République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER RECHERCHE

En
Informatique

Option
Intélligence Artificielle

Thème

Etude comparative entre les méthodes de décomposition structurelles et les méthodes de décomposition sémantiques pour la résolution des CSPs

Réalisé par : Mlle. KHIATI Yamina

présenté le 10/10/2021 devant le jury composé de :

Encadrant	**Pr Amoun Kamal**	**Professeur**	**U. A/Mira Béjaia.**
Examinateur	**Dr Aloui Soraya**	**Maître de conf. A**	**U. A/Mira Béjaia.**
Examinateur	**Dr Mehaoued Kamal**	**Maître-assistant A**	**U. A/Mira Béjaia.**

Béjaia, Septembre 2021.

⁂ *Remerciements* ⁂

Grâce à Dieu vers lequel vont toutes les louanges, ce travail s'est accompli.
Je tiens tout d'abord à remercier sincèrement M. Amroun kamal pour son encadrement et pour le soutien qu'il m'a apporté pour l'accomplissement de ce travail de fin d'étude et surtout pour ses qualités professionnelle et humaine.
Je tiens, également, à remercier vivement les membres du jury pour l'honneur qu'ils m'on fait en acceptant de juger ce modeste travail. J'amerai également remercier ma familles et tous mes amis et camarades pour leur soutien et aide qui m'en donné la force d'aller toujours vers l'avant.
Je souhaite témoigner mon gratitude et vifs remerciements envers tous ceux qui ont contribué, de prés ou de loin à la réalisation de ce travail.

⁕ *Dédicaces* ⁕

Je dédie ce modeste travail aux :
Personnes les plus chéres dans mes yeux, mes parents.
ma chere soeur Fatma et frére Anis.
A toutes ma famille.
A mes sincéres amis, copines (Djidji, Sylia).
Tous ceux qui sont proches de mon coeur et dont je n'ai pas cité leurs noms.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les problèmes de satisfaction de contraintes (CSP) sont au coeur de nombreuses applications en intelligence artificielle et recherche opérationnelle, telles que l'allocation de ressources, la planification, l'ordonnancement. Les Problèmes de satisfaction de contraintes sont un cadre générique introduit par Montanari dans les années 70 et qui permettent de formaliser un très grand nombre de problèmes tels que les problèmes de coloration de graphe, les problèmes d'ordonnancement, les problèmes de placement,etc.
Un CSP est défini comme étant un ensemble de contraintes impliquant un certain nombre de variables. L'objectif consiste simplement à trouver un ensemble de valeurs à affecter aux variables, de sorte que toutes les contraintes soient satisfaites. La recherche dans le domaine des CSP est motivée par le besoin de concevoir des méthodes efficaces pour les résoudre. Dans un CSP, le but est de trouver une solution qui satisfasse toutes les contraintes. Mais la plupart des CSP appartiennent à une classe de problèmes appelés NP-complets pour lesquels tous les algorithmes connus nécessitent, dans le pire des cas. un temps exponentiel. En essayant de diminuer le coût de la résolution d'un CSP, plusieurs méthodes ont été proposées par des chercheurs.
La méthode de résolution de base est le Backtrack standard qui schématiquement procéde à l'énumération de toutes les affectations possibles jusqu'a trouver une solution. Dans le pire des cas, cette méthode peut explorer la totalité de l'espace de recherche dont la taille est bornée par d^n avec d la taille maximale des domaines du probléme et n le nombre de variables. Elle peut donc s'avérer trés couteuse. Elle se révèle totalement inefficace en pratique à cause du grand nombre de redondances inhérentes à cette technique. Plusieurs voies ont été proposées pour l'améliorer que ce soit en pratique ou en théorie : les améliorations effectives de la technique du backtracking et la décomposition des problèmes. Le filtrage permet de réduire la taille du problème ou de le modifier de sorte à construire un probléme équivalent mais généralement plus facile à résoudre. Il existe differents degrés de filtrage, plus ou moins efficaces et plus ou moins couteux, qui se basent sur les notions de cohérence locale ou générale et de propagation de contraintes. Associées à des techniques de retour en arriére non chronologique, de consistance avant et d'apprentissage, ces améliorations ont mené à des méthodes telles que FC et MAC qui sont bien plus efficaces en pratique, mais dont la complexité théorique demeure en $O(md^n)$ où m est le nombre de contraintes et d est la taille maximale des domaines des variables. Pour réduire cette complexité, plusieurs méthodes

de décomposition structurelles ont été proposées. Chaque méthode de décomposition défénit son propre concept de largeur qui peut être interprétée comme une mesure de cyclicité du graphe ou de l'hypergraphe de contraintes. Parmi ces méthodes, on retient principalement : Cycle cut-set, Hypercutset, Biconnected components, Tree décomposition, Hinge décomposition, Hypertree décomposition, Generalized Hypertree décomposition. Ajoutant à ces méthodes, d'autre nouvelles méthodes ont été proposé pour une complexité meilleure par rapport à la complexité des méthodes de decompositions structurelles à savoir, la technique de décomposition de domaines guidée par la sémantique des contraintes de distance. cette décomposition sémantique, que nous l'avos nommée SDD (Semantic Domain Decomposition), isole les disjonctions dans l'espace de recherche. Les domaines des coordonnées d'un même point sont décomposés et réduits par la SDD en utilisant les propriétés de monotonie et de convexité des contraintes de distance. Nous montrons comment cette technique peut être combinée avec différents filtrage locaux.
l'intérêt de cette technique est illustrée sur des problèmes de satisfaction de contraintes de distance. Ce type de contraintes intervient dans de nombreux domaines d'application comme la conception optimale de robots et plus généralement les problèmes de modélisation géométrique, les problèmes de conformation moléculaire.
Une autre méthodes de résolution, est la résolution des CSPs conservatrice où on les a dévisé en deux réduction, la résduction d'exclusion en tant que composant(as-composant), où on convertit le probléme en certain nombre d'instance CSP dans lesquels chaque domaine est un as-composant, et alors soit fournit une solution, soit permet d'éléminer certains éléments de certains des domaines d'origine. Et la deuxiéme réduction est la réduction de Maroti qui est applicable aux instance, dans lesquels tous les domaines sont des as-composant.
Dans ce mémoire, nous allons comparer différentes méthodes.
Le manuscrit est structuré comme suit :
Chapitre 1 : nous allons présenter les généralitées sur les probléme de satisfaction de contrainte où on a précisé les différentes méthodes de résolution.
Chapitre 2 : nous allons présenter les différentes méthodes de décomposition structurelles présenté dans la littérature.
Chapitre 3 : nous allons parler des différentes méthodes de décomposition sémantique.
Chapitre 4 : nous allons faire une comparaison concernant les méthodes vu dans les deux chapitres précedents.

Chapitre 1

Les problémes de satisfaction de contraintes

1.1 Introduction

Les problémes de satisfaction de contrainte CSP(constraint satisfaction problems) constituent un formalisme important de l'intelligence artificielle pour exprimer et résoudre les problémes réels. Le principe des CSP a été introduit par Montanari. Un réseau de contraintes (CSP) est constitué d'un ensemble de variable X, dont chacune doit être affecter à une valeur essue de son domaine associé [10].

Définition 1.1.1 (Instance CSP) Un probléme de satisfaction de contraintes $P = (X, D, C)$ est définie par :
- Un ensemble de variable $X = x_1, ..., x_n$.
- Un ensemble de n domaines finis et discrets $D = D_{x1}, ..., D_{xn}$ où D_{xi} est le domaine associé à la variable x_i.
- Un ensemble de contraintes $C = c_1, ..., c_n$ où c_i est défini sur un ensemble de variables x_i.
- Un ensemble de relations de $R = R_1, ..., R_n$, où R_i est l'ensemble des combinaison de valeurs qui satisfont la contrainte c_i, il s'agit de la relation associé à c_i.
R_i est un sous ensemble du produit cartisien $D_{x1}, ..., D_{xn}$.
- L'arité d'une contrainte et le nombre de variables sur lesquelles elle porte, on dit que la contrainte est :
- Unaire si son arité est égale à 1.
- Binaire si son arité est égale à 2.
- N-aire si son arité est égale à n [8].

Définition 1.1.2 (Instanciation)
Soit un CSP (X, D, C), on appelle l'instanciation le fait d'affecter certains variables par des

valeurs (prises dans leurs domaines respectifs) elle est notée : $x \leftarrow v$.

Définition 1.1.3 (Instanciation consistante)
Soit $P = (X, D, C, R)$ et $y \subset X$, une instanciation A sur Y est consistante si :
$\forall c_i \in C, c_i \subset Y, A[c_i] \in R_i$ [9].
Définition 1.1.4 (Solution)
Une solution d'un CSP $P = (X, D, C)$ est une instanciation de $s \in S$ qui satisfait toutes les contraintes [11].

1.1.1 *Représentation graphique*

- A chaque CSP $P = (X, D, C)$ est associé à un graphe de contraintes obtenue en représentant chaque variable par une aréte entre x_i et x_j.
- Ce graphe est appelé graphe de containte. En général une instance CSP quelconque (n-aire) est représenté par ; un hypergraphe de contrainte dont les sommets sont les variables et il existe une hyperaréte $X_i, ..., X_k$, si ce dernier ensemble est la porté d'une certaine contrainte [1].

1.1.2 *Exemples d'instances de CSP*

Dans cette partie nous allons cité quelques exemples d'instances de CSP :

Exemple 1 (Les 4 reines)
Le probléme des 4 reines s'énnonce comme suit :" étant donné un échiquier de taille 4", placer 4 reines sur cet echiquier de sorte que : deux reines ne peuvent pas étre placées sur une même ligne, une même colonne ou une même diagonale.

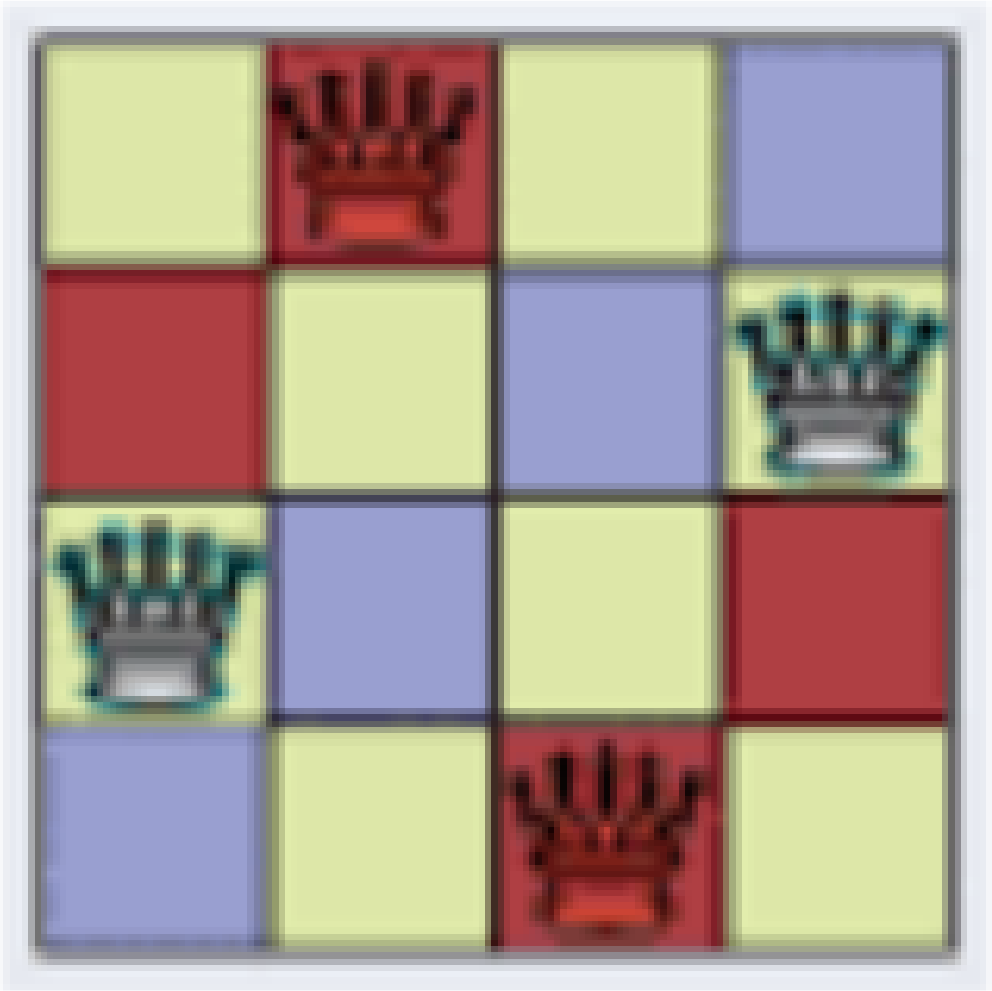

FIGURE 1.1 – Exemple des 4 reines

- Modélisation du probléme :

on definit le probléme $P_{4-reines} = (X, D, C, R)$ avec :

- $X = X_1, X_2, X_3, X_4$.
- $D = D_1, D_2, D_3, D_4$.
- $C = C_1, C_2, C_3, C_4$ avec :

$C_1 = X_1, X_2, R_1$.
$C_2 = X_1, X_3, R_2$.
$C_3 = X_1, X_4, R_3$.
$C_4 = X_2, X_3, R_4$.
$C_5 = X_2, X_4, R_5$.
$C_6 = X_3, X_4, R_6$.

- $R = R_1, R_2, R_3, R_4, R_5, R_6$ avec :

$R_1 = (1,3);(1,4);(2,4);(3,1);(4,1)(4,2)$
$R_2 = (1,2);(1,4);(2,1);(2,3);(3,2);(3,4);(4,1);(4,3)$
$R_3 = (1,2);(1,3);(2,1);(2,3);(2,4);(3,1);(3,2);(3,4);(4,2);(4,3)$
$R_4 = (1,3);(1,4);(2,4);(3,1);(4,1);(4,2)$
$R_5 = (1,2);(1,4);(2,1);(2,3);(3,2);(3,4);(4,1);(4,3)$
$R_6 = (1,3);(1,4);(2,4);(3,1);(4,1);(4,2)$
Une solution à ce probléme est : $X_1 = 2$, $X_2 = 4$, $X_3 = 1$, $X_4 = 3$.

Exemple 2 (Coloriage de carte)
Le probléme de coloriage de graphe avec 3 couleurs consiste à colorier toutes les régions d'une carte, où chaque région doit être colorier avec une des trois couleurs disponible de telle sorte que deux régions voisine n'aient pas la même couleur.

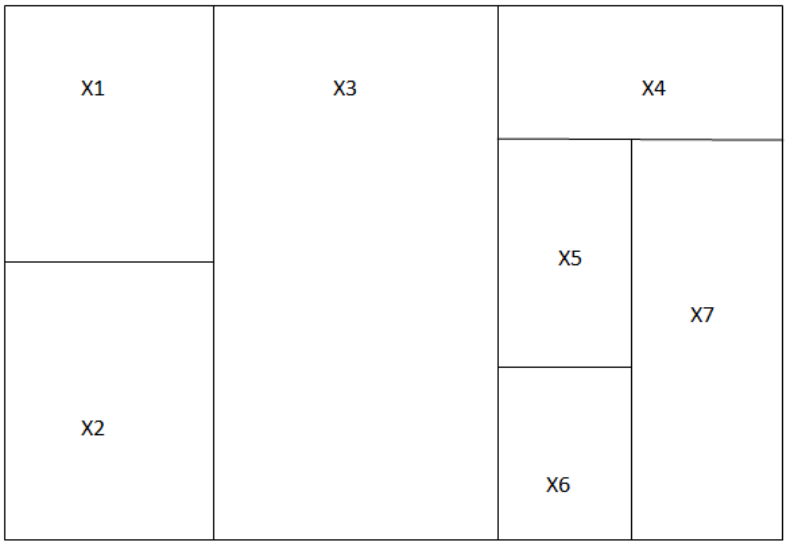

FIGURE 1.2 – Exemple de 3-coloration

- Modilisation du probléme :
- Les variables du probléme sont les 7 régions à colorier : $X_1, X_2, X_3, X_4, X_5, X_6, X_7$.
- Les domaines sont : D_1,D_2,D_3,D_4,D_5,D_6,D_7 telles que $\forall i \in 1,2,3,4,5,6,7$, $D_i = Jaune, rouge, noire$.
-Les contraintes sont :$C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}$ teslles que :
$C_1 = X_1, X_2$.
$C_2 = X_1, X_3$.
$C_3 = X_2, X_3$.
$C_4 = X_3, X_4$.
$C_5 = X_3, X_5$.
$C_6 = X_3, X_6$.
$C_7 = X_4, X_5$.

$C_8 = X_4, X_7$.
$C_9 = X_5, X_7$.
$C_{10} = X_5, X_6$.
$C_{11} = X_6, X_7$.
- Les relations sont : $R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}, R_{11}$ telles que : $\forall i \in 1...11$;
$R_i = (Rouge, Jaune), (Rouge, Noire), (Jaune, Rouge), (Juane, Noire), (Noire, Rouge), (Noire, Juane)$
Une solution à ce probléme est :
$X_1 = Noire$, $X_2 = Juane$, $X_3 = Rouge$,$X_4 = Noire$, $X_5 = Jaune$, $X_6 = Noire$, $X_7 = Rouge$.

1.2 *Les méthodes de résolution*

La solution efficace des problémes de satisfaction des contraintes a été pour de nombreux ans un objectif important de la recherche sur l'IA. Dans cette séction nous allons détaillés ces méthodes.

- Les méthodes complétes La plupart des algortithmes de recherche complet sont basés sur l'algorithme Backtrack standard.

- Les méthodes incomplétes Les méthodes incomplétes reposent sur des heuristique permettant d'etudier des zones spécifiques de l'espace de recherche dans le but d'atteindre une solution.
Ces methodes peut donner une réponse acceptable en un temps raisonnable, mais elles sont incapable de donner un résultat complet, car l'espace des solutions ne sera pas totalement exploré.

- Les méthodes hybrides

Les méthodes hybrides représentent une hybridation des méthodes complétes et les méthodes incompléte, plusieurs algorithmes hybrides ont été proposés : combinaison d'une méthode énumérative et une tree décomposition, etc.

- Les algorithmes de filtrages

l'idée de cette téchnique est de transformer un CSP en un autre CSP dont l'espace de recherche est réduit par rapport a celui du CSP d'origine, mais dont l'espace de solutions est le même.
Cette technique évitera à une procdure de recherche de parcourir certaines zones de l'espace de recherche qui méneraient sûrement à un echec.

- Les méthodes se décomposition structurelles

Ces téchniques tirent profit de la structure de probléme pour améliorer la résolution. Cette structure est capturée à travers une représentation du probléme en graphe ou l'hypergraphe de contrainte.
Afin d'appliquer la technique de décomposition, il faut que le CSP soit acyclique et arc consistant (qu'on va voir dans (1.2.1.2), et qu'il existe un algorithme capable de résoudre les instances CSP en temps polynomiale [12].

1.2.1 *Filtrage*

Le filtrage permet de réduire la taille du probléme ou de le modifier de sorte à construire un probléme equivalent moins plus facile à résoudre. Il existe des différents degrésde filtrafe qui se basent sur les notions de consistance locales ou générale et de propagation de contraintes.
La premiére stratégie consiste à supprimer des domaines, les valeurs inconsistantes. Une valeur est inconsistante si elle ne participe pas à aucune solution du probléme et de ce fait sa suppression conduit à un CSP équivalent de plus petite taille. Il aussi possible de supprimer des tuples au niveau des relations dont il est sûr qu'ils ne ménent pas à une solution [9].
La propriété de la consistance est atteinte lorsque aucun tuple d'une relation ou valeur d'un domaine ne pourra étre supprimer. Plusieurs formes de consistance sont définit en fonction des caractéristiques des contraintes : Consistance de nœud, consistance d'arc, consistance du chemin, K-consistance.

1.2.1.1 *Consistance de nœud NC*

On dit qu'une instance CSP est consistante de nœud, si pour chaque variable x de X, toute contrainte unaire partant sur x coincide avec le domaine de x. L'idée est de supprimer l'ensemble des valeurs ne satisfisant pas la contrainte [26].
- Exemple de CSP non consistant de nœud
Si C contient la contrainte "$y > 3$", et si le domaine de Y contient les valeurs 1,2,3,4,5,6, alors le CSP n'est pas consistant de nœud.
pour qu'il sois consistant de nœud, il faut enlever du domaine de Y les valeurs 1,2,3 qui violent la contrainte "$y > 3$".

1.2.1.2 *Consistance d'arc AC*

Un CSP (X, D, C) est consistant d'arc si pour tout couple de variable (x, y) de X et pour tout valeur $a \in D_x$, il existe une valeur $b \in D_y$; telles que l'affectation partielle $\{x = a, y = b\}$ satisfasse toutes les contraintes binaire de C. L'idée est de supprimer des valeurs des domaines des variables qui satisfont pas lapropriété de consistance d'arc [27].
Exemple de consistance d'arc
Considérons le CSP ne contenant que la contrainte $x < y$ avec $D_x = [5, ..., 10]$ et $D_y = [3, ..., 7]$.
Ce CSP n'est pas arc consistant car, si on considére la valeur 8 pour x, aucune valeur b dans D_y nous permet d'obtenir $8 < b$.
L'algorithme qui enleve les valeurs des domaines des variable d'un csp jusqu'à ce qu'il soit consistant d'arc s'appelle Arc consistancy AC.

1.2.1.3 *Consistance de chemin PC*

La consistance de chamin est une propriété simialire à la cohérence d'arc, mais prend en compte des paires de variable au lieu d'une seule.Une paire de variable est cohérente avec une troisiéme variable, si chaque évaluation cohérente de la paire peut-être étendue à l'autre variable de ma,iére à ce que toutes les contarintes binaires soient satisfaites.
formellement, x_i et x_j sont des chemins consistants avec x_k si, pour chaque paire de valeurs (a, b) qui satisfait la contrainte binaire entre x_i et x_j, il existe une valeur c dans le domaine de x_k, telle que (a, c) et (a, b) satisfont la contrainte entre x_i et x_k et entre x_j et x_k respectivement.
Exemple de consistance de chemin

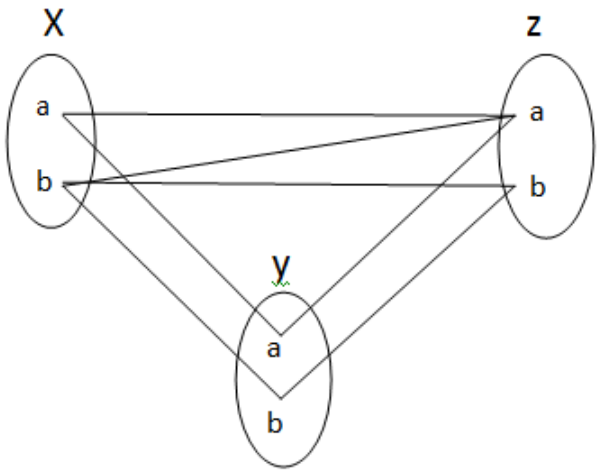

FIGURE 1.3 – Exemple de P n'est pas en pc

P n'est pas PC car le couple de variable x, z n'est pas PC.
En effet, pour l'instanciation $I = (x, b), (z, a)$, il n'existe pas de valeur dans le domaine de la variable y qui compatible avec cette instanciation [13].

1.2.1.4 *k-consistance*

Un CSP est k-consistance si toute affectation consistante sur $k-1$ variable peut-être étendue en une instanciation consistante sur k-variable.
Un CSP est fortement k-consistant s'il est i-consistant pour tout $i < k$.
La 1-consistance correspond à la consistance de nœud, la 2-consistante correspond à la consistance d'arc alors que la 3-consistance correspond à la consistance de chemin.

1.2.2 *Filtrage des CSPs n-aires*

Dans cette séction, nous allons parler de l'arc consistance généralisée, l'interconsistance, puis la consistance relationnelle.

1.2.2.1 *Arc Consistance généralisée GAC*

Une contarinte C est arc consistante généralisée (Generalized Arc consistent GAC), si et seulement si pour chaque variable x_i dans la portée de (C) et chaque valeur V dans D_i il y'a au moins une affectation à la portée (C) qui affecte v à x_i et satisfait c [14].

1.2.2.2 *Algorithme GAC*

Une contrainte c est une est consistante d'arc généralisée si et seulement si elle existe au moins un sipport pour chaque littérale de c. Un probléme P est GAC si et seulement si chaque contrainte de P est GAC.
L'application de GAC consiste à supprimer des domaines toutes les valeurs qui ne sont pas prise en charge sur une contrainte.
De nombreux algorithmes ont été conçus pour établir le GAC selon la nature des contraintes.

1.2.2.3 *Interconsistance*

Un probléme $P = (X, D, C)$ est dit interconsistant s'il posséde des relations non vide et si tout tuple cohérent d'une contrainte c peut-être étendue de maniére cohérente à toute autre contrainte qui intersecte c [15].

1.2.2.4 *Consistance relationnelle*

Les consistances relationnelles offerent une caractérisation élevée du niveau de consistance local nécessaire pour assurer la consistance globale pour des contraintes d'une rigueur donnée [16].

- **Contrainte relationnellement arc consistante**
  Un probléme $P = (X, D, C)$ est relationnellement arc consistant, si toute instanciation consistantpour toutes les variables d'une contrainte sauf une peut-être étendue à la variable finale de façon à satisfaire la contrainte [16].

- **Contrainte relationnellement chemin consistante**
  Un ptobléme est relationnellement chemin consistant, si toute affectation cohérente pour toutes les variables d'une paire du contrainte, sauf une peut-ête étendue à la variable finale de maniére à satisfaire les deux contraintes [15].

1.2.3 *Algorithmes énumératifs*

Ces algorithmes consistent à visiter toutes les configurations possible de l'espace de recherche. Ces algorithmes seront détaillés dans ce qui suit.

1.2.3.1 *Le retour-arrière chronologique (Backtrack)*

BT effectue un retour chronologique sur la dernière variable affectée dans le cas où toutes les valeurs de la variable courante x ont été essayées sans succés pour étendre de manière consistante l'affectation courante. Cependant, il est possible que cette variable ne soit pas en cause dans cet échec. Par exemple, si cette variable n'est pas liée à x par une contrainte, elle ne peut être en cause dans l'échec. Alors, une nouvelle valeur pour cette variable va mener au même échec. Pour éviter ces échecs répétés, il est possible de revenir sur une variable en cause plus en amont dans l'ordre d'affectation. Ce retour en arriére non chronologique est appelé Backjump. Plusieurs méthodes ont été définies dans cette optique. Elles diffèrent dans leur façon d'analyser les échecs pour permettre de revenir aux véritables causes [9].

1.2.3.2 *Algorithmes avec retour arrière non chronologique*

- **Algorithme de BackJumping**
  L'algorithme Backjumping est une modification du Backtrack qui peut remonter de plusieurs couches dans l'arbre de recherche lorsqu'il détecte une affectation qui ne peut pas être étendue à une solution [9].

- **Algorithme Graph-Based Backjumping**
  Le retour arriére est basé sur le graphe de contraintes. Si l'affectation courante ne peut-être étendue de façon consistante à une variable x, alors l'algorithme revient en arriére à la variable appartenant au voisinage x. Soit y cette variable, si toutes les valeurs du domaine de y ont été essayé, GBJ revient sur la variable la plus profonde qui appartienne soit au voisinage de y, soit au voisinage de toute variable instancier aprés y qui soit une cause potentille de l'échec d'une extention de l'affectation courante et ainsi de suite [17].

1.2.3.3 *Heuristiques d'ordre*

Pour améliorer les techniques énumératives, y'a deux types d'heuristique ont été developpées.

- **Heuristique sur l'ordre d'instanciation des variables**
  L'idée générale des heuristiques sur l'ordre d'instanciation des variables consiste à instancier en premier les variables les plus "critiques", c'est-à-dire, celle qui interviennet dans beaucoup de contarainte et/ou qui ne peuvent prendre que trés peu de valeurs.
  L'ordre d'instanciation des variables peut-être : statique, ou dynamique.

✓Statique : quand il est fixé avant de commencer la recherche. L'idée est d'instancier en premier les variables les plus courantes, c'est-à-dire celle qui participe au plus grand nombre de contrainte. Parmi ces heuristiques, nous pouvons citer :
Maxdeg : les variables sont ordonnées selon un ordre décroissant de leurs degré d'apparition dans les contraintes.
Mindeg : cette heuristique est constriute on inverssant le principe de maxdeg, elle consiste à ordonner les variables selons l'ordre décroissant de leurs degré d'apparition dans les contraintes [28].
✓dynamique : quand la prochaine variable à instancier est choisi dynamiquement à chaque étape de la recherche par exemple, l'heuristique "first-fail" consiste à choisir à chaque étape, la variable dont le domaine a le plus petit nombre de valeur localement consistante avec l'affectation partiel en cour [28].
Parmi ces heuristiques, nous pouvons citer :
dom+deg : cette heurstique consiste à choisir la variable qui a le plus petit domaine courant. Dans le cas d'égalité, choisir la variable avec le plus grand degé.
dom/deg : cette heuristique consiste à choisir la variable qui donne le rapport minimal entre la taille du domaine courant et le degré de la variable [13].

- **Heuristique sur l'ordre des valeurs à affecter en priorité aux variables**
  Parmis ces heuristiques, nous citons mi-conflit, où les valeurs sont choisi par ordre croissant du nombre de conflit qui leurs est associer. Pour une valeur, le nombre total de conflits correspond à la somme du nombre de conflit pour cette valeur présente dans les domaines courants des variables voisins à cette variable [20].

1.2.3.4 *Algorithmes avec filtrage avant*

Dans backtrack, le test de consistance esy effectué en fonction des variables déja instancier. La technique du look-ahead, procéde dés l'affectation de la variable x, a la suppression de toutes les valeurs des variables non instancier incompatible avec cette valeur de x.

- **Algorithme Forwwad checking** Lors de l'instanciation d'une variable x_i, l'application de ForwarChecking engendre la suppression des valeurs qui sont incohérentes avec l'instanciation courante, desdomaine de variables futures situées dans le voisinage x_i. La complixité en temps de FC, dans le pire des cas, est édentique à celle de BT, c'est-à-dire $O(m.d^n)$. Par contre, FC obtient généralement de meilleurs résultats que BT [18].

- **Algorithme Maintaining Arc Consistency MAC** Maintaining Arc Consistency MAC a été considéré comme l'un des plus efficaces pour résoudre des problémes de satisfaction de contrainte difficiles et de grande taille. au cour de cet algorithme se trouve l'algorithme de cohérence de l'arc AC dont l'efficacité joue un rôle essentiel dans la performance global de

MAC.
pendant le recherche, la cohérence des arcs est maintenue aprés chaque instanciation d'une variable afin d'élaguer l'espace de recherche .
Il existe plusieurs version de MAC selon l'algorithme AC utilisé, MAC3 s'il utilise AC3, MAC2001 s'il utilise AC2001, etc [19].

1.2.3.5 *Algorithmes Backtrack avec mémorisation*

Définition : Good et Nogood. Un Good (resp.Nogood) est une affectation partielle consistante qui peut (resp. ne peut pas) s'étendre en une solution. Un Good (resp. Nogood) minimal est tout Good (resp. Nogood) non composé lui-même d'un Good (resp. Nogood). Durant la recherche, il existe certaines parties du probléme qui sont résolues à plusieurs reprises. Ces redondances peuvent être évitées en enregistrant le travail effectué pour une réutilisation ultérieure. Généralement, les informations mémorisées sont des affectations sur des sous-ensembles de variables. Dans le cas où une affectation de ce type ne peut être étendue en une solution, le terme de nogood est assez souvent utilisé. Si par contre, elle peut etre étendue de maniére consistante sur une partie du probléme, on parle de good. Ces deux notions permettent d'éviter un grand de redondances et ainsi d'avoir parfois de meilleures bornes de complexités [9].

1.3 *Conclusion*

Dans ce chapitre, nous avons présenté la définition d'un CSP où on a présenté des exemple des CSP connu, puis nous avons présenter les différentes approches de résolution proposées dans la littérature, sachant qu'on a concentré sur les méthodes ennumératives qui s'appuie sur les heuristiques et sur des techniques de filtrage.
Dans le deuxiéme chapitre, nous allons le consacrer pour présenter les défférentes méthodes de décomposition structurelle présentées dans la littérature.

Chapitre 2

Les techniques de décomposition structurelles

2.1 Introduction

Aprés avoir étudier les problémes de satisfaction de contrainte, nous allons aborder un nouveau chapitre afin d'étudier et de comparer des différentes méthodes de décomposition structurelle.
Dans ce chapitre, nous commençons par faire des rappels sur les graphes et hypergraphes, pour ensuite présenter des méthodes de décomposition sur lesquelles sont basée plusieurs méthodes de résolution puis on fini par une conclusion.

2.2 *Graphes et hypergraphe*

Dans cette séction, on va présenter des propriétes d'un graphe et hypergraphe :

2.2.1 *Graphe*

un graphe G est une paire (X, C) avec X l'ensemble des sommets du graphe et C l'ensemble de ses arcs.

2.2.2 *Connexité*

Un graphe G est connexe, si et seulement si tous ses sommets sont mutuellement accessible, c'est-à-dire, il existe un chemin entre chaque paire de ses sommets. Il désigne le temps passée pour l'envoie et la recéption d'un message ou d'un paquet de donnéés entre l'emetteur et le distinataire
.

2.2.3 *composante connexe*

Une composante connexe CC d'un graphe $G = (V, E)$ est un sous ensemble maximal de sommets tel que deux quelconques d'entre eux soient reliés par une chemin :
Si $x \in CC$ alors,
- $\forall y \in CC$, il existe un chemin reliant x à y.
- $\forall z \in V \setminus CC$, il n'est existe pas de chemin reliant de x à z.

2.2.4 *Cycle*

Un cycle dans G, est une chaine $(x_{u_1}, x_{u_2}, ..., x_{u_p}), (P > l)$ qui contient au moins trois sommets distincts, et tel que$x_{u_1} = x_{u_p}$.

2.2.5 *Chaine*

Une chaine entre deux sommets x_i et x_j dans G est une séquence $(x_{u_1}, x_{u_2}, ..., x_{u_p})$ de sommets de somment G telle que :
- $x_{u_1} = x_i$ et $x_{u_p} = x_j$.
- $\forall v - 1 \leq v \leq p - 1, x_{u_v}$ et $x_{u_{v+1}}$ sont voisins.

2.2.6 *Hypergraphe*

un hypergraphe H est une paire (X, C) avec X l'ensemble des sommets du graphe et C l'ensemble de ses hyperarétes. Un hypergraphe qui ne contient que des hyperarêtes d'arité 2 est un graphe [2].

2.3 *Traitabilité*

Dans cette section, nous allons étudier la définition de la traitabilité ensuite, on va présenté l'acyclicité d'un hypergraphe, puis le join Tree, aprés on va voir c'est quoi un graphe triangulé.

2.3.1 *Définition*

Une classe I d'instance CSP est dite "traitabilité" s'il existe un algorithme en temps polynomial pour déterminer les membre de cette classe, et un algorithme en temps polynomial qui résout toutes les situations de cette classe [21].

2.3.2 *Acyclicité d'un hypergraphe*

Un hypergraphe est dit "acyclique" s'il posséde un arbre de jointure.

2.3.3 *JoinTree*

Un arbre de jointure (JoinTree) pour un hypergraph de contrainte de contrainte H est un arbre dont les nœuds sont des arêtes de H telles que chaque fois que le même sommet $X \in V$ apparait dans 2 hyperarêtes s_1 et s_2 dans S, alors s_1 et s_2 sont connectées, et X apparait dans chaque nœud sur le chemin unique reliant s_1 et s_2 dans JT [22].

2.3.4 *Graphe triangulé*

Un graphe G est triangulé si tout cycle de G de longueur strictement superieur à 3 posséde une corde, c'est-à-dire une arête reliant deux sommets non consécutifs du cycle.

2.4 *Principales méthodes structurelles*

Dans cette section, nous allons étudier les différentes méthodes de décomposition structurelle.

2.4.1 *Principe de ces méthodes*

Le principe des téchniques de décomposition structurelle est de calculer un arbre de jointure équivalent pour un hypergraphe de contrainte donnée. Chaque nœud de cet arbre est un sous-probléme pour lequel on trouve toute les solutions.
Autrement dit, c'est de transformer un CSP quelconque en un CSP traitable équivalent : un CSP ayant les même solutions que le CSP de départ.
Les méthodes de décompositions consiste à regrouper les sommets et les arêtes de l'hypergraphe d'un CSP en clusters de variable ou d'arête et d'organiser les clusters entre eux pour former un CSP dont la structure est un arbre.
Ces méthodes sont basé sur la notion de la largeur (width) qui est le critére de mesure la cyclicité de l'hypergraphe de contrainte, telle que pour uneconstante k fixe, toute instance CSP peut avoir une décomposition de largeur inférieur ou égale à k. Aprés la décomposition, le CSP peut-être résolu en un temps polynomial (la résolution n'est plus exponentielle en taille de probléme maisen fonction de cette largeur).
Dans ce qui suit, nous allons explorer quelques une des méthodes de décomposition structurelle les plus interissante dans la littérature [22][23].

2.4.2 *Méthode Biconnected components (Bicomp)*

La méthode biconnected component est une méthode qui se base sur la notion de composante biconnexe pour la décomposition des CSP binaire.
Une composante C biconnexe est un ensemble de sommets qui n'ont pas d'arêtes de séparation.
Une décomposition bicomponent de $H = (G, E)$ est un couple (T, X) tel que T est un arbre et X une fonction d'étiquetage qui asocie, soit une composante biconnexe de H où, un sommet singleton contenant un séparateur de H. On a deux nœuds p et q de T sont connecté si $X(p)$ est une composante biconnexe et que $X(q)$ est un séparateur contenu dans $X(p)$ [24].

2.4.3 *Méthode cycle cutset (CCM)*

La méthode cycle cutset [18] ou méthode coupe cycle développée pour les CSP binaires, se base sur le principe que l'instanciation d'une variable revient à la supprimer du réseau de contrainte associé à un CSP. Ceci permet donc de changer la connectivité du graphe.
On appelle ensemble coupe cycle (cycle Cutset) un ensemble de variables dont l'instanciation permet la création d'un graphe acyclique. De cette manière, l'identification de l'ensemble coupe cycle permet donc l'application de l'algorithme de résolution d'arbre au reste de problème pour trouver la solution en un temps polynomial.
L'algorithme général de résolution avec cette méthode consiste a utiliser une technique de résolution de type backtrack et de tester à chaque instanciation de variable la cyclicité du graphe résultant jusqu'à ce que celui-ci soit reconnu comme étant acyclique. Dans ce cas, l'ensemble des variables instanciées forme l'ensemble coupe cycle. Un algorithme de résolution d'arbre est utilisé pour résoudre le problème restant. Si une solution est trouvée alors l'algorithme s'arrête sinon il effectue un backtrack sur les variables de l'ensemble coupe cycle [23].

2.4.4 *Méthode Hinge*

Définition : soit H un hypergraphe, $H \subset E$, et $F \subset (E - H)$. Alors F est appelé connexe par rapport à H. Si, pour deux arêtes quelconque $\{e, f\} \in F$, il existe une suite d'arêtes $e_1, ..., e_n$ dans F telles que :
(i) $e_1 = e$.
(ii) pour $i = 1, ...n - 1, [(e_i \bigcap e_i + 1) - (\bigcup H)] \neq \{\}$.
(iii) $e_n = f$.
Définition : soit $G(V, E)$ un hypergraphe réduit et connexe, et soit un sous ensemble d'arêtes $E \subseteq H$ contenant au moins deux arêtes.
Soit $c_1, ..., c_m$ les composantes connexes de $(E - H)$ par rapport à H. H est dit un Hinge si pour $i = 1, ..., m$, il existe une arête $h_i \in H$ telle que :

$(\bigcup H_i) \bigcap (H) \subseteq h_i$.
Un hinge est dit minimal s'il ne contient aucun d'autre Hinge.
Une décomposition Hinge est un arbre tel que toutes les conditions suivantes soit vérifiées :
(1) Les sommets (nœuds) de T sont des Hinge minimals.
(2) Chaque arête est contenant dans au moins un sommet de T.
(3) Deux sommets adjacents A et B de T partagent précisement une arête L de plus, L se compose exactement des variables partagées par A et B.
(4) les variables partagées par deux nœuds de T sont entiérement contenue dans tout nœud sur le chemin le reliant [12].

2.4.5 *Méthode hypercutset*

Il s'agit d'une simple modification de la méthode CUTSET où le cutset est composé des hyperarêtes plutôt que des sommets de l'hypergraphe donné.
Un cycle hypercutset d'un hypergraphe H est un ensemble $H \subseteq E$ de telle sorte que le sous hypergraphe de H induite par des variables est acyclique.
La largeur Hypercutset de H est 1 si, H est acyclique, sinon, c'est la cardinalité minimale sur tous ses hypercutset de cycle possible [12].

2.4.6 *Méthode de décomposition arboréscente (Tree decomposition)*

Une décomposition arboriscente d'un graphe $G = (X, C)$ est un couple (E, T) où $T = (I, F)$ est un arbre(I est un ensemble de nœuds et F un ensemble d'arêtes) et $E = \{E_i; i \in I\}$ une famille de sous-ensemble de X, telle que chaque sous-ensemble E_i correspond à un nœud i de T et verifier :
(i) $\bigcup_{i \in I} E_i = x$,
(ii) pour chaque arête $\{x, y\} \in c$, il existe $i \in I$ avec $\{x, y\} \subseteq E_i$.
(iii) pour tout $i, j, k \in I$, si k est sur le chemin entre i et j dans T, alors $E_i \bigcap E_j \subseteq E_k$.
La longeur d'une décomposition arborescente w^+ est égale à $max_{i \in I}|E_i| - 1$.
La longeur arborescente dite Tree-width w de G est la largeur minimale pour toutes les décompositions arborescentes de G. Si la Tree-width d'un hypergraphe est 1 alors l'hypergraphe est acyclique [18].

2.4.7 *Méthode Tree clustering (Tcluster)*

La tree clustering s'appuie sur l'algorithme de triangulation mis au point qui transforme un graphe quelconque en un graphe triangulé. Les cliques maximales résultant de cet algorithme formeront les contraintes du CSP acyclique équivalent. L'algorithme de triangulation consiste en deux étapes :
1) Ordonner les sommets du graphe primal en utilisant l'heuristique *Maximum Cardinality Search*

(MCS).
2) ajouter récursivement des arêtes entre chaque couple de nœuds non adjacents qui sont connectés par des nœuds supérieurs dans l'ordre calculé précédemment.
L'heuristique MCS numérote les sommets du graphe primal de 1 à n (n : nombre de variable de CSP) de telle sorte que le prochain numéro à attribuer est assigné au sommet ayant le plus de voisins déjà numérotés [23].

2.4.8 *Méthode hypertree décomposition généralisée (GHD)*

Définition : Hypertree. Soit $H =< V, E >$ un hypertree associé à un hypergraphe H est un triplet $< T, \chi, \lambda >$ où $T =< N, E >$ est un arbre enraciné et χ et λ deux fonctions d'étiquettage des nœuds de T. Les fonction χ et λ associe à chaque nœud $p \in N$ deux ensembles $\chi(p) \subseteq V$ où V est l'ensemble des variables de CSP et $\lambda(p) \subseteq E$.
Définition : Hypertree décomposition. Une hypertree decomposition(HTD) d'un hypergraphe $h =< V, E >$ est un hypertree $HD =< T, \chi, \lambda >$ qui satisfait les conditions suivantes :
1. Pour chaque hyperarête $h \in E$, il existe $p \in nœuds(T)$ tel que $var(h) \subseteq \chi(p)$ où $var(h)$ est l'ensemble des variables de l'hyperarête h.
2. Pour chaque $v \in V$, l'ensemble $p \in nœuds(T)|v \in \chi(p)$ induit un arbre connecté.
3. pour chaque $p \in nœuds(T), \chi(p) \subseteq var(\lambda(p))$.
4. Pour chaque $p \in nœuds(T), var(\lambda(p)) \bigcap \chi(T_p) \subseteq \chi(p)$.
La largeur de l'hypertree décomposition généralisé $< T, \chi, \lambda >$ est $max_{p \in nœuds(T)}|\lambda(p)|$. Lhypetree width (hw($H$)) d'une hypertree décomposition est la largeur minimum de toutes ses hypertree decompositions.
Définition : Hypertree décomposition généralisée. Une hypertree decomposition généralisée est une hypertree decomposition qui ne satisfait pas la condition 4 de la définition précedente.
la longeur d'une hypertree decomposition généralisée $< T, \chi, \lambda >$ est $max_{p \in nœuds(T)}|\lambda(p)|$. L'hypertree width (Ghw(h)) d'une hypertree decomposition généralisée est la largeur minimum de toutes ses hypertree decompositions généralisées [29].

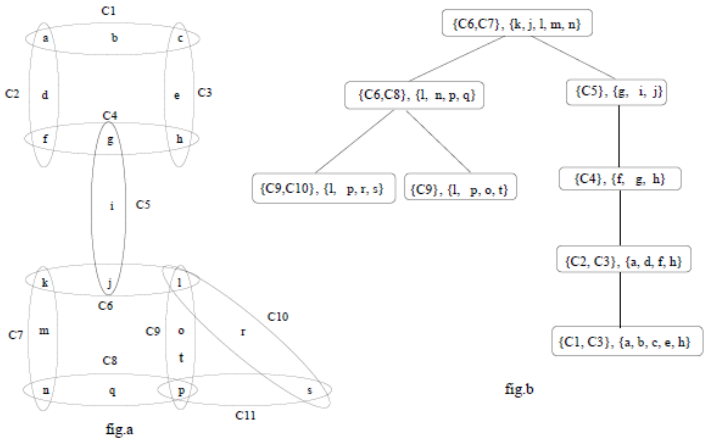

FIGURE 2.1 – Exemple de décomposition GHD

Définition : Une hyperarête h est **fortement couverte** dans un hyperarbre $\langle T, X, \lambda \rangle$ s'il existe un noeud p tel que $var(h) \subseteq X(p)$ et $h \in \lambda(p)$.
Définition (GHD complète) Une décomposition en hyperarbre (généralisée) $\langle T, X, \lambda \rangle$ d'un hypergraphe $H = \langle V, E \rangle$ est **compléte** si chaque hyperarete h de $H = \langle V, E \rangle$ est fortement couverte dans $\langle T, X, \lambda \rangle$.

2.4.8.1 *Calcule d'une GHD*

Pour une décomposition en hyperarbre généralisée, il existe deux approches qui sont proposée dans la littérature : les méthodes exactes et les méthodes heuristiques.
- **Méthodes exactes** : le premier algorithme proposé pour trouver une décomposition en hyperarbre optimale est *opt-k-decomp*. Etant donné un hypergraphe $\mathcal{H} = \langle V, E \rangle$ une constante k, *opt-k-decomp* calcule une décomposition en hyperarbre de $\mathcal{H}$ de largeur $\leq ksi(htw(\mathcal{H}) \leq k)$. Sinon $htw(\mathcal{H} > k)$, cet algorithme retourne échec. *opt-k-decomp* a une complexité temporelle de $O(m^{2k}n^2)$ où m est le nombre d'hyperarêtes, n est le nombre de sommets et k est une constante. Pour améliorer cet algorithme plusieurs alogorithme sont proposés comme *Red-k-decomp*, *det-k-decomp*.
Comme si ces méthodes exactes necessite beaucoup de mémoire et beaucoup de temps d'exécution, d'autres méthodes ont été proposées sous le nom des méthodes heuristiques afin de régler ce probléme.
- **Méthodes heuristiques méta-heuristiques** : Plusieurs méthodes heuristiques et méta-heuristiques ont été propoées dans la littérature ; on trouve *Korimort* où il a proposé une heuristique basée sur la connectivité des sommets de l'hypergraphe et Dermaku et al. ont proposé les heuristiques suivantes : BE (Bucket Elimination), DBE (Dual Bucket Elimination) et les techniques de partitionnement de l'hypergraphe, etc [2].

2.4.9 *Cyclic-clustering*

La méthode du Cyclic-clustering combine celle du Tree Clustering avec celle de Cycle (hyper)cutset. Elle essaie de tirer profit de leurs avantages respectifs. La premiére phase consiste à calculer un ensemble de cliques maximales de de l'hypergraphe de contraintes du CSP. Ces cliques maximales définissent un hypergraphe qui recouvre les contraintes du probléme. Cet hypergraphe induit une décomposition plus fine que celle qui est donnée par une décomposition arborescente du probléme dans laquelle les clusters recouvrent les cliques maximales. Cependant, l'hypergraphe n'est pas forcément acyclique. De ce fait, la seconde phase va consister à appliquer la méthode du Cycle (hyper)cutset sur cet hypergraphe. On obtient donc un cutset de taille réduite par rapport à une application directe de la méthode sur le CSP [9].

2.4.10 *Backtracking on tree-Decomposition*

Cette méthode sera évoquée trés souvent dans la suite, nous allons donc la présenter plus en détails. Elle est basée sur une décomposition arborescente du graphe de contraintes du CSP. Cette derniére induit un ordre d'affectation partiel sur les variables qui permet de tirer profit de la structure du probléme pour éviter beaucoup de redondances.
La complexité en temps de BTD est en $O(exp(w+1))$ tandis que sa complexité spatiale est en $O(n.s.exp(s))$, avec s la taille de la plus grande intersection entre clusters [9].

2.5 Classification des méthodes structurelles

Dans cette section, nous allons présenté une classification des méthodes structurelle étudier dans la séction précédente.
Cette classification des méthodes structurelle est résumé dans la figure ci-dessous, où une fléche d'une méthode A vers une autre B veut dire que B est plus générale que A.
Dans notre classification, on remarque que generalized hypetree decomposition est la plus générale des classe étudier.

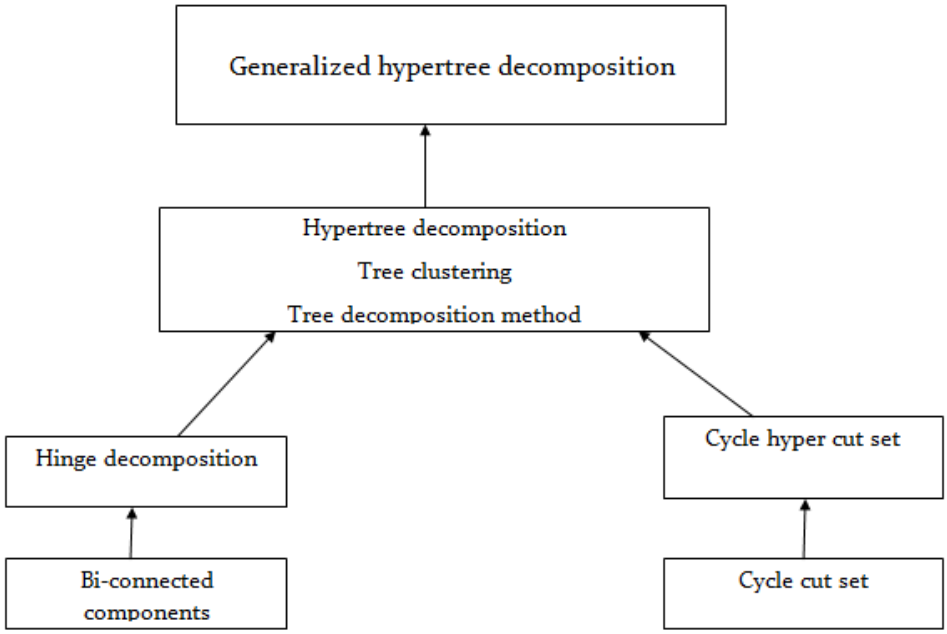

FIGURE 2.2 – Hiérarchie des méthodes de décomposition structurelles [2]

2.6 Conclusion

Dans ce chapitre, nous avons présenter les différentes méthodes stucturelle pour la résolution des CSP présenté dans la littérature.
Dans le prochain chapitre, nous allons présenter une autre méthode de décomposition qui est la décomposition sémantique pour la résolution des CSP.

Chapitre 3

Les techniques de décomposition sémantiques

3.1 Introduction

Dans un probléme de satisfaction de contrainte (CSP) l'objectif est de trouver une affectation de valeurs à un ensemble donné de variables, sujets à des contraintes spécifiées.
La plupart des outils pour la résolution de CSP continus s'appuient sur des consistances locales qui encadrent l'espace solution.
Dans ce chapitre, nous allons aborder les CSP conservatrices où on va présenter la résolution en as-composant, puis la réduction de Maroti, deuxiément, on va entamer une autre décomposition sémantique, cette méthode de décomposition représente une décomposition de domaine guidée par la sémantique des contrainte de distance, et puis on fini une conclusion.

3.2 CSP conservatrice

Les probléme de satisfaction par contrainte conservatrice constitut un cas de CSP générale, dans lequel mes valeurs autorisée de chaque variable peuvent être restreintes d'une maniére arbitraire. Les problémes de ce type sont bien étudiés pour les homorphismes de graphe.
Les CSP ont été étudiés pour les language constitués d'une seule relation symetrique binaire, c'est-à-dire équivalent au probléme de l'homorphisme des grahes.
La conjecture de dichotomie a été comfirmé pour les CSP conservatrices, donc l'algorithme de résolution des CSP developpé dans les traveaux précédents utilise une approche alegebrique, qui relie un language de contrainte a une collection de polymorphisme du graphe, c'est-à- dire, utiliser des opérations d'un même ensemble qui préserve toutes les relations du language. Par exemple, pour caractériser les CSP de 2 éléments résoluble en temps polynomial, il suffit de considérer que 4 types d'opperations constante, semi-réseau, majoritaire($(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$), et affine $(x - y + z)$.
Téoreme : Soit Γ un language de contrainte sur un ensemble A. Le CSP conservateur utilise

les relations en temps polynomial, si pour tout sous-ensemble à 2 élements $a, b \subseteq A$, il existe une opération f sur A, telque f sur a, b est soit une opération semi-réseau, majoritaire où une opération affine.

3.2.1 *Probléme de satisfaction de contrainte*

$[n]$ est un ensemble, et soit $A_1, ..., A_n$ des ensembles, tout ensemble $A_1 \times ... \times A_n$ un tuple $(n - aire)$.
une relation n-aire sur $A_1 \times ... \times A_n$ est tout ensemble de tuple sur ces ensemble.
La relation R sur un produit sous-direct de $A_1, ..., A_n$ si $Pr_i = A_i$ pour tout $i \in [n]$ sacahant que Pr_i désigne la projection de R sur i.
Une introduction à l'algébre universelle et à l'approche algebrique du VSP peut-être trouvée dans [4], [7], [6], [5] Soit A une collection d'ensemble finis, le CSP $(A) = (V, \delta, C)$ où :
V : un ensemble fini d variable.
δ : une fonction de domaine où : $\delta(v) = A$ attribuant un domaine de valeur pour chaque variable.
C : un ensemble de contraintes où : $C =< S, R >$ tel que $S = (v_1, ..., v_k)$ est une suite de variable de V et R une relation sur $\delta(v) \times ... \times \delta(v_k)$ appelée la relation de contrainte.
Un probléme P dit 3-minimal, s'il contient une contrainte $\langle w, S_w \rangle$ pour chaque élement-3 $w \subseteq v$, et pour tout $W_1, W_2 \subseteq V$ tel que $|W_1| = |W_2| = 3$ et $|W_1 \bigcap W_2| = 2$, $Pr_{w_1 \bigcap w_2} Sw = Pr_{w_1 \bigcap w_2} Sw_1 \bigcap Pr_{w_1 \bigcap w_2} Sw_2$.
La relation $R \subseteq A_1 \times ... \times A_k$ est un sous-algébre du produit direct $\mathcal{A}_1 \times ... \times \mathcal{A}_k$ noté $R \leq \mathcal{A}_1 \times ... \times \mathcal{A}_k$.
Avec $CSP(\mathbb{A})$ on note la classe d'instances $CSPp = (V, \delta, C)$ telle que $\delta(v)$ est l'univers d'un des membres de $\mathbb{A}$, et toute relation de contrainte est une sous-algébre du produit direct desalgébres de domaines.
Une algébre est dite conservatrice si chaque sous-ensemble de son univers est une sous-algébre.
Un polynôme unaire d'une algébre $\mathcal{A}$ est une application $P(\mathcal{A}) = \mathcal{A}$ poue laquelle il existe une opération à terme $t(X, y_1, ..., y_k)$ et des élements $a_1, ..., a_k \in \mathcal{A}$ tel que $P(x) = t(x, a_1, ..., a_k)$.
Le polynôme unaire $p(x)$ est independent si $P(P(x)) = P(x)$.

3.2.1.1 *Graphique, chemin et les trois operations de base*

Si $\mathbb{A}$ est une classe d'algébre conservatrice férmé par des sous-algébres alors tout sous-algébres $\mathcal{B}$ pour tout $\mathcal{A} \in \mathbb{A}$.
Par conséquent, si un $CSP(\mathbb{A})$ est résoluble en temps polynomial, alors pour tout sous-algébres à 2 éléments $\mathcal{B}$ de $\mathcal{A}$ (on suppose : $B\{0, 1\}$ il existe une opération terme $f_{\mathcal{B}}$ de $\mathbb{A}$ telle que $f_{\mathcal{B}}^{\mathcal{B}}$ est l'une des opérations permettant la traçabilité du CSP sur un ensemble de 2-élements.
$f_{\mathcal{B}}^{\mathcal{B}}$: est soit une opération semi-réseau (conjonction ou disjonction), ou l'operation majoritaire

$(x \vee y) \wedge (y \vee z) \wedge (z \vee x)$, ou, une oppération affine $x - y + z(mod(2))$.
Téoreme : Soit $\mathbb{A}$ une classe fini d'algébre conservatrice. Le probléme $CSP(\mathbb{A})$ peut-être résolu en temps polynomial si et seulement si pour tout $\mathcal{A} \in \mathbb{A}$ et tout sous-algébre à 2-élements $\mathcal{B}$ de $\mathcal{A}$, il existe une opération à terme $f_{\mathcal{B}}$ de $\mathbb{A}$ telle que $f_{\mathcal{B}}^{\mathcal{B}}$ est soit semi-réseau, soit majoritaire ou affine.
Définition : Pour une relation $R \leq A_1, ..., A_n$ un digraphe $G(R)$ peut-être définie d'une façon naturelle :
✓les tuples $a, b \in R$ forment une arête de semi-réseau orienté de a vers $b(a \leq b)$ si $a[i] = b[i]$ ou $a[i] \leq b[i]$ pour tout $i \in [n]$.
✓Les tuples a, b forment une arête majoritaire si $a[i] = b[i]$ où $a[i]b[i]$ est majoritaire pour chaque $i \in [n]$.
✓Les tuples a, b forment une arête affine, si $a[i] = b[i]$ où $a[i]b[i]$ est affine pour chaque $i \in [n]$.
Définition : Une suite de sommets $a_1, ..., a_k$ de $G(R)$ est un chemin si, tout a, a_{i+k} est sois un semi-réseau, soit une arréte affine.
Un ensemble $S \subseteq R$ est dit connecté s'il existe un chemin de chaque élement de S à chaque élement de S.

3.2.2 *Propriété de graphe étiqueté des algébres*

3.2.2.1 As-composant, relation et connectivité

Définition : soit $\mathcal{A} \in \mathbb{A}$ une algébre conservatrice. Un ensemble $B \subseteq A$ est dit as-composant si pour tout $b \in A$ et $a \in A - B$ l'aréte ab est sois semi-réseau soit majoritaire dirigé de a vers b.
Lemme : soit $R \leq \mathcal{A} \times \mathcal{B}$ un sous-directet soit $\mathbf{A}$,$\mathbf{B}$ des as-composant de $\mathcal{A}$ et $\mathcal{B}$ respectivement, telle que $\mathbf{R} = R \bigcap (\mathbf{A} \times \mathbf{B}) \neq \emptyset$. Puis R est un produit sous direct de $\mathbf{A}$, $\mathbf{B}$.
Lemme : soit $R \leq \mathcal{A} \times \mathcal{B}$ un sous-directet soit $\mathbf{A}$, $\mathbf{B}$ des as-composant de $\mathcal{A}$ et $\mathcal{B}$ respectivement, telle que il existe, $a \in \mathbf{A}$ avec $a \times \mathbf{B} \subseteq R$. Puis $\mathbf{A} \times \mathbf{B} \subseteq R$.
Lemme : soit $\mathcal{A}, \mathcal{B}$ des produit sous-direct d'algébre conservatrice et sois $R \subseteq \mathcal{A} \times \mathcal{B}$ des produits sous-direct et liée. Soit $\mathbf{A}, \mathbf{B}$ des as-composant de $\mathcal{A}, \mathcal{B}$ respectivement, tel que $R \bigcap (\mathbf{A} \times \mathbf{B}) \neq \emptyset$. Puis $\mathbf{A} \times \mathbf{B} \subseteq R$.
Lemme : soit $R \leq \mathcal{A}_1 \times ... \times \mathcal{A}_n$ un produit sous direct des algébres conservatrices. $\mathcal{A}_1, ..., \mathcal{A}_n$, et soit $\mathbf{A}_i$ un as-composant de $\mathcal{A}_i$ pour $i \in [n]$, tel que $(a_1, ..., a_n) \in R$ pour un certain $a_i \in \mathbf{B}_i$. Puis $\mathbf{R} \bigcap (\mathbf{A}_1 \times ... \times \mathbf{A}_n)$ est un produit sous-direct de $\mathbf{A}_i$ et $\mathbf{R}$ est un as-composant de R.

3.2.2.2 *Rectangularité*

Définition : soit $R \leq \mathcal{A}_i \times ... \times \mathcal{A}_n$ un sous-direct et soit $\mathcal{C}_i \subseteq \mathcal{C}_j$ est un as-composant de $\mathcal{A}_i, \mathcal{A}_j$ respectivement. La position i, j sont dit $\mathcal{C}_i$,$\mathcal{C}_i$-apparentés si $a[i] \in \mathcal{A}_i$ si et seulement si $a[j] \in \mathcal{A}_j$.
Définition : Un ensemble $I \subseteq [n]$ est appelé un *brin* par rapport a as-composant. $\mathbf{A}_1, ..., \mathbf{A}_n$ de $\mathcal{A}_1, ..., \mathcal{A}_n$ respectivement, si elle est maximal telle que tout $i, j \in I$ sont $\mathbf{A}_i, \mathbf{A}_j$-apparentés.

3.2.3 *Résolution des CSP conservatrice*

Soit $\mathbb{A}$ une classe fini d'algébre conservatrices fermé par sous algébres. Dans cette section, nous présentons un algorithme résolvant $CSP(\supset)$, aprés avoir etudier deux réduction de probléme : La réduction d'exclusion en as-composant, la réduction de maroti.

3.2.3.1 *La réduction d'exclusion en as-composant*

La premiere réduction convertit le probléme en un certain nombre d'instance CSP dans lesquelles chaque domaine est un as-composant et, alors soit fournit une solution soit permet d'éléminer certains élements de certains des domaines d'origine.
Soit $P = (V\delta, C)$ une instance $CSP(\mathbb{A})$ nous supposant que P est en 3-minimal, une collection de as-composant $\mathbf{A}_v\delta(v)$ pour chaque $v \in V$ est appelé une collection cohérente si : pour une contrainte $\langle(v_1, ..., v_n), R\rangle$ l'ensemble $R\bigcap(A_{v1}, ..., A_{vn})$ est non vide.
Un brin de P par rapport à $\mathbf{A}_v, V \in V$ est un ensemble maximal, $W \in V$ tel que une partition $W_1 et W_2$ de W certains $w_1 \in W_1$ et $w_2 \in W_2$ sont dans le même brin par rapport à $\mathbf{A}_{v_1}, ..., \mathbf{A}_{v_n}$ d'une contrainte $\langle(v_1, ..., v_n), R\rangle \in C$.
Proposition : Soit $P = (V, \delta, C)$ une instance 3-minimal et $W \subseteq V$. Ensuite, toute collection cohérente partielle sur W peut-être étendue à une collection cohérente.
Lemme : Soit R une relation n-aire et $I[n]$ pour tout $a \in Pr_iR$ telle que $a[i]$, $i \in I$ fait partie d'un as-composant, il existe $b \in R$ de telle sorte que $b[i], i \in [n]$, appartenant a un as-composant et que $b[i] = a[i], i \in I$.
Lemme : Soit $\mathbf{A}_1, ..., \mathbf{A}n$ une collection consiste pour une relation n-aire R alors
$(\mathbf{A}_1 \times ... \times \mathbf{A}n)\bigcap R \neq \emptyset$.

3.2.3.2 *La réduction de Maroti*

Soit $\mathbb{A}$ une classe d'algebre finies fermées par des sous-algébres. Supposons que $\mathbb{A}$ a une opperation de terme f satisfait les conditions suivantes pour certain $\mathcal{A} \in \mathbb{B}$.
1-$f(x, f(x, y)) = f(x, y)$ pour tout $x, y \in \mathcal{A}$.
2-$\mathbb{A}$ est fermé sous retracte via des pôlynomes unaire $f(a, x)$, $f(x, a)$.
3-Pour tout $a \in \mathcal{A}$ l'application $x \to f(a, x)$ n'est pas surjective.
4- L'ensemble C de $a \in \mathcal{A}$ telle que $x \mapsto (x, a)$ est surjective engendre un sous-ensemble propre de $\mathcal{A}$ Puis $CSP\mathbb{A}$ réductible en temps pôlinomial.
comme nous avons précis

- **La réduction et la modification** Soit $P = (V, \delta, C)$ une instance du $CSP(\mathbb{A})$ et $p_v : \delta(v) \longrightarrow \delta(v)/v \in V$ l'application P_v est dite :
  Cohérente : Si pour tout $\langle S, R\rangle \in C, S = (v_1, ..., v_k)$ et tout tuple $a \in R$ le tuple $(p_{v_1}(a[1]), ..., p_{v_k}(a[k]))$ appartient à R.

Permutationnelle : Si toutes sont des permutations.
Idempotente : Si toutes sont des idempotente. La construction suivante utilise une opération idempotente binaire **.** satisfait l'identité $x(x.y) = x.y$ alors $t(p)$ designe l'instance (V, δ, C) où :
- $V = \{(v, b)|v \in V, b \in \delta(V) = \{b.x|x \in \delta(v)\}\}$.
- Les domaines sont définie par :
$\delta(v, b) = b.\delta(b) = \{b.x|x \in \delta(v)\}$.
- C des contraintes de deux types :
Le premier type : pour chaque $v \in V$ il contient la contrainte $\langle S_v, R\rangle$ où $S_v = ((V_1 b_1, ..., (V_k b_k))$ pour une énumération $b_1, ..., b_k$ des éléments de $\delta(v)$ et $R_v = \{b_1.c, ..., b_k.c)|c \in \delta(v)\}$.
Le deuxiéme type : pour tout $C = \langle S, R\rangle \in C, s = (v_1, ..., v_k)$ et $a \in R$ il existe une contrainte $D_{c,a} = \langle S_{c,a}, R_{c,a}\rangle$ donnée par $S_{c,a} = ((v_1, a[1]), ..., (v_k, a[k]))$ et $R_{c,a} = \{a.x|x \in R\}$
La dérniére construction est réservé uniquement pour les algébres conservatrices.
Soit $B_v$un ensemble de tout les $b \in \delta(v)$ de telle sorte que ab est une arête semi-réseau pour aucun $a \in \delta(v)$.

3.2.3.3 *L'algorithme et son temps d'exécution*

Soit $P = (V, \delta, C)$ de $CSP(\mathbb{A})$ pour résoudre les instances libre semi-réseau. Chaque arête de $G(\delta(v))$ tel que $v \in V$.
$\forall v \in V$ et $\forall a, b \in \delta(v)$.
$m(x, y, z) = W(g(x, y, z), g(y, x, z), g(z, x, y))$ est une opération majoritaire si ab est une arête majoritaire, et une opération affine ab est une arête affine.
Si P est compact, nous appliquons la réduction de maroti, comme décrit dans la section 4.2.
Si P est un domaine avec un prope as-composant, nous appellerons la réduction d'exclusion as-composant, et sois trouver une solution, et soint réduire certains domaines. Cette réduction fait des appels récursifs avec des instances que lesquel chaque domaine est un as-composant [30].

3.3 *Les CSP numériques*

La satisfaction d'un système de contraintes numériques est un problème qui a de nombreuses applications dans les science de l'ingénierie, un (NCSP) comme doit les autres CSP est défiué par un ensemble de domaine D, variable X, est de contraintes C. Les méthodes de résolution des CSPs numérique sont basées sur une approche de type branche et prune. L'idée de ces méthodes est de découper le domaine d'une variable afin de diviser le problème en deux sous-problème.
On utilisant une méthode de filtrage, nous permet de réduire ou de éliminer chaque sous-problème puis, un nouveau domaine est choisi pour être découpé et le processus recommence jusqu'à ce que tous les domaines soint de taille inférieure à un certain paramètre de précision.

3.3.1 *Décomposition sémantique des domaines*

Dans cette section nous allons parler sur une technique de décomposition sémantique pour les contraintes de distance nommée SDD (Demantic Domain Décomposition). Cette méthode consiste à découper et réduire les domaine des points mise en jeu dans une contrainte de distance en utilisant ses propriétés de monotonie :
La forme canonique des équations de distance $X^2 + Y^2 - D^2 = 0$ est utilisé pour ésoler ces parties monotones sur $\mathbb{R}^+ \times \mathbb{R}^+$, $\mathbb{R}^+ \times \mathbb{R}^-$,$\mathbb{R}^- \times \mathbb{R}^+$,$\mathbb{R}^- \times \mathbb{R}^-$.

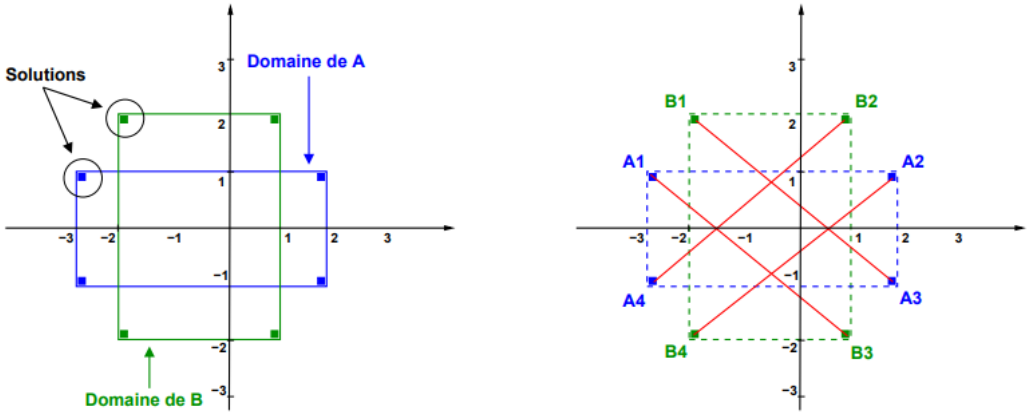

FIGURE 3.1 – La figure de gauche montre les domaines de A et B après un filtrage par 2B-consistance

Dans la figure, la technique utilisé est le filtrage par 2 B-consistance, cette technique ne permet pas de réduire ces domaines. Le côté gauche de la figure, montre les contumunns de solutions sont proches des bornes des domaines. A droite, la figure montre que la SDD divise le CSP initial en 4-CSP correspondant aux 4 sous-espaces de solution.
Ces sous-espaces sont modélisés par un graphe de sous domaine dont les arêtes sont les segments : A_1B_3, A_2B_4, A_3B_1 et A_4B_2.

3.3.2 *Décomposition sémantique d'un CSP*

Définition : Nous utilisons la définition classique des CSP, telle qu'elle est annoncée par marckworth.
Dans le cas des CSP continue, les domaine de variables sont représentées par des intervalles, de la forme $D_x = [\underline{x}, \overline{x}]$.
Le problème de satisfication de contraints de distance consiste à déterminer les positions de n points satisfaisant un ensemble de relations de distance euclidienne dans un espace borné.
La contrainte de distance représente une equation quadratique qui met en jeu les coordonnées de

deux points.
$P_i(x_i^1, x_i^2, ..., x_i^p)$ et $P_j(x_j^1, x_j^2, ..., x_j^p)$ et une variable δ_{ij} :
On peut déduire que la forme la plus générale de CSP est la suivante :

$$\sum_{l=1}^{p}(x_{i_k}^l - x_{j_k}^l)^2 = \delta_k^2 \text{ avec}\delta_k \in \delta_k$$

$P = \{P_i(x_i^1, x_i^2, ..., x_i^p)\}/\ 1 \leq i \leq n$ un ensemble de n points de $\mathbb{R}^p$.
$D = \{D_i^1, D_i^2, ..., D_i^p / 1 \leq i \leq n$ un ensemble de domaine associé aux composantes des points telle que : $D_i^k = [\underline{x_i^k}, \overline{x_i^k}]$. Le domaine de point P_i est D_i qui représente la boite $D_i^1 \times D_i^2 \times ... \times D_i^p$.
$\sigma = \{\sigma_k\}, 1 \leq k \leq m$ un ensemble de m intervalles associées aux distances euclidienne δ_k entre les couples de points (P_{ik}, P_{jk}) definie un ensemble de m contraintes.

$\forall k \leq m, c_k$

$$\sum_{l=1}^{p}(x_{i_k}^l - x_{j_k}^l)^2 + (y_{i_k}^l - y_{j_k}^l)^2 = \delta_k^2 \text{ avec } \delta_k \in \delta_k$$

Nous proposons une stratégie de résolution qui tire profit des informations spécifiques aux contraints de distance. les caractéristiques les plus importantes de cette méthode sont résumées dans ce qui suit :
- Un découpage simultané de toutes les variable liées par une contrainte.
- Une stratégie de choix de point de coupe qui exploite les propriétés sémantique des contraintes.
Dans ce qui suit nous présentons cette technique qui représente le but de ce chapitre.

3.3.2.1 *Décomposition des domaines guide par la sémantique des contraintes*

Définition : sémantic Domain decomposition SDD considérons la contrainte $C(x, y)$: $x^2 + y^2 = \delta^2$ où $(x, y) \in D$.
On définit l'ensemble $SDD(C, D)$ des sous domaines engendrés par la décomposition sémantique de domaine D relativement à la contrainte C. Ces sous domaines sont définit par :
$D_+^+ = \square\{(x, y) \in D; x \geq 0 \wedge y \geq 0 \wedge c(x, y)\}$
$D_+^- = \square\{(x, y) \in D; x \leq 0 \wedge y \geq 0 \wedge c(x, y)\}$
$D_-^- = \square\{(x, y) \in D; x \geq 0 \wedge y \leq 0 \wedge c(x, y)\}$
$D_-^+ = \square\{(x, y) \in D; x \geq 0 \wedge y \leq 0 \wedge c(x, y)\}$
où $\square S$ designe la plus petite boite contenue tous les éléments de l'enemble S.
La fonction SDD réalise cette décomposition sémantique en considérant une contrainte de distance sous forme canonique C et le domaine $D = D_x \times D_y$ de vecteur associé à C.

3.3.2.2 *Réécriture de CSP initial*

Considérons un CSP $P = (X, D, C)$ constitué exclusivement de m contraintes de distance de la forme :
$c_k : (x_{jk} - x_{ik})^2 + (y_{jk} - y_{ik})^2 = \delta_k^2$.

La réécriture du CSP initial consiste à mettre toutes les contraintes sous forme canonique. Pour chaque contrainte c_k on introduit deux variables additionnelles $X_k$et Y_k correspondant aux coordonnées du vecteur $\overrightarrow{P_{ik}P_{jk}}$. Les domaines V_k des vecteurs (X_k, Y_k) sont calculés en utilisant l'arithmétique des intervalles. La réécriture du système consiste à ajouter les contraintes de changement de repère et à remplacer c_k par sa forme canonique :

$\{c\prime_k = X_k^2 + Y_k^2 = \delta_k^2$.
$X_k = x_{jk} - x_{ik}$.
$Y_k = y_{jk} - y_{ik}.\}$
L'ensemble des domaines est donc de la forme :
$DV_1, ..., V_m, D_1, ..., D_n$ où D_i est le domaine de P_i avec $1 \leq i \leq n$ et V_k est le domaine de vecteur (X_k, Y_k) associé à la contrainte $c\prime_k$ avec $1 \leq i \leq m$.

Les contraintes $c\prime_k$ étant sous forme canonique, les domaines V_k des vecteurs (X_k, Y_k) pourront être décomposés en utilisant la technique de décomposition sémantique étudié dans la sous-section précédente.

3.3.2.3 *Algorithme de décomposition sémantique*

L'algorithme DS utilise un ensemble Q contenant des sous-domaines du domaine initial et initialisé à D. S est alors extrait de l'ensemble Q et l'on vérifie si le point fixe est atteint.
Si c'est le cas, S est ajouté à R, l'ensemble destiné à contenir les sous-domaines engendrés par DS.
Sinon, on choisit un vecteur dont le domaine V_k n'a pas encore été décomposé par SDD. Pour tous les sous-domaines engendrés par la décomposition sémantique du domaine du vecteur choisi, on ajoute à Q les sous domaines de S correspondants s'ils vérifient la propriété de consistance local. L'appel à la fonction LC utilise toutes les contraintes de pour réduire les domaines de toutes les variables [31].

Function 2 DS($\mathcal{X}$,$\mathcal{D}$,$\mathcal{C}$)

```
1: R ← ∅
2: Q ← {D}
3: while Q ≠ ∅ do
4:    Choose S from Q                                   %% S = {V_1,...,V_m,D_1,...,D_n}
5:    if V_k = SDD(V_k,c_k) for all V_k ∈ S then
6:       R ← R ∪ {S}
7:    else
8:       Choose V_k from S
9:       Let c_k ∈ C the constraint associated to V_k.
10:      for all s ∈ SDD(V_k,c_k) do
11:         S' ← S[V_k ← s]          %% S' = {V_1,...,V_{k-1},s,V_{k+1},...,V_m,D_1,...,D_n}
12:         S' ← LC(X,S',C)                          %% Local filtering of the domain S'
13:         if S' ≠ ∅ then
14:            Q ← Q ∪ {S'}
15:         end if
16:      end for
17:   end if
18: end while
19: return R
```

FIGURE 3.2 – Algorithme de décomposition sémantique

3.4 Conclusion

Nous avons présenté dans ce chapitre une nouvelle méthode de recherche basée sur les différentes décompositions sémantique connu pour la résolution des CSP.
Dans ce chapitre, nous avons présenter les différentes méthodes stucturelle pour la résolution des CSP présenté dans la littérature.
Dans le prochain chapitre, nous allons faire une etudes comparative entre les méthodes structurelle étudier dans le deuxiéme chapitre et les méthodes décompositions sémantiques étudier dans ce chapitre.

Chapitre 4

Comparaison entre les méthodes de décomposition structrulles et sémantiques

4.1 Introduction

Dans les chapitres précédent, on a présenté les méthodes de décomposition structurelle et sémantique, où on démontré les différentes caractéristéques de chaque méthodes avec leurs fonctionnement.
Dans ce chapitre, nous allons comparer quelques méthodes structurelles par rapport à quelques méthodes sémantiques.

4.2 *comparaison entre les méthodes étudier*

4.2.1 Compléxitéé

Le tableau ci-joint représente la compariason entre les compléxités de quelques méthodes structurelle et sémantiques.

Méthodes	Type	Compléxité	
		En temps	En espace
Cycle cutset	Structurelle	$O(exp(\|S\|+2))$	linéaire
Cycle clustering	Structurelle	$O(exp((w^- + 1) + \|S^-\|))$	$O(n.s.d^s)$
BTD	Structurelle	$O(exp(w+1))$	$O(n.s.exp(s))$
Tree clustering	Structurelle	$O(exp(w+1))$	$O(exp(w+1))$
B-Comp	Structurelle	$O(n.d^k)$	/
Hinge	Structurelle	$O(\|V\|.\|E\|^2)$	/
Décomposition sémantique	Sémantique	$O(m)$	/
CSP conservatrice	Sémantique	$O(\|V\|.s\xi^{r^l}p(\|\mathcal{P}\|))$	/

Tableau 4.1 – Tableau comparatif entre les méthodes structurelles et sémantiques

- **Comparaison entre deux méthodes étudier "Cycle cutset et décomposition sémantique**

  Dans cette partie nous allons comparer deux méthodes étudier dans le chapitre précédent. La complexité temporelle de la méthode Cutset est en $O(d^{k+2})$ où k représente la taille de l'ensemble coupe cycle et d est la taille de plus large domaine. En effet, le coùt nécessaire à la recherche d'une affectation consistante sur l'ensemble coupe cycle est $O(d^k)$ pour chaque affectation consistante, il faut résoudre le CSP acyclique induit, dont la complexité de cette tâche est en $O(d^2)$ du coup, la comlexité totale est en $O(d^{k+2})$ par contre, la méthode décomposition sémantique utilise la décomposition SDD (décomposition des domaines) pour choisir le prochain domaine à décomposer qui engendre un nombre de sous-domaine minimle est strictement superieur à 1, où on déduit la complexité temporelle de cette méthode est en $O(m)$ où m représente le nombre de vecteurs, c'est-à-dire, de contrainte de systeme.

4.2.2 Benchmarks

-**Modified Renault** : le probléme *Renault* original est un probléme de configuration obtenu à partir de la Renault Mégane. C'est un probléme d'usinage, où chaque variable représente une option ou une caractérisation d'un composant de véhicule et les contraintes sont les configurations possibles de la Mégane.
Il se présente en deux versions : version normalisée et version simple. Cette famolle invoque de large relations et une large arité pour les contraintes. La classe "modified Renault" contient 50 instances structurées évoquant des domaines allant jusqu'à 42 valeurs. La taille de la plus grande relation est de 48721 tuples.
- **Dubois** : Cette phase contient 13 instances insatisfiables générées de façon aléatoire. La plus large relation contient 4 tuples.
- **Pret** : cette classe contient 8 instances définies en 60 ou 150 variables. L'arité de chaque contrainte est de 3 et le nombre de tuple d'une relation est 4.
- **Random** : cette classe contient 20 instances aléatoires. L'arité maximale des contarintes est de 20. Chaque relation est définie par 10000 tuples.
- **Probléme classique de pentagone** : Le problème du pentagone est un CSP bien connu dans la communauté de la programmation par contraintes sur les domaines continus. Il est souvent utilisé pour illustrer les problèmes que rencontrent les méthodes de filtrage local, lorsque l'espace solution est fractionné, Il s'agit d'un CSP géométrique en 2D, constitué de 5 points P_1, P_2, P_3, P_4 et P_5 sur le cercle unitaire, tels que les segments P_1P_2,P_2P_3,P_3P_4,P_4P_5 et P_1P_5 soient de longueur égale à d .

4.2.3 Résultats et analyse

Aprés l'études des méthodes vu dans les chapitre précédents et la comparaison faite suivant les déffirentes instances on déduit que la méthode hypertree est la plus efficace comparant aux

méthodes car elle représente la méthodes la plus générales, ainsi la plus puissante méthode.
Aprés la comparaison faite entre la méthodes décomposition sémantique et l'algorithme de bissection, les performance de la décomposition sémantique sont bien meilleures que celle d'un algorithme de recherche basé sur la bissection.

4.3 *Conclusion*

Dans ce chapitre, nous avons introduit une comparaison entre 2 methodes vu dans le 3éme chapitre.
La comparaison est faite entre la méthode Cutset et décomposition sémantique, où, on a choisi la complexité comme paramétre de comparaison. Nous concluons que les méthodes sémantiques sont plus éfficaces que les méthodes structurelles.

Conclusion et perspectives

Dans ce mémoire, nous nous sommes intéressés à l'exploitation des méthodes de décomposition structurelles et les méthodes de decomposition sémantiques pour la résolution des CSPs. L'état de l'art que nous avons fait nous a permis de constater que les coûts du calcul des décompositions optimales sont défférents d'une méthodes a une autre sois structurelles ou sémantiques où on a choisi la compléxité comme paramétre de comparaison temporelle et spaciale. Comme perspectives nous envisageons d'implémenter quelques méthodes structurelles et quelques méthodes sémantiques pour avoir une idée claire sur le comportement pratique de ces méthodes.
Les résultats éxperémentaux obtenues sont encouragants.

Bibliographie

[1] B. Madeline, *Algorithmes évolutionnaires et résolution de problèmes de satisfaction de contraintes en domaines finis*. PhD thesis, Université Nice Sophia Antipolis, 2002.

[2] K. Amroun, *Décomposition d'hypergraphes pour la résolution des problèmes de satisfaction de contraintes*. PhD thesis, Université Béjaia, 2014.

[3] K. Amroun, Z. Habbas, and W. Aggoune-Mtalaa, "A compressed generalized hypertree decomposition-based solving technique for non-binary constraint satisfaction problems," *AI Communications*, vol. 50, no. 2, pp. 241–267, 2016.

[4] A. A. Bulatov, "Complexity of conservative constraint satisfaction problems," *ACM Transactions on Computational Logic (TOCL)*, vol. 12, no. 4, pp. 1–66, 2011.

[5] H. P. Sankappanavar and S. Burris, "A course in universal algebra," *Graduate Texts Math*, vol. 78, 1981.

[6] A. Bulatov, P. Jeavons, and A. Krokhin, "Classifying the complexity of constraints using finite algebras," *SIAM journal on computing*, vol. 34, no. 3, pp. 720–742, 2005.

[7] A. A. Bulatov and P. Jeavons, "An algebraic approach to multi-sorted constraints," in *International Conference on Principles and Practice of Constraint Programming*, pp. 183–198, Springer, 2003.

[8] T. Lambert, *hybridation de méthodes complétes et incomplétes pour la résolution de CSP*. PhD thesis in Informatique, Ecole doctorale STIM, Octobre 2006.

[9] S. N. Ndiaye, *Calcul et exploitation de recouvrements acycliques pour la résolution V(CSP)*. PhD thesis in Informatique, Ecole doctorale de mathématiques et d'informatique de marseille, décembre 2007.

[10] A. El Mouelhi, P. Jégou, C. Terrioux, and B. Zanuttini, "Sur la complexité des algorithmes de backtracking et quelques nouvelles classes polynomiales pour csp," in *Huitièmes Journées Francophones de Programmation par Contraintes-JFPC 2012*, 2012.

[11] T. Schiex, H. Fargier, and G. Verfaillie, "Problemes de satisfaction de contraintes valués," *Revue d'Intelligence Artificielle*, vol. 11, no. 3, pp. 339–373, 1997.

[12] G. Gottlob, N. Leone, and F. Scarcello, "A comparison of structural csp decomposition methods," *Artificial Intelligence*, vol. 124, no. 2, pp. 243–282, 2000.

[13] D. E. Dehani, *La substituabilité et la cohérence de tuples pour les réseaux de contraintes pondérées*. PhD thesis in Informatique, Université d'Artois, Fevrier 2014.

[14] I. P. Gent, C. Jefferson, I. Miguel, and P. Nightingale, "Data structures for generalised arc consistency for extensional constraints," in *AAAI*, vol. 7, pp. 191–197, 2007.

[15] C. Bessiere, K. Stergiou, and T. Walsh, "Domain filtering consistencies for non-binary constraints," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 800–822, 2008.

[16] T. Walsh, "Relational consistencies," in *Research Report APES-28-2001, APES Research Group*, Citeseer, 2001.

[17] Y. Bounab, *Programmation par contrainte des emplois du temps*. PhD thesis in Ingénieurie des systémes d'information, Université Abd Al-hamid Ibn Badis Mostaganem, Fevrier 2012.

[18] H. Kanso, *Résolution des problèmes (W) CSP et# CSP par approches structurelles : Calcul et exploitation dynamique de décompositions arborescentes*. PhD thesis, Aix Marseille Universitè, 2017.

[19] C. Lecoutre, C. Likitvivatanavong, S. Shannon, R. Yap, and Y. Zhang, "Maintaining arc consistency with multiple residues," *Constraint Programming Letters (CPL)*, vol. 2, pp. 3–19, 2008.

[20] N. Paris, *Intégration de techniques CSP pour la résolution du problème WCSP*. PhD thesis, Artois, 2014.

[21] D. Cohen, P. Jeavons, and M. Gyssens, "A unified theory of structural tractability for constraint satisfaction and spread cut decomposition," in *IJCAI*, vol. 5, pp. 72–77, Citeseer, 2005.

[22] Y. Zheng and B. Y. Choueiry, "New structural decomposition techniques for constraint satisfaction problems," in *International Workshop on Constraint Solving and Constraint Logic Programming*, pp. 113–127, Springer, 2004.

[23] A. A. Abdelmalek and Z. Habbas, *Exploitation de l'hypertree decomposition pour la resolution des problemes de satisfaction de contraintes*. PhD thesis, Université de Béjaia, 2008.

[24] Z. Miklós, "On the parallel complexity of structural csp decomposition methods," *Algorithms and Complexity in Durham*, pp. 107–118, 2007.

[25] A. Ait-Amokhtar, K. Amroun, and Z. Habbas, "Décompositions arborescentes pour la résolution des problèmes de satisfaction de contraintes," in *Actes du Cinquième Colloque sur l'Optimisation et les Systèmes d'Information COSI'2008*, p. 194.

[26] `https://perso.liris.cnrs.fr/christine.solnon/Site-PPC/session3/consistance-noeud-sess3.htm`, (Consulté 24 avril 2021).

[27] `https://perso.liris.cnrs.fr/christine.solnon/Site-PPC/session3/consistance-arc-sess3.htm`, (Consulté 24 avril 2021).

[28] `https://perso.liris.cnrs.fr/christine.solnon/Site-PPC/session3/e-miage-ppc-sess3.htm`, (Consulté 17 mai 2021).

[29] R. A. Ouamer, M. e. F. H. e. H. D. e. H. A. Aidene, Mohamed et Demri, and B. e. S. B. e. M. M. S. Merakeb, Abdelkader et Oukacha, "Actes du cinquième colloque sur l'optimisation et les systèmes d'information cosi'2008,"

[30] A. A. Bulatov, "Conservative constraint satisfaction re-revisited," *Journal of Computer and System Sciences*, vol. 82, no. 2, pp. 347–356, 2016.

[31] H. Batnini and M. Rueher, "Décomposition sémantique pour la résolution de systèmes de contraintes de distance,"

RÉSUMÉ

Ce mémoire ce concentre sur la comparaison entre quelques méthodes structurelles et sémantiques où on a choisi la complexité comme métrique de comparaison, aprés avoir fait une études sur les généralités des CSPs puis les pricipales méthodes structurelles et sémantiques où on a choisi les problémes de satisfaction de contrainte conservatrices dans lesquels les valeurs autorisées peuvent étre restreintes de maniére arbitraire, et une technique de décomposition de domaines guidée par la sémantique des contraintes de distance.
D'aprés l'étude qu'on a fait, nous avons conclué que les méthodes de décomposition sémantique sont plus efficace que les méthodes structurelles.

ABSTRACT

This thesis focuses on the comparison between some structural and semantic methods where we have chosen complexity as a comparison metric, after having made a study on the generalities of CSPs then the main structural and semantic methods where we have chosen the problems of satisfaction of conservative constraint in which allowed values can be arbitrarily restricted, and a domain decomposition technique guided by the semantics of distance constraints.