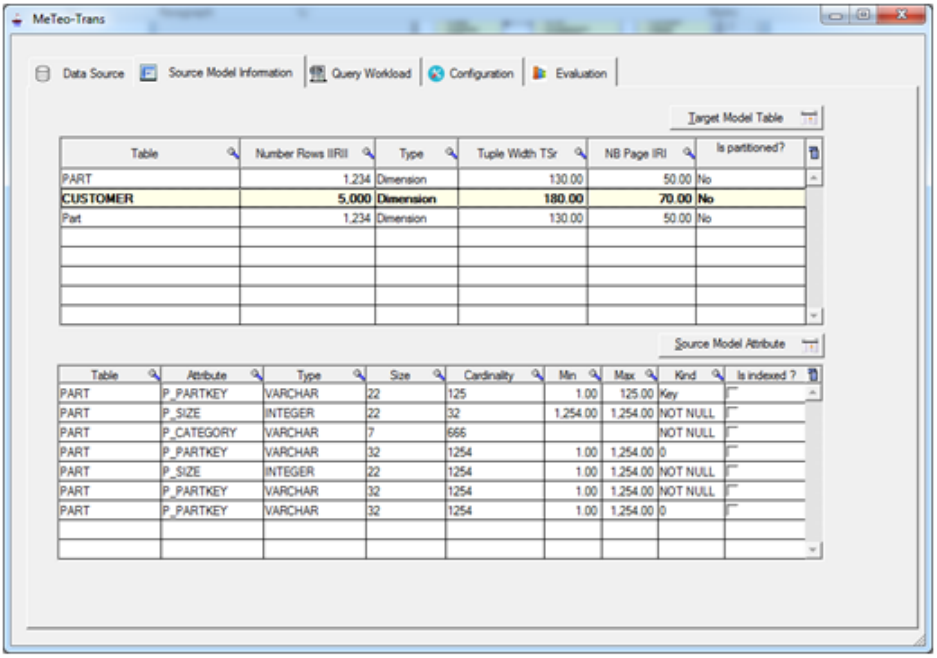

FIGURE 4.7 – Visualiser les statistuqeys des objets du modèle source

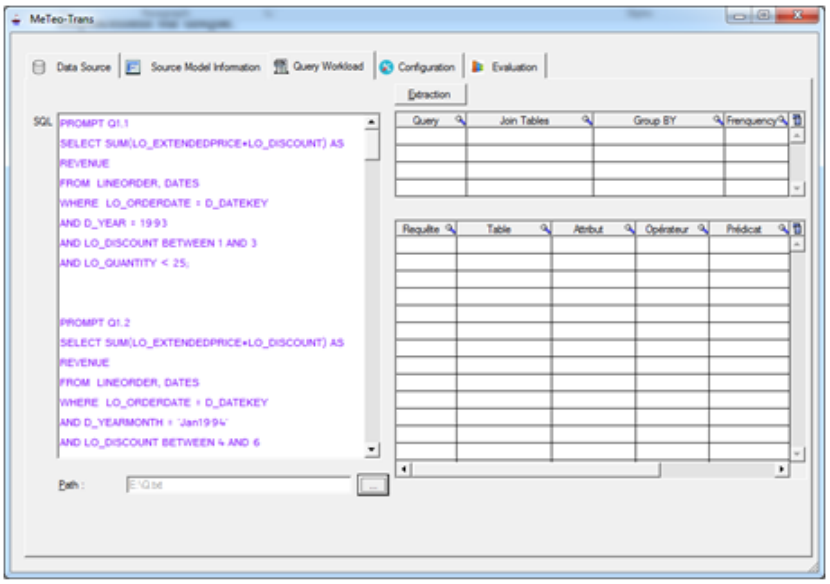

FIGURE 4.8 – Chargement de requêtes

Initialisation des paramètres

Pour lancer la sélection d'un modèle cible optimale, le concepteur doit introduire les paramètres systèmes (nombre de fragments) et les paramètres de l'algorithme génétique . Apés la configuration, le concepteur lance le processus de sélection

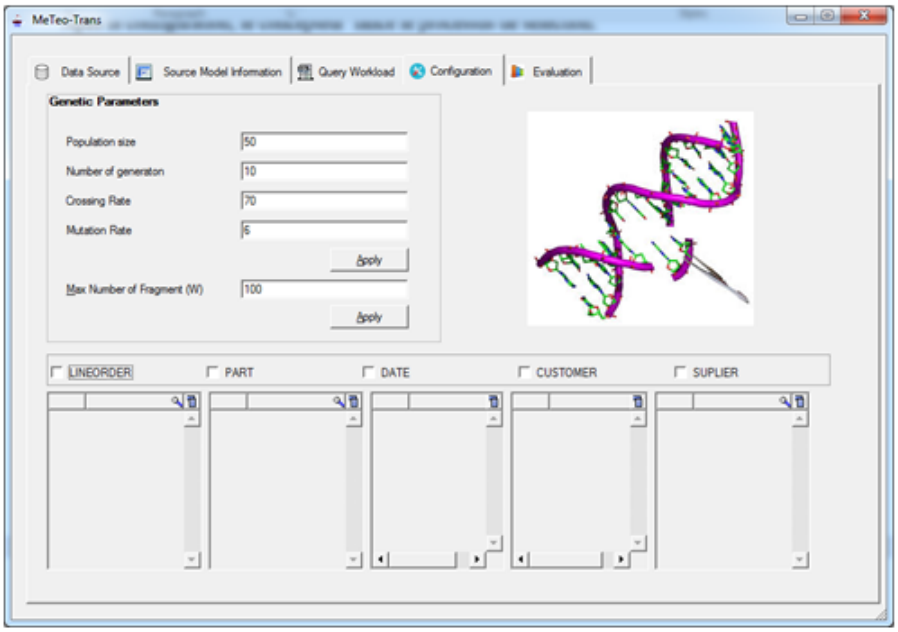

FIGURE 4.9 – Initialisation des paramètres (Personnalisation de la configuration)

Visualisation de résultats

L'outil permet de visualiser les résultats (figure 4.10), il montre les actions effectués sur le modèle source avec le coût en terme d'entre sortie disque et le scipt de materialisation.

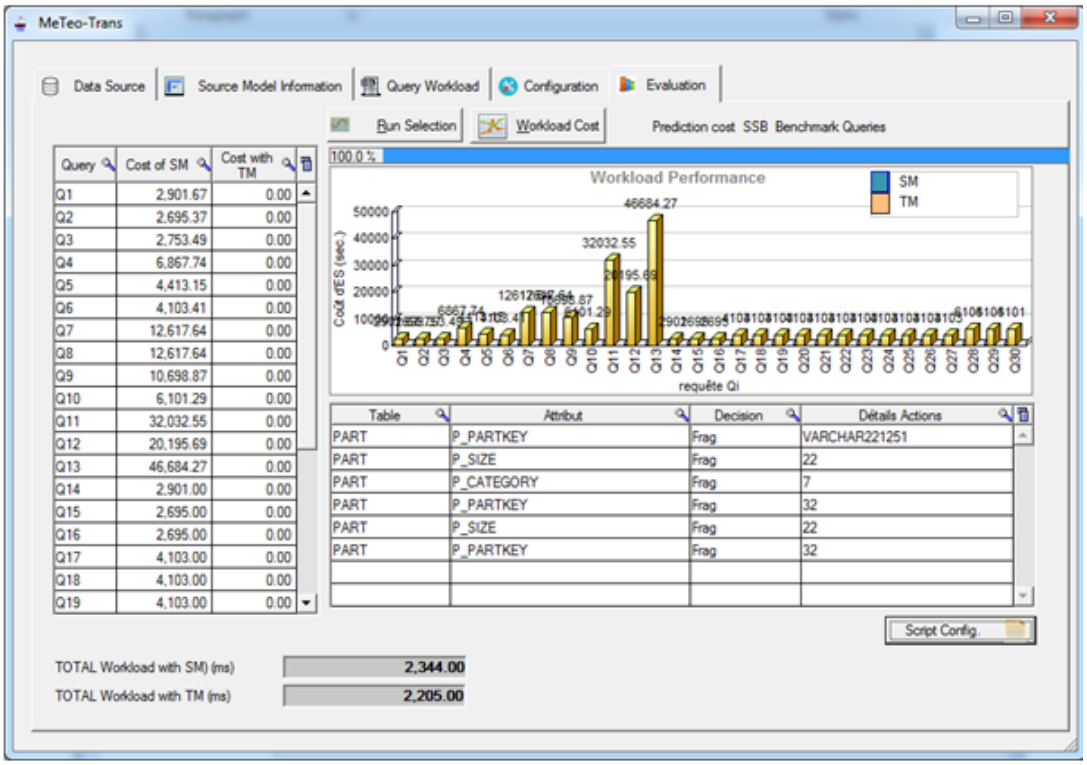

FIGURE 4.10 – Visualisation de résultats

4.3.2 Conclusion

Dans ce chapitre, nous avons présenté notre outil développé, ce dernier permettant d'assister les concepteurs dans leur tâche pour simuler le comportement de la requête selon différents schémas de transformation cible. Il offre des interfaces conviviales et simples pour permettre aux administrateurs de naviguer et de visualiser l'état de son schéma source.

Conclusion Générale et perspective

Dans notre travail, nous nous sommes intéressés à la transformation des modèles en prenant en considération une certaine qualité de service (QoS) comme les performances des requêtes décisionnelles exécutées sur un modèle de données relationnel. Pour cela, nous avons proposé une nouvelle approche de sélection de modèle cible basée sur l'algorithme génétique. Notre approche est basée sur deux types d'action « split merge » où chacun est responsable d'une tâche dans le processus d'optimisation de réduire le coût d'exécution des requêtes.

Nous avons présenté et discuté dans l'étude de l'implémentation de l'outil prototype que nous avons mené pour montrer la faisabilité de l'approche proposée. Les résultats de ce test sont encourageants et montrent la faisabilité de notre approche. Néanmoins, dans le but d'avoir de meilleures performances, des améliorations peuvent être apportées.

En conclusion, nous pouvons résumer notre contribution aux points suivants :

1. modélisation de problèmes de transformation comme un problème d'optimisation sous contraintes.

2. Proposition d'une démarche de transformation dirigée par une adaptation de l'algorithme génétique : trois primitives principales caractérisent cette approche, l'optimisation locale et l'optimisation globale qui prend en charge l'impact inter solution. L'optimisation globale vise à minimiser le coût d'exécution total de la charge de requêtes.

3. Implémentation du système via un outil supportant l'approche proposée.

Perspectives

Le travail présenté dans ce mémoire ouvre plusieurs perspectives de recherche. En effet, il serait intéressant d'étudier des améliorations qui peuvent être apportées à ce travail, nous pouvons citer :

- Réalisation d'une étude expérimentale pour montrer les qualités de l'approche proposée.- Considérer d'autres transformations comme par exemple la transformation de formalisme DEVES vers Les réseaux de Petri.

- Études comparatives avec d'autres algorithmes comme Hill climbing, etc.

Bibliographie

[1] Moussa Amrani, Jürgen Dingel, Leen Lambers, Levi Lúcio, Rick Salay, Gehan Selim, Eugene Syriani, and Manuel Wimmer. Towards a model transformation intent catalog. In *Proceedings of the First Workshop on the Analysis of Model Transformations*, pages 3–8, 2012.

[2] Jasmina Arifovic. Genetic algorithms and inflationary economies. *Journal of Monetary Economics*, 36(1) :219–243, 1995.

[3] Islem Baki, Houari Sahraoui, Quentin Cobbaert, Philippe Masson, and Martin Faunes. Learning implicit and explicit control in model transformations by example. In *International Conference on Model Driven Engineering Languages and Systems*, pages 636–652. Springer, 2014.

[4] Jean Bézivin. On the unification power of models. *Software & Systems Modeling*, 4(2) :171–188, 2005.

[5] Jean Bézivin, Frédéric Jouault, Peter Rosenthal, and Patrick Valduriez. Modeling in the large and modeling in the small. In *Model Driven Architecture*, pages 33–46. Springer, 2004.

[6] X Blanc and MDA en Action. Ingénierie logicielle guidée par les modèles, 2005.

[7] Carl Friedrich Bolz, Adrian Kuhn, Adrian Lienhard, Nicholas D Matsakis, Oscar Nierstrasz, Lukas Renggli, Armin Rigo, and Toon Verwaest. Back to the future in one week—implementing a smalltalk vm in pypy. In *Workshop on Self-sustaining Systems*, pages 123–139. Springer, 2008.

[8] Imed Chouchani. Utilisation d'un algorithme génétique pour la composition de services web. 2010.

[9] Venegono Inferiore IT Colombo. Salehi et a. 2014.

[10] OMG CORBA and IIOP Specification. Object management group. *Joint revised submission OMG document orbos/99-02*, 1999.

[11] L Davis. handbook of genetic algorithms van nostrand reinhold new york. *DavisHandbook of Genetic Algorithms1991*, 1991.

[12] Samba Diaw, Redouane Lbath, and Bernard Coulette. Etat de l'art sur le développement logiciel basé sur les transformations de modèles. 2010.

[13] Sonia Dimassi, Mehdi Jemai, Bouraoui Ouni, and Abdellatif Mtibaa. Meta-heuristics : For the problem of partitioning hardware/software. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pages 1–3. IEEE, 2015.

[14] Xavier Dolques, Marianne Huchard, and Clémentine Nebut. Génération de transformation de modèles par application de l'arc sur des exemples. In *LMO : Langages et Modèles à Objets*, pages 61–75. Cépaduès Editions, 2009.

[15] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy. 1991.

[16] Kathryn A Dowsland. Modern heuristic techniques for combinatorial problems, chapter 2–simulated annealing. *Advanced Topics in Computer Sciences. Blackwell Scientific Publications*, pages 20–69, 1993.

[17] Nicolas Durand, Jean-Marc Alliot, and Joseph Noailles. Algorithmes génétiques : un croisement adapté aux fonctions partiellement séparables. *Journées' Evolutions Artificielles*, 94, 1994.

[18] Martin Faunes, Houari Sahraoui, and Mounir Boukadoum. Generating model transformation rules from examples using an evolutionary algorithm. In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pages 250–253. IEEE, 2012.

[19] Martin Faunes, Houari Sahraoui, and Mounir Boukadoum. Genetic-programming approach to learn model transformation rules from examples. In *International Conference on Theory and Practice of Model Transformations*, pages 17–32. Springer, 2013.

[20] Sanford Friedenthal, Alan Moore, and Rick Steiner. Omg systems modeling language (omg sysml) tutorial. In *INCOSE Intl. Symp*, volume 9, pages 65–67, 2006.

[21] Iván García-Magariño, Jorge J Gómez-Sanz, and Rubén Fuentes-Fernández. Model transformation by-example : An algorithm for generating many-to-many transformation rules in several model transformation languages. In *International Conference on Theory and Practice of Model Transformations*, pages 52–66. Springer, 2009.

[22] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5) :533–549, 1986.

[23] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. 1988.

[24] MDA Guide. Version 1.0, object management group, omg. Technical report, on 01-05-2003, 2003.

[25] Julia Handl and Bernd Meyer. Improved ant-based clustering and sorting in a document retrieval interface. In *International Conference on Parallel Problem Solving from Nature*, pages 913–923. Springer, 2002.

[26] Jin-Kao Hao, Philippe Galinier, and Michel Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle*, 13(2) :283–324, 1999.

[27] JH Holland. Adaptation in natural and artificial systems, univ. of mich. press. *Ann Arbor*, 1975.

[28] Marouane Kessentini, Houari Sahraoui, Mounir Boukadoum, and Omar Ben Omar. Search-based model transformation by example. *Software & Systems Modeling*, 11(2) :209–226, 2012.

[29] Marouane Kessentini, Manuel Wimmer, Houari Sahraoui, and Mounir Boukadoum. Generating transformation rules from examples for behavioral models. In *Proceedings of the Second International Workshop on Behaviour Modelling : Foundation and Applications*, pages 1–7, 2010.

[30] Jan Karel Lenstra. Clustering a data array and the traveling-salesman problem. *Operations Research*, 22(2) :413–414, 1974.

[31] Khaled Mesghouni. *Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production*. PhD thesis, Lille 1, 1999.

[32] Z Michalewicz. Genetic algorithms+ data structures= evolution programs, 3rd edn.© springer, 1996.

[33] Zbigniew Michalewicz and Cezary Z Janikow. Handling constraints in genetic algorithms. In *Icga*, pages 151–157, 1991.

[34] OMG MOF. Meta object facility (mof) core specification omg available specification version 2.0, 2006.

[35] Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, and Philippe Vanderheyden. Eclipse development. *Using the Graphical Editing Framework and the Eclipse Modeling Framework. IBM RedBooks.(IBM Corp.)*, 379, 2004.

[36] Robert Ornduff and Darwin. Darwin's botany. *Taxon*, pages 39–47, 1984.

[37] Christos H Papadimitriou. Complexity theory. *Reading : Addison Wesley*, 1994.

[38] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1998.

[39] Ajith Kumar PJ and Shijo Thomas. A geographical based facility location approach to ensure efficient supply chain management.

[40] Michael Strommer, Marion Murzek, and Manuel Wimmer. Applying model transformation by-example on business process modeling languages. In *International Conference on Conceptual Modeling*, pages 116–125. Springer, 2007.

[41] Michael Strommer and Manuel Wimmer. A framework for model transformation by-example : Concepts and tool support. In *International Conference on Objects, Components, Models and Patterns*, pages 372–391. Springer, 2008.

[42] MR Tyrrell and JHN Wolfe. New prosthetic venous collar anastomotic technique : combining the best of other procedures. *British journal of surgery*, 78(8) :1016–1017, 1991.

[43] Dániel Varró. Model transformation by example. In *International Conference on Model Driven Engineering Languages and Systems*, pages 410–424. Springer, 2006.

[44] Manuel Wimmer, Michael Strommer, Horst Kargl, and Gerhard Kramler. Towards model transformation generation by-example. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 285b–285b. IEEE, 2007.

[45] Nobukazu Yoshikawa and Toyohiko Yatagai. Phase optimization of a kinoform by simulated annealing. *Applied optics*, 33(5) :863–868, 1994.

[46] Xiao Zhang, Sylvie Renaud, and Michael Paice. Cellulase deinking of fresh and aged recycled newsprint/magazines (onp/omg). *Enzyme and microbial technology*, 43(2) :103–108, 2008.