REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : SIOD12/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Systèmes d'information, Optimisation et Décision (SIOD)**

Le CSP

Pour le problème de gestion de temps

Par :

ACHOURI Aya

Soutenu le 26/06/2022 devant le jury composé de :

Bouchana Belkacem		Président
HOADJLI Hadia	MAA	Rapporteur
TORKI Fatima Zohra		Examinateur

Année universitaire 2021-2022

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة محمد خيضر بسكرة

تصريح شرفي

(خاص بالالتزام بقواعد النزاهة العلمية لإنجاز بحث)

أنا الممضي أسفله،

السيد (ة): .....عاشوري اية.....

الصفة: .....**طالبة**.....

الحامل لبطاقة التعريف الوطنية رقم: ....119990230048580008.....

والصادرة بتاريخ: ....2016/12/17.....

المسجل بكلية: العلوم الدقيقة وعلوم الطبيعة والحياة

قسم: الإعـــــلام الآلـي

والمكلف بإنجاز مذكرة تخرج في الماستر عنوانها:

Le CSP Pour le problème de gestion de temps

أصرح بشرفي أني ألتزم بمراعاة المعايير العلمية والمنهجية ومعايير الأخلاقيات المهنية والنزاهة الأكاديمية المطلوبة في إنجاز البحث المذكور أعلاه.

التاريخ: .....2022/06/21.....

توقيع المعني:

Remerciement

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, ce travail ne serait pas aussi riche et n'aurait pas pu avoir le jour sans l'aide et l'encadrement de Mme ***Hadia Hoadjli****, on le remercie pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce mémoire.*

*Nous souhaitons adresser nos remerciements les plus sincères au corps professoral et administratif de l'****Université Mohamed Khider – BISKRA***

Pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée

Dédicace :

Un grand merci a :

Ma mère MEROUANE Nabila

Mon père ACHOURI Tayeb

Pour leur amour, leurs conseils ainsi que leur soutien inconditionnel, à la fois moral et économique, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

Je voudrais exprimer ma reconnaissance envers ma sœur Souhila et mes frères Youcef et Idriss qui m'ont apporté leur soutien moral et intellectuel au long de ma démarche.

Le petit ange et la raison de bonheur mon petit frère Abed el Malek.

Je dédie ce travail à mes parents qui sont la cause de chaque pas dans ma vie, toute ma famille, mes amis, les personnes qui comptent beaucoup dans ma vie qu'ils soient encore en vie ou qu'ils nous ont quittées.

Résumé :

Le problème de la gestion du temps fait partie des problèmes qui occupent beaucoup de monde, notamment dans le domaine du travail. Chaque domaine particulier a beaucoup de bases qu'il faut organiser en même temps. Pour que quelqu'un propose un emploi du temps qui organise ces diverses tâches sans aucun défaut, il est considéré comme difficilement impossible.

Pour cette raison, nous avons choisi la programmation par contraintes comme moyen de développer un programme qui fixe un échéancier prenant en compte toutes les contraintes afin d'obtenir une répartition temporelle idéale qui économise l'effort humain et le temps.

تلخيص

يعتبر مشكلة تنظيم الوقت من أكثر المشاكل التي تشغل الكثير من الاشخاص خاصة في ميدان العمل فكل مجال معين لديه الكثير من الاساسيات الواجب عليه تنظيمها في نفس وقت وليقوم شخص باقتراح جدول زمني ينظم مختلف هاته المهام بدون اي خلل فيعتبر بالكاد مستحيل.

لهذا السبب اخترنا البرمجة عن طريق القيود كسبيل لوضع برنامج يقوم بوضع جدول زمني يراعي كل القيود للحصول على توزيع زمني مثالي يوفر على الانسان الجهد والوقت.

Abstract

The problem of time management is one of the problems that occupy many people, especially in the field of work. Each particular area has many bases that need to be organized at the same time. For someone to come up with a schedule that organizes these various tasks without any flaws, it is considered hardly impossible.

For this reason, we have chosen constraint programming as a way to develop a program that sets a schedule taking into account all the constraints in order to obtain an ideal time distribution that saves human effort and time.

Table des matières

Introduction

La planification est une phase très essentielle pour n'importe qu'il type domaines que ce soit une entreprise ou une école. Le problème de gestion d'emploi du temps parmi les problèmes les plus difficiles et les plus étudier. Le problème d'emploi du temps consiste à définir un certain nombre d'affectations qui permettent d'assigner plusieurs ressources (humaines, matérielles, …etc.) Sur une période de temps, tout en respecte les contraintes imposées par les entités citées (disponibilité des ressources humaines, matérielles, …etc.).

La programmation par contraintes est devenue une approche efficace pour modéliser et résoudre les problèmes de planification.

Nous allons essayer, à travers ce mémoire, de proposer une approche de résolution du problème d'emploi du temps universitaire.

Le premier chapitre de ce mémoire étudier la programmation par contraintes et le problème de satisfaction des contraintes CSP et les différentes méthodes de résolution des problèmes de programmation de contraintes.

Le deuxième chapitre est une petite étude sur la planification et le planning, après présenter les problèmes d'emploi du temps sur les différents domaines et précisément le problème d'emploi universitaire à la fin de ce chapitre une petite présentation sur des travail qui résolu le problème de génération d'emploi du temps universitaire avec les méthodes de CSP.

Le troisième chapitre traite la phase de conception de notre système après le positionnement des différents besoins qu'il doit fournir.

Le dernier chapitre aborde la réalisation de l'application avec une brève description.

En conclusion, nous citons quelques perspectives contenant des voies d'amélioration de notre système.

Chapitre 1 : La programmation par contraintes.

Introduction :

La programmation par contrainte modélise les problèmes par un ensemble de relations logiques et de contraintes.

Les problèmes de satisfaction de contraintes (CSP) sont des questions mathématiques définies comme un ensemble d'objets dont l'état doit satisfaire un certain nombre de contraintes ou de limitations. Les CSP représentent les entités d'un problème sous la forme d'un ensemble homogène de contraintes finies sur des variables, qui est résolu par des méthodes de satisfaction de contraintes.

Les CSP font l'objet de recherches à la fois en intelligence artificielle et en recherche opérationnelle

Dans ce chapitre nous présentons brièvement la programmation par contraintes qui résout Les problèmes de satisfaction de contraintes (CSP). Nous rappelons aussi les caractéristiques qui définissent le csp. Et nous discutons les différentes méthodes qui résolvent ce type de problèmes.

1. Historique CSP :

La satisfaction de contraintes est née pendant les années 1970 motivée par les travaux de Montana ri, Waltz et beaucoup plus Alan Mack Worth. A la fin des années 1970 le chercheur Mack Worth a défini le formalisme de problème de satisfaction des contraintes CSP et un ensemble de technique de résolution.

La satisfaction de contraints est rapidement devenue une discipline importante de la résolution des problèmes combinatoire qui a des excellentes performances. Dans ces premiers temps il été comme une branche de l'algorithmique, la satisfaction de contraintes a été intégrée dans divers langages de programmation. Pour des raisons techniques essentielles, les premières intégrations ont été réalisées dans le cadre de la programmation logique ce qui a donné naissance à la programmation logique avec contraintes ou CLP (Contraint Logic Programming). Plus tard, les techniques de satisfaction de contraintes ont été intégrées dans d'autres langages, notamment des langages à objets.

Les recherches actuelles concernent l'amélioration des algorithmes, dans la tradition de Mackworth.

2. Programmation par contraintes :

La programmation par contraintes c'est une technique qui est apparu à la fin de 1980 pour résoudre les problèmes de la programmation logique et l'intelligence artificielle. Elle modélise les problèmes par un ensemble de relations logiques et de contraintes.

Les contraintes peuvent être exprimées sous différentes formes : table de valeurs compatibles, formules mathématiques, … etc.

Une contrainte est une relation logique entre plusieurs variables qui limitent l'ensemble des valeurs que peuvent prendre ces variables simultanément.

Une contrainte c ∈ C sur les variables xi1, ..., xik pour i1, …, ik ∈ {1, …, n} est une relation mathématique entre ces variables. On note var(c) l'ensemble des variables intervenant dans la contrainte c.

3. Le modèle CSP :

Un problème de satisfaction de contraintes (CSP) consiste à trouver l'assignation consistante de valeurs à des variables prenant leurs valeurs dans des domaines discrets et finis. Il est à noter que la solution d'un CSP n'est pas forcément la solution optimale. Traditionnellement, l'optimisation est faite en sélectionnant la meilleure de ces solutions. (1)

Un problème de satisfaction des contraintes (CSP) est défini par trois valeurs :

- Premièrement l'ensemble des variables,
- Deuxièmes le domaine ou l'ensemble des valeurs possibles pour chaque variable,
- Finalement l'ensemble des contraintes qui définissent les valeurs que la variable peut le prendre.

La résolution d'un CSP consiste à trouver les valeurs les plus appropriés pour chaque variable de telle sorte que l'ensemble de contraintes soit satisfait.

Une autre classification des techniques de résolution d'un CSP est proposée par Yung et Yang (Yung et Yang, 1999). Les trois catégories de cette classification sont : (1)

1. La réduction du problème qui consiste à retirer les valeurs et les étiquettes de valeur redondantes. (1)

2. La synthèse de solution en construisant incrémentalement un treillis appelé « graphe du problème minimal » qui représente le problème minimal. (1)
3. La recherche qui inclut les algorithmes à retour arrière (backtracking algorithms), les algorithmes de vérification prévisionnelle (forward checking algorithms), les systèmes de maintenance de la vérité et la propagation de contraintes. (1)

Définition formelle d'un CSP : (2)

Un problème de satisfaction de contraintes (CSP) est défini par un triplet P= (X, D, C) où :

- X = {x1, x2,…, xn} est l'ensemble fini des n variables du problème.

- D = {Dx1, …, Dxn} est l'ensemble des n domaines finis pour les variables. Dxi est l'ensemble des valeurs possibles pour la variable xi.

- C = {c1, …, cm} est l'ensemble des m contraintes qui lient les variables entre elles

4. Types de problèmes CSP : (3)

- CSP avec des domaines finis (et discrets).
- CSP Booléens : les variables sont vraies ou fausses.
- CSP avec des domaines continus (et infinis) – Par exemple, problèmes d'ordonnancement avec des contraintes sur les points (début/fin) dans le temps.
- CSP avec des contraintes linéaires.
- CSP avec des contraintes non linéaires.

5. Les caractéristiques de csp :

Un CSP est défini par le triplet (X, D, C) ou le X est l'ensemble des variables du problème le D est l'ensemble des domaines pour des variables le C contraintes.

Arité : (4)

L'arité d'une contrainte c ∈ C est donnée par la cardinalité de var(c) ; autrement dit le nombre de variables sur lequel porte c. Une contrainte c est dite :

- 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 Un CSP est dit binaire, si toutes ses contraintes sont d'arité inférieure ou égale à deux.

Affectation : (4)

L'affectation est le fait d'instancier certaines variables par des valeurs (prises dans leurs domaines respectifs.

Une affectan est une fonction :

$$s : X \rightarrow \prod_{i=1}^{n} Dxi$$

Telle que s(xi) ∈ Dxi pour i ∈ [1..n]

s = (d1, d2, ..., dn) une affectation des variables avec la valeur d1 pour la variable x1, d2 pour la variable x2,..., dn pour la variable xn.

Espace de recherche : (4)

L'espace de recherche d'un CSP est l'ensemble des affectations possibles, qui est égal au produit cartésien de l'ensemble des domaines des variables ; une configuration S sera assimilée à un élément de cet ensemble S (i.e. s = (d1, ... ,dn)).

Solution :

Une solution d'un CSP P = (X, D, C) est une affectation s ∈ S qui satisfait toutes les contraintes. Sol(P) représente l'ensemble des solutions.

$$Sol(P) = \{s \in S \mid \forall c \in C, s \in c\}$$

L'ensemble des solutions correspond à tout élément de l'espace de recherche (affectations) appartenant aussi aux valeurs permises pour chaque contrainte.

Exemple 1 (CSP simple) : (4)

Soit le CSP (X, D, C) suivant :

- X = {a, b, c, d},
- Da = Db = Dc = Dd = {0, 1},
- C = {a ≠b, c ≠ d, a + c < b}.

Ce CSP comporte 4 variables a, b, c et d, chacune pouvant prendre 2 valeurs (0 ou 1). Une solution possible pour ce problème est a = 0, b = 1, c = 0 et d = 1.

Étant donné un CSP (X, D, C) sa résolution consiste à affecter des valeurs aux variables, de telle sorte que toutes les contraintes soient satisfaites.

6. Application de csp :

Placement des n-reines (2)

Le problème des n-reines consiste à placer n reines sur un échiquier de n*n cases de telle manière qu'aucune reine ne soit pas prise par une autre. Le but est donc de placer une seule reine sur une même ligne, colonne et diagonale.

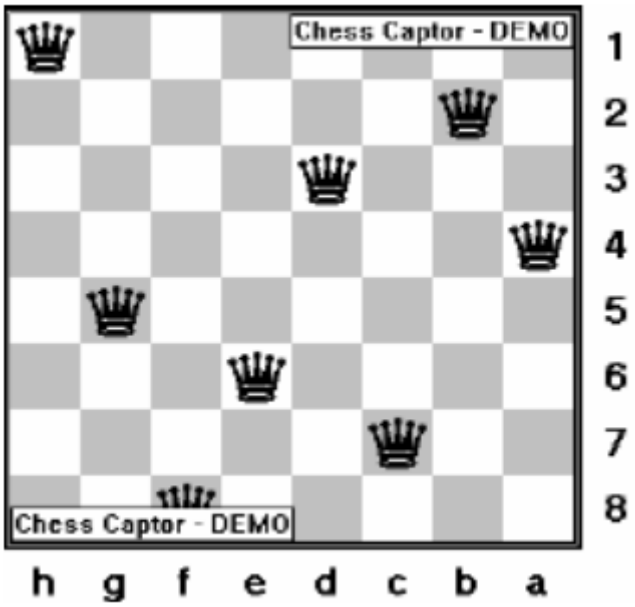

*Figure 1:: Une solution réalisable pour le placement de 8 reines sur un échiquier 8*8*

Il y a plusieurs façons de modéliser ce problème sous forme d'un CSP. Une possibilité est de considérer chaque colonne i comme une variable xi. Une variable possède alors un domaine de (1 à n) qui désigne le numéro de ligne où se place la reine de la colonne.

Formellement, et pour le problème avec 8 reines, nous avons :

- X = {x1, x2, …, x8}
- D = {Dx1, Dx2, …, Dx8} avec D(xi) = {1, …, 8}
- C est défini par, ∀ i, ∀ j ≠i : les reines doivent être sur des lignes différentes, soit xi ≠ xj , et les reines doivent être sur des diagonales différentes, soit xi + i ≠ xj + j et xi - i ≠ xj – j Une solution réalisable de ce problème est illustrée en Figure 1.

Coloriage de la carte d'Australie (2)

La question qui se pose est la suivante : peut-on colorier la carte des états de l'Australie avec trois couleurs ?

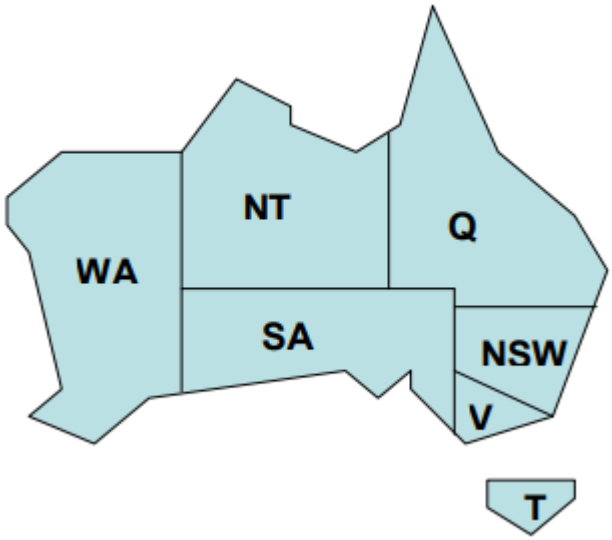

Figure 2:Carte de l'Australie

Figure 2 comporte 7 régions devant être coloriées en rouge, bleu et vert. Pour obtenir le CSP équivalent, chaque région sera représentée par une variable. A chaque variable on attribue un domaine qui correspond à l'ensemble des couleurs. Pour chaque paire de régions adjacentes, une contrainte binaire est créée entre les variables correspondantes. Cette contrainte interdit d'affecter une couleur identique à ces deux variables. Formellement :

- X = {WA, NT, SA, Q, NSW, V, T}

- D = {DWA, DNT, DSA, DQ, DNSW, DV, DT} avec DWA = DNT = DSA = DQ = DNSW = DV = DT = {rouge, bleu, vert} WA NT Q SA NSW V T

- C = {WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q, SA ≠ Q, SA ≠ NSW, SA ≠ V, Q ≠ NSW, NSW ≠ V}

Une solution réalisable pour ce problème est la suivante (voir Figure 3) :

Figure 3:Solution possible pour le coloriage de la carte de l'Australie

Jeu Sudoku : (2)

Le jeu Sudoku est un jeu de logique qui consiste à compléter une grille n*n, avec n=p^2 et p>=2, avec les chiffres de 1 à n. Les contraintes sont qu'un chiffre ne doit pas être utilisé deux fois par ligne, par colonne et par carré de n cases. (2)

Prenons l'exemple illustré en Figure 4. Le CSP équivalent de cette grille pourra être obtenu en représentant chaque case par une variable ; si la case est vide, le domaine de la variable est [1,…,9] ; si la case est pré-affectée, le domaine de la variable est restreint à la valeur pré-affectée. On définit des contraintes de différences pour toutes les cases qui se trouvent dans les mêmes carrés de 3*3, sur les mêmes lignes et sur les mêmes colonnes. (2)

Formellement et pour l'exemple, le CSP équivalent est : (2)

- X = {x11, ..., x99}
- D = {Dx11, ..., Dx99}
- C = x11≠x12, ..., x11≠x19, x21≠x22, ..., x91≠x99 (contraintes de ligne)

X11≠x21, ..., x11≠x91, x12≠x22, ... x19≠x99 (contraintes de colonne)

AllDiff(x11,...,x33),…, AllDiff(x77,...,x99) (contraintes de carré)

	1	2	3	4	5	6	7	8	9
1	8	6			2				
2				7				5	9
3									
4					6		8		
5		4							
6			5	3					7
7									
8		2					6		
9			7	5		9			

*Figure 4:Exemple d'une grille SUDOKU 9*9 à compléter*

7. Variantes du CSP : (1)

Il existe différentes variantes du CSP original. Par exemple, dans les ICSP (Interval CSP) l'objectif n'est plus de trouver l'assignation des variables qui satisfait aux contraintes, mais les intervalles de ces variables où les contraintes sont respectées (Yung et Yang, 1999).

Les systèmes multi agents étant des systèmes distribués, nous nous intéressons plus particulièrement aux travaux effectués dans un autre sous-domaine des CSP : les CSP distribués ou DCSP. (1)

Un DCSP est un CSP dans lequel les variables et les contraintes sont distribuées entre des agents. Ces variables se divisent en deux ensembles disjoints : les contraintes intra-agents et les contraintes inter-agents. Une contrainte intra-agent n'est connue que par un agent. une contrainte inter-agent est connue par tous les agents ayant une variable dans cette contrainte. Comme dans le cas centralisé, résoudre un DCSP consiste à trouver une assignation de valeur aux variables en ne violant aucune contrainte (bien que la littérature des DCSP se concentre principalement à la résolution des contraintes inter-agents). Trouver une assignation de valeur aux variables inter-agents peut être vu comme réaliser la cohérence ou la consistance d'un système multi agent (1) .

8. Les méthodes de résolution csp :

Il existe différentes approches pour résoudre les problèmes de satisfaction de contraintes. Nous montrons un résumé non-exhaustif des différentes approches existantes pour aborder un CSP, Algorithmes Génétiques, Décomposition, Réparation par Escalade. Représentation par Réseaux de Contraintes.

Nous pouvons aussi classifier en deux classes les méthodes de résolution : Méthodes Complètes et Incomplètes. Les méthodes complètes font une recherche arborescente en instanciant les variables une par une et en effectuant un retour en arrière en cas d'échec. Les méthodes incomplètes font une réparation d'une configuration en parcourant de manière non systématique (aléatoire) l'espace de recherche.

Dans la section suivante nous présenterons différentes techniques et méthodes de résolution : (2)

- Les techniques basées sur les notions de consistance avec les algorithmes de filtrage et de propagation de contraintes.
- Les méthodes de résolution complètes, c'est la méthode la plus basique qui consiste à l'énumération de toutes les combinaisons possibles vers des méthodes plus élaborées qui utilisent les notions de consistance et de filtrage.
- Les méta-heuristiques qui sont des méthodes de résolution incomplètes souvent capables de donner une bonne solution en un temps acceptable.
- Les méthodes hybrides qui combinent des techniques issues de la programmation par contraintes avec des méthodes incomplètes comme la recherche locale et les algorithmes génétiques.

1. Le graphe de contraintes :

Le graphe de contraintes d'un CSP binaire est un graphe simple où les sommets représentent les variables et les arêtes représentent les contraintes. S'il existe une contrainte c telle que var(c) = {x, y} on trouve une arête entre les sommets des variables x et y.

Le concept de graphe de contraintes peut être défini pour des CSP binaire et non binaires. Donc Les contraintes sont représentées par des hyperarêtes qui relient des sous-ensembles de variables impliquées dans une même contrainte.

Le voisinage d'un sommet du graphe est l'ensemble des sommets adjacents de ce graphe, c'est-à-dire la liste des sommets auxquels on peut accéder directement depuis le sommet courant.

Le degré d'un sommet du graphe est le nombre d'arêtes qui entrent et qui sortent du sommet en cours.

2. Résolution des CSP par des méthodes complète :

2.1Generate-and-test et backtracking:

Generate-and-test :

Est Une méthode simple pour la résolution des CSP prend toutes les combinaisons des valeurs des variables et les tester si elles représentent des solutions (vérifient les contraintes).

Le nombre de possibilités testées est alors le cardinal du produit cartésien des domaines des variables, les problèmes de grandes tailles devient impossible à envisager.

Le backtracking :

Est sans nul doute la méthode la plus répandue pour une recherche systématique. Pour cette méthode, les variables sont instanciées les unes après les autres et ce jusqu'à obtenir une affectation complète. Seulement, contrairement à un generate and-test, cette méthode teste la faisabilité à chaque étape de la résolution, c'est-`a-dire que pour chaque instanciation de variables, les contraintes dont les variables sont déjà instanciées sont vérifiées, sur l'affectation partielle courante. (4)

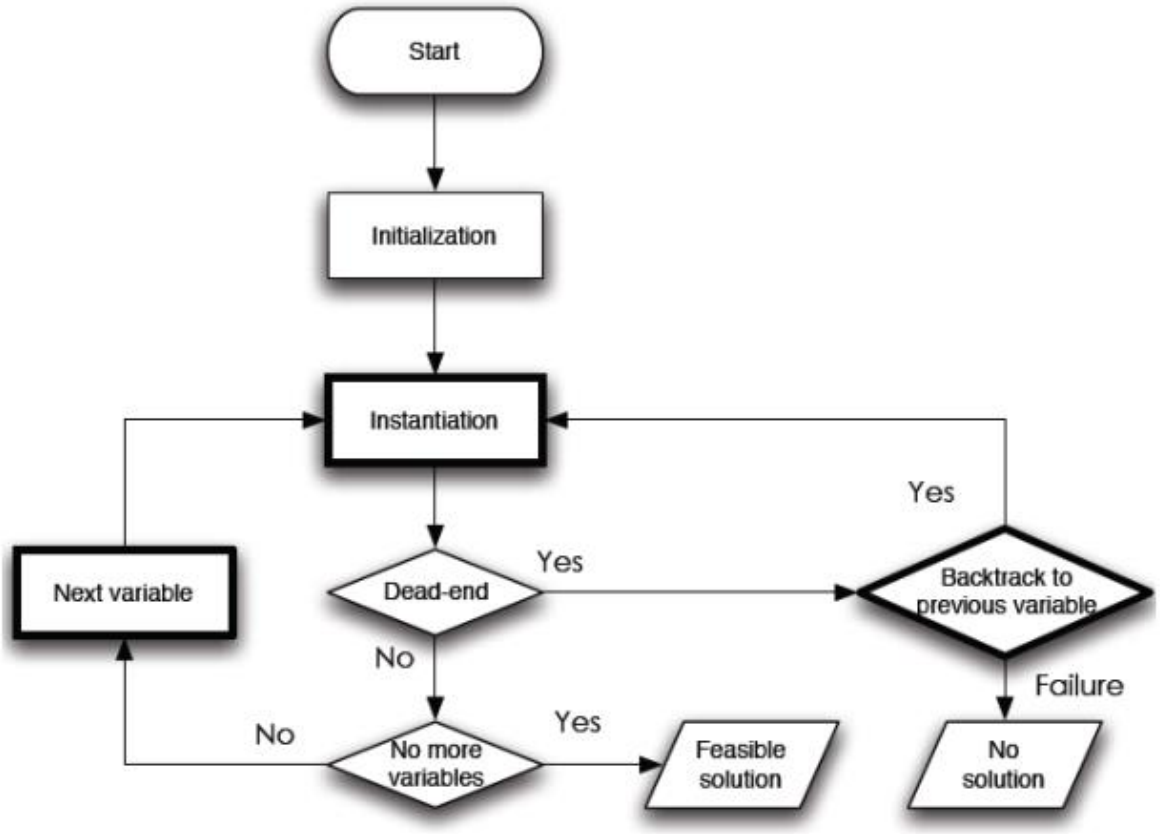

Figure 5: Schéma général du backtracking

L'avantage majeur du backtracking est qu'il est capable de résoudre tous les CSP de tous les types de contraintes.

Son inconvénient majeur est sa complexité temporelle exponentielle.

Notion de consistance :

Notion de consistance est associée à la notion contrainte. Une contrainte force les variables à ne prendre que certaines valeurs, la consistance intervient pour supprimer les valeurs qui ne satisfaire pas cette contrainte.

Cette méthode est utilisée pour réduire l'espace de recherche donc on supprime les valeurs dites inconsistance qui n'est pas validé pour une ou plusieurs contraintes.

Consistance de nœud :

La consistance de nœud ne concerne que les contraintes unaires. Etant donné un CSP (X, D, C), pour le rendre consistant de nœud il faut et il suffit pour chaque contrainte unaire de C portant sur les variables de supprimer toutes les valeurs inconsistantes dans le domaine de ces variables. (2)

Consistance d'arc :

La consistance d'arc concerne les contraintes binaires, elle est dite aussi 2-consistance. Une contrainte binaire qui porte sur deux variables xi et xj est dite arc consistante, si chaque valeur dans i Dx possède au moins une valeur dans j Dx tels que ces deux valeurs vérifient les contraintes sur le couple (xi, xj). (2)

Consistance d'hyper-arc :

L'arc consistance ne concerne que les contraintes binaires, tandis que la consistance hyper-arc généralise cette notion pour les contraintes n-aires. (2)

2.2Algorithmes de filtrage et propagation de contraintes :

Le filtrage et la propagation sont des algorithmes définis pour assurer la consistance. L'objectif est de réduire les domaines pour rendre les CSP consistants relativement aux propriétés de chaque contrainte.

Le filtrage des valeurs inconsistantes est utile pour instancier les variables et permet de retirer certaines valeurs des domaines tout en conservant les solutions. Les différents algorithmes de filtrages reposent bien sûr sur le type des contraintes exprimées.

Le filtrage supprimer pour une contrainte donnée c qui porte sur une variable xi l'ensemble des valeurs appartenant à Dxi ne satisfaisant pas la contrainte c.

La propagation de contraintes, permet d'assurer la consistance des domaines avec chaque contrainte.

2.3Les métaheuristiques :

Les métaheuristiques sont des stratégies de la recherche optimale. Visant à déterminer les solutions presque optimales à partir d'exploration de l'espace de recherche efficacement.

Les métaheuristiques visent principalement à minimiser ou maximiser une fonction de coût donnée appelée fonction fitness dans le sens où elle vise à évaluer la qualité d'une solution. Ce qui pour les CSP correspond par exemple à minimiser la somme des contraintes violées pour des configurations complètes.

Un ensemble de propriétés intéressantes qui caractérisent les métaheuristiques : (2)

- Les métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale.

- Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales.

- Les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.

- Les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.

- Les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.

- Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire appel à un problème spécifique.

- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.

- Les métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum pour mieux guider la suite du processus de recherche

Ces propriétés définissent le comportement de toutes les métaheuristiques pendant la recherche d'une solution Ces propriétés définissent le comportement de toutes les métaheuristiques pendant la recherche d'une solution, en allant de la recherche locale jusqu'aux algorithmes génétiques qui sont parmi les plus complexes. Les métaheuristiques visent principalement à minimiser ou maximiser une fonction de coût donnée appelée fonction fitness dans le sens où elle vise à évaluer la qualité d'une solution. Ce qui pour les CSP correspond par exemple à minimiser la somme des contraintes violées pour des configurations complètes. (2)

2.3.1Recherche locale :

La recherche locale est une méthode générale utilisée pour résoudre des problèmes d'optimisation, c'est-à-dire des problèmes où l'on cherche la meilleure solution dans un ensemble de solutions candidates. La recherche locale consiste à passer d'une solution à une autre solution proche dans l'espace des solutions candidates (*l'espace de recherche*) jusqu'à ce qu'une solution considérée comme optimale soit trouvée, ou que le temps imparti soit dépassé. (5)

```
Function RechercheLocale
1.    Construire une configuration S de départ
2.    while un critère d'arrêt n'est pas vérifié do
3.          S=Select( N(S))
4.    endwhile
5.    return Meilleure configuration rencontrée
```

Figure 6:: Fonctionnement global de la recherche locale

La première étape d'une méthode de recherche locale est la construction d'une configuration de départ. Cette étape est simple, on utilise généralement une configuration aléatoire ou des algorithmes gloutons pour construire une bonne configuration de départ. Le reste est un processus itératif. Il consiste à remplacer la configuration courante S par une autre, voisine de S. Il s'arrête lorsque le critère d'arrêt prédéfini est vérifié : un nombre d'itérations, temps d'exécution, etc. A la fin du processus, la meilleure configuration trouvée durant la recherche est retournée. (2)

2.3.2Recherche par voisinage :

Les méthodes de recherche à base de voisinage s'appuient toutes sur un même principe, à partir d'une solution $x_0$considérée comme point de départ. La recherche consiste à passer d'une solution voisine par déplacements successif. L'ensemble des solutions que l'on peut atteindre à partir d'une solution x est appelé voisinage de cette solution. (6)

De manière générale, les opérateurs de recherche locale s'arrêtent quand une solution localement optimale est trouvée, c'est-à-dire quand il n'existe pas de meilleures solutions dans le voisinage. (6)

L'inconvénient de ce type de solution est de tomber sur des minima locaux et il serait intéressant de pouvoir s'en échapper. Il faut alors permettre à l'opérateur de recherche locale de faire des mouvements pour lesquels la nouvelle solution retenue sera de qualité moindre que la précédente, c'est le cas immédiat des méthodes du recuit simulé, bruitage et de la recherche tabou. (6)

La recherche par voisinage est une étape clé dans la recherche locale car elle permet de définir des ensembles de configurations d'une solution donnée par une série de changements.

Soit S un espace de recherche (appelé aussi espace des configurations), un voisinage est une fonction

$$N : S \rightarrow P(S)$$

Qui associe à chaque élément s de l'espace de recherche S, un ensemble de voisins (élément de l'ensemble des parties P) $N(s) \subseteq S$. On appelle N (s) le voisinage de s.

A partir de la configuration courante, le choix du voisin qui sera la prochaine étape de la recherche, se fait par une évaluation. Dans le contexte des problèmes de satisfaction de contraintes, est liée au nombre de contraintes violées.

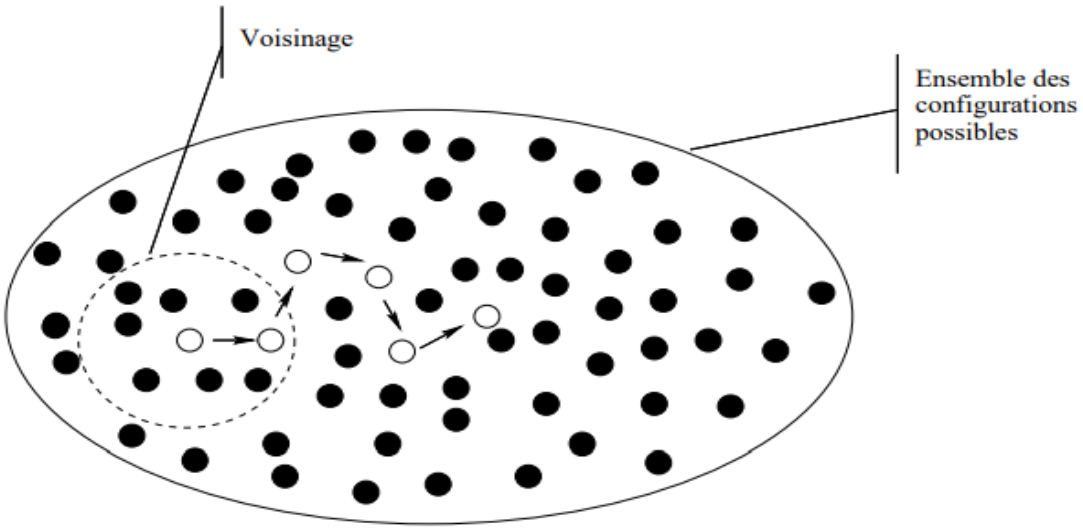

Figure 7: Chemin de recherche locale

La figure 7montre le chemin d'une recherche locale où la fonction d'évaluation des configurations complètes permet de sélectionner un voisin a chaque pas.

2.3.3Algorithmes de recherche locale :

Le recuit simulé :

Le recuit simulé est considéré comme la première métaheuristique pourvue d'une stratégie pour échapper aux minimums locaux.

Proposé par trois chercheurs de la société IBM en 1983, le recuit simulé est une méthode de recherche locale inspirée d'un processus utilisé en métallurgie, Cet algorithme est toujours très utilisé notamment dans le domaine des réseaux. (2)

La méthode de descente n'accepte qu'une configuration qui améliore la configuration courante et donc s'arrête dans le premier minimum local rencontré. (2)

Algorithme 3.3 : Recuit Simulé

```
1  Soit x une configuration initiale
2  Soit 𝒯 une fonction de refroidissement et T une température initiale
3  début
4      répéter
5          recherche dans le voisinage : choisir x' ∈ 𝒩(x)
6          calculer la variation d'énergie ΔE = f(x') − f(x)
7          tirer aléatoirement un réel p dans [0, 1]
8          si ΔE < 0 ou e^(−ΔE/T) > p alors
9              x ← x'
10         fin
11         T ← 𝒯(T)
12     jusqu'à critère d'arrêt non satisfait;
13 fin
```

Figure 8:algorithme recuit simuler

2.3.4Recherche tabou :

La métaheuristique tabou est une méthode de voisinage, utilisant des techniques qui permettent d'éviter les optima locaux et les cycles. Cette métaheuristique a été développée par (Glover, 1989, 1990 ; Glover and Laguna, 1997) et elle connaît beaucoup de succès grâce aux résultats très satisfaisants obtenus sur un grand nombre de problèmes. L'objectif de cette métaheuristique est la diversification à court terme (7)

La méthode recherche tabou est la méthode la plus populaire parmi les méthodes de métaheuristiques. Elle se base sur une descente avec sélection du meilleur voisin et sur une mémoire à court terme des précédentes configurations rencontrées.

Pour chaque itération on explore le voisinage du point s pour trouver la meilleur solution s'du voisinage de s (telle que f(s') < f(x)).

Une fonction objectif *f* doit, être minimisée dans un espace d'états X. Pour chaque point s de X. on définit, un voisinage N(s). Pour un CSP, c'est généralement l'ensemble des configurations différant de s par la valeur d'une variable. L'ensemble X et la définition du voisinage se traduisent par un graphe d'états G où deux états i et j sont lies par un arc (*i*, j) si et seulement si ***j*** appartient, au voisinage N(ï). La méthode taboue est une procédure, itérative qui, en partant d'un point initial de X, tente d'atteindre la solution optimale pour f en exécutant à chaque pas un mouvement, dans le graphe d'états G. Chaque itération consiste d'abord à produire dans le voisinage N(s) de la configuration courante a, l'ensemble V des configurations admissibles. (8)

Algorithme Recherche tabou :

Algorithme 3.4 : Recherche tabou

```
1  Soit x une configuration initiale
2  Soit l une liste tabou dont la taille maximale est k
3  début
4      répéter
5          si l est de taille k alors
6              supprimer l'élément le plus ancien i de l
7          fin
8          ajouter x a l
9          soit x' t.q. x' ∉ l et ∀y ∈ N(x) ∩ l, f(x') ≤ f(y)
10         si f(x') < f(x) alors
11             x ← x'
12         fin
13     jusqu'à critère d'arrêt non satisfait;
14 fin
```

Figure 9: Algorithme Recherche tabou

3.Algorithme génétique :

Algorithme génétique c'est un algorithme qui traite plusieurs solutions à la fois et maintiennent plusieurs solutions candidate en même temps.

Les algorithmes génétiques sont des systèmes adaptatifs inspiré de l'évolution naturelle. Ils peuvent être utilisés comme techniques de résolution de problèmes complexes et pour la

recherche de grands espaces problématiques. Les algorithmes génétiques appartiennent aux techniques d'hasard guidée, qui tentent de trouver l'optimum. J.H. Holland a présenté ce concept en début des années soixante-dix. La puissance des algorithmes génétiques et autres techniques similaires (simulation recuit, stratégies évolutives) réside dans le fait qu'ils sont capables de trouver l'optimum global dans des espaces multimodaux (espaces avec de nombreux locaux optimaux). Ils fonctionnent avec l'ensemble des solutions potentiels, ce qu'on appelle la population. Chaque élément de solution (individuel) est mesuré par la forme physique fonction. La valeur fitness représente la qualité mesure d'un individu, de sorte que l'algorithme peut sélectionner des individus avec un meilleur matériel génétique pour produire de nouveaux individus et plus loin générations. (9)

. L'algorithme est un processus itératif exécutant séquentiellement quatre opérateurs génétiques : la sélection, le croisement, la mutation et le remplacement. La sélection consiste à choisir les individus qui pourront se reproduire et se fait généralement de façon aléatoire en priorisant les meilleurs individus (10)

Les individus sont regroupés en paires pour l'étape de croisement et celui-ci s'effectue entre deux individus parents pour produire deux 18 enfants ayant les gènes de leurs parents. (10)

L'algorithme génétique utilise la notion de sélection naturelle sur une population : un ensemble d'individus correspondant à une représentation des solutions potentielles. Généralement un individu est un chromosome défini par ses gènes.

L'évaluation est une fonction qui évalue chaque solution potentielle selon le problème donné. Dans le cas des CSP cette fonction évalue le nombre de contraintes violées.

Chromosome :

Chaque variable du CSP est associée à une gêne du chromosome qui le représente. Le gène G_i correspondant une variable $X_i$est un sous-domaine.

C'est lors de l'initialisation de l'algorithme génétique que les individus originaux sont générés : pour chaque chromosome et pour chaque gène un sous-domaine aléatoire (de la taille adéquate) du domaine de la variable correspondante est construit. (11)

Valuation :

L'adaptation d'un individu est calculée lors de la résolution du CSP limité aux sous-domaines générés pour chaque variable : pour l'individu $G_1 G_2 \ldots . G_n$, les contraintes $X_i \in G_i$ sont ajoutées ; la résolution est alors réalisée de façon standard par étiquetage des variables. Si une solution est trouvée, l'adaptation est calculée par simple application de la fonction de cout aux valeurs des variables instanciées (f ($X_1 \ldots X_n$))Si aucune solution n'est trouvée (i.e. le sous-espace$G_1 G_2 \ldots . . G_n$ n'en contient pas), on attribue une adaptation nulle `a l'individu (ceci peut être raffiné pour chaque problème particulier). (11)

Mutation :

Dans le cas de CSP :

Un gène est codé par un sous-domaine des valeurs possibles de la variable correspondante ; par analogie, la mutation d'un individu consiste à remplacer l'un de ses gènes/sous-domaines par un autre sous-domaine choisi aléatoirement. Un gène est ainsi capable par mutation seule de parcourir exhaustivement son espace de recherche, c'est-à-dire l'ensemble des parties de cardinal k du domaine de la variable associée (si on suppose que le sous-domaine de chaque gène a un cardinal k). (11)

Croissement :

Dans le cas de CSP, cette méthode de croisement est utilisable directement : le croisement explore ainsi l'espace des solutions en essayant de mélanger deux sous espaces différents représentés par les individus parents.

L'inconvénient majeur de la résolution de csp par les algorithmes génétiques **:** est Par exemple lorsque les sous-domaines sont de petite taille ces opérateurs conservateurs produisent des individus proches des individus initiaux et l'exploration de l'espace de recherche risque d'en être handicapée.

Conclusion :

Nous avons décrit dans ce chapitre les principes de la Programmation Par Contraintes.et les différentes méthodes de résolution des CSP. Ces méthodes asseyent de trouver des solutions meilleures pour Les problèmes de satisfaction de contraintes.

En effectuant cette étude nous avons mis le point sur les difficultés rencontrées dans la résolution de ce type de problèmes. Et à notre tour nous avons envisagé de choisir une méthode qui s'inscrit dans le cadre de la recherche locale et qui permet une certaine souplesse dans la gestion des contraintes.

Chapitre 2 : le problème de l’emploi du temps.

Introduction :

La planification est un processus qui programme des actions et des opérations dans un domaine précis, avec des objectifs précis, et des moyens précis.

Problème d'emploi du temps universitaire, c'est un problème difficile à réaliser, Sa résolution consiste à ordonnancer les tâches d'un ensemble d'enseignants et de groupes appartenant à des sections en leur allouant un ensemble de locaux et en leur fixant des créneaux horaires.

Dans ce chapitre Nous allons présenter la problématique de la planification.et les différents types d'emploi temps précisément l'emploi du temps universitaire et leur problématique.

1.C'est quoi la planification :

La planification est l'action de planifier, c'est-à-dire d'organiser dans le temps une succession d'actions ou d'évènements afin de réaliser un objectif particulier ou un projet. Elle permet de déterminer la portée totale de l'effort, définir ou affiner les objectifs et développer la ligne d'actions à mener pour atteindre ces objectifs.

La planification est un processus qui programmer des actions et des opérations dans un domaine précis, avec des objectifs précis, avec des moyens précis. Elle vise à affecter les ressources humaines pour chaque intervalle de temps sur un horizon donné.

1.1Les avantages de planification :

- **Gagner en efficacité et en productivité** : un travail mieux structuré, des temps de pause, une vision plus éclairée des tâches et échéances, etc.
- **Se dégager du temps libre** : que ce soit pour être mieux à l'écoute de vos collaborateurs, vous formez ou bien simplement pour souffler et prendre l'air.
- **Se libérer d'une dose de stress** : un esprit libéré du poids des incertitudes, une charge mentale diminuée.

1.2Les types de planification :

1.2.1La planification stratégique :

La planification stratégique est un processus permettant de déterminer la situation actuelle de l'entreprise et les ambitions de ses dirigeants en ce qui concerne l'avenir, puis à partir l'établir un plan d'action qui tient compte des occasions et des menaces de l'environnement ; des forces et des faiblesses de l'entreprise.

1.2.2La planification opérationnelle :

La planification opérationnelle consiste à éclater les objectifs en sous objectifs spécifiques et à attribuer à chaque centre de responsabilité les sous objectifs que doivent réalisez, ainsi que le rôle et la responsabilité qu'il doit suivre dans la réalisation des objectifs globaux et la mise en œuvre de la stratégie retenue.

Pour chaque domaine d'activité : selon la fonction, la division, le plan opérationnelle doit comporter :

- L'objectif à atteindre.
- Les moyens à mettre en œuvre pour réaliser l'objectifs.
- Le programme a appliqué suivant un calendrier de réalisation précis.

2.C'est quoi un planning :

Un planning est un programme établi à l'avance qui indique les diverses tâches à accomplir au cours d'une période déterminée, selon les jours, les heures. C'est la répartition d'activités diverses sur un espace de temps défini

Un planning est un élément primordial à la réussite d'un projet. Le planning comme un guide qui vous montre le chemin de A (existant) vers B (résultat) pendant un temps déterminé. Ce guide définit alors les différentes activités à réaliser pour aller vers B. Il les ordonnance et en contrôle la bonne exécution.

Il permet d'attribuer les ressources en fonction des tâches à effectuer et au délai imparti pour chaque activité. Il prend en compte les contraintes – Exemple : La réception des matériaux

doit impérativement être terminée à telle date pour en contrôler la conformité puis enclencher leur transformation.

Les plannings peuvent être utilisés pour planifier les horaires de présences des personnels ou les tâches effectuées par les personnels :

3.Types de planning :

3.1Planning des horaires de présence :

Ce type de planning est utilisé pour prévoir les horaires de présence du personnel sans préciser les tâches journalières à effectuer soit pour des raisons de sécurité, soit pour une meilleure souplesse.

3.2Planning des tâches :

Ce type de planning est utilisé dans les entreprises à haute technicité, comportant plusieurs métiers et compétences distincts, où il est souhaitable d'affecter le personnel en fonction des tâches. Ce qui exige une décomposition fine des opérations et le repérage des tâches que chaque personne est capable d'accomplir.

4.Domaines d'application :

4.1Plannings dans le domaine de la santé :

Les plannings dans les domaines de la santé sont des calendriers de travail où figurent à la fois le temps, et l'affectation des personnels (jours et horaires de travail, congés et repos). Ils sont établis au niveau de chaque équipe, ils sont à la fois une tâche, un document d'organisation du travail, et un élément contribuant à la gestion administrative du personnel. Cette tâche est parmi les plus difficiles et les plus délicates. Difficile parce qu'elle repose sur la recherche de solutions combinatoire, répond à des contraintes multiples, remise en cause de manière fréquente par l'absentéisme et délicate car elle impose toujours une négociation avec les acteurs (médecins, infirmiers) de l'équipe et la direction du service de soins et l'administration. Les documents établis sont des calendriers sur lesquels on inscrit les affectations des médecins et des infirmiers ; ils sont généralement des tableaux à double entrée avec en ligne le personnel et en colonne le temps. (12)

4.2Plannings dans le domaine de transport :

Le transport est une activité complexe qui fait intervenir des investissements lourds, du personnel qualifié et une informatique très coûteuse. En effet, dans le transport routier, il est toujours nécessaire de gérer aux mieux les ressources existantes en optimisant les investissements. Comme les clients exigent toujours plus de flexibilité, il faut offrir des services sur mesure, replanifier en permanence et en temps réel et gérer le personnel qualifié qui est une opération très complexe car il faut tenir compte de plusieurs contraintes (contrats, temps de travail, pénurie du personnel qualifié, …). (12)

4.3Plannings dans le domaine de la pédagogie :

La confection d'horaires (ou confection d'emploi du temps) dans les établissements scolaires est un travail très important, difficile à réaliser, c'est typiquement un problème de résolution de contraintes, NP-complet, dont la solution n'est pas, a priori, connue dans le cas général

Fournir une solution nécessite d'être capable de s'adapter aux changements dynamiques de l'environnement en tenant compte de la diversité des contraintes telles que l'interdépendance des programmes d'enseignement, la multitude des matières étudiées et les contraintes sur ces matières (cours, cours magistraux, TD, TP…), la durée des cours, les contraintes de disponibilité des enseignants, la disponibilité limitée des salles. C'est un problème qui peut être défini comme un problème qui fait assigner quelques évènements dans un nombre limité de périodes. Il peut être divisé en deux catégories principales : la confection d'horaires des cours et la confection d'horaires des examens. (12)

5.Problème d'emploi du temps universitaire :

5.1Le problème d'emploi du temps :

Est l'un des problèmes de calcul difficile dans l'ordonnancement. Dans le processus de génération d'emploi du temps selon le type de l'emploi à générer le but est de trouver des créneaux horaires adaptés à un nombre de tâches nécessitant des ressources limites. Les objectifs sont différents selon les contraintes variées.

5.2 Problème d'emploi du temps universitaire :

Problème d'emploi du temps universitaire, c'est un problème difficile à réaliser, Sa résolution consiste à ordonnancer les tâches d'un ensemble d'enseignants et de groupes appartenant à des sections en leur allouant un ensemble de locaux et en leur fixant des créneaux horaires.

C'est un problème de résolution de contraintes Pour réaliser une solution nécessite d'être capable de s'adapter à la diversité des contraintes telles que l'interdépendance des programmes d'enseignement, la multitude des matières étudiées et les contraintes sur ces matières (cours, TD, TP…), la durée des cours, les contraintes de disponibilité des enseignants, la disponibilité limitée des salles.

cours dans l'espace de temps en satisfaisant un ensemble de contraintes.

La génération automatique d'emploi du temps universitaire peut être décrite comme l'allocation des ressources (Créneaux Horaires, Salle …) aux événements (Séances de cours, TD et TP), tout en essayant de satisfaire un ensemble de contraintes. Le cœur du problème réside dans les contraintes qui existent au sein de chaque ressource et entre les ressources mêmes. Il existe différentes approches et méthodes pour résoudre le problème d'emploi du temps. La solution d'un problème de satisfaction de contraintes est l'affectation de toutes les variables à des valeurs de telle sorte que toutes les contraintes soient satisfaites.

6.Travaux connexes :

6.1Réalisation d'une application pour la génération automatique d'emploi du temps : (13)

Le mémoire d'université l'arbi ben Mehdi OEB (2016/2017) intitulé par réalisation d'une application pour la génération automatique d'emploi du temps universitaire Réaliser par l'étudiant Kahil Mostafa sadek. Propose une approche pour la résolution de problème d'emploi du temps basé sur le problème de satisfaction des contraintes il utilise la programmation logique avec contraintes pour résoudre ce problème d'emploi du temps.

La méthode de résolution utilisé : *(13)*

La stratégie utilisée dans ce travail pour résoudre le problème d'EDT est une méthode distribuée (DCSP).

Une telle stratégie satisfaite au maximum les contraintes. Car elle prend en considération toute sorte de conflits y compris ceux qui peuvent s'introduire dans le cas où il y a beaucoup de ressources partagées entre les départements.

La programmation logique par contraintes offre une résolution efficace aux problèmes de satisfaction des contraintes, Elle représente aussi un moyen très efficace à la résolution des problèmes combinatoires et notamment le problème génération d'emploi du temps.

Langages de programmations et environnement utilisé *: (13)*

Pour réaliser leur application ils ont utilisé trois langages de programmations :JAVA, SQL et PROLOG avec une interopérabilité entre eux .JAVA a été choisi pour se charger de l'interaction homme machine, le SQL traite les requêtes venues de java pour les opérations sur la base de données. PROLOG s'occupe de la résolution après la récupération de toutes les données requises.

DESCRIPTION DE L'APPLICATION : *(13)*

- L'application contient deux espaces : espace responsable du rectorat et espace responsable du département.
- Pour qu'un utilisateur puisse accéder à son espace, il doit d'abord se connecter via ses coordonnées.
- Après la connexion, chaque utilisateur est dirigé vers son espace.
- La tache génération d'emploi du temps est effectuée par le responsable du département.

6.2Métaheuristique à base de populations appliquée à la génération d'un emploi du temps d'un département à l'université : (14)

Présenté par BOUKELMOUNE Houssam de Université Mohammed Seddik Ben Yahia de Jijel Faculté des Sciences Exactes et Informatique Département d'Informatique Promotion 2021.

Dans ce travail ils ont résolu le problème de l'emploi du temps à l'aide des algorithmes génétiques, en utilisant l'algorithme proposé par Khonggamnerd &Innet, 2009.

Description de l'algorithme génétique : (14)

Les étapes de l'algorithme génétique développé sont :

1. Création d'une solution initiale (population initiale) présentée par des chromosomes.
2. Définir la fonction de fitness et calcul de la valeur de fitness de chaque chromosome.
3. Sélection de deux chromosomes parents P1 et P2 qui possèdent la plus grande valeur de fitness.
4. Faire un croisement des parents choisis pour créer une nouvelle génération (enfant).
5. Faire une mutation des gènes dans le chromosome enfant créé qui possède la meilleure valeur de fitness.
6. Continue le processus jusqu'à ce que la valeur de fitness soit satisfaite ou que l'utilisateur mette fin au processus.

La présentation du chromosome de l'AG : (14)

Les chromosomes appliqués par l'AG sont construits en un groupe d'éléments (E) , Les éléments comprennent trois types de données, qui sont enseignant (L), matière (M) et Salle (S). Chaque élément peut être présenté sous la forme : E= {L, M, S}. Un ensemble de E représentera comme éléments constitutifs des chromosomes. Ainsi, E = {E1, …, En} représentera comme éléments d'entrée de l'emploi de temps tandis que chacun de E (1-n) comprendra un ensemble conséquent de L, M et S Chacun d'eux a un sous-ensemble de chacun. Il peut être représenté par L = {L1, L2, L3, …, Ln1}, qui représentera les enseignants de ce semestre. De la même manière qu'un ensemble de matières et un ensemble de salles dans ce semestre seront comme : M = {M1, M2, M3, …, Mn2} et S = {S1, S2, S3, …, Sn3}, respectivement. Un chromosome sera enchaîné suivant les séquences de périodes qui donnent la longueur de ce chromosome comme indiqué dans l'équation.

$$lChrom = G * P(n) \quad (4)$$

Tels que :

lChrom : La longueur ou le nombre de bits de chaque chromosome,

G : Nombre de Groupe et de section d'étudiants.

P(n) : Nombre de périodes dans l'emploi du temps.

groupes	Groupe1				Groupe2				
indices	0	1	....	29	30	31	....	....	59
périodes	1	2	....	30	31	32	....	....	60
Chrom	E1	E3		E10	E15	E11			E17

La fonction de fitness : (14)

La fonction fitness est utilisée pour mesurer la pertinence de l'emploi du temps et choisir le meilleur. La Fonction de fitness est construite à partir des contraintes de l'emploi du temps. La fonction de fitness que nous avons utilisée est donnée par l'équation :

$$\frac{1}{\sum \text{contrainte dur}e + \sum \text{contrainte souple} + 1}$$

Opérateurs génétiques : (14)

Une application AG est un processus pour améliorer et développer les chromosomes des meilleures solutions en sélectionnant le chromosome qui a la meilleure valeur fitness. Ensuite, un tel chromosome subi le processus génétique, qui comprend deux opérateurs génétiques : le croisement et la mutation. Le processus de croisement commence par deux chromosomes initiaux appelés parents et produit deux enfants. Les données de L, M et S ont été intégrées au niveau de l'élément (l'évènement) de l'emploi du temps. Le processus de croisement et de mutation dans l'emploi du temps sera exécuté à l'intérieur de chaque chaîne de chromosomes, occupée par des groupes d'étudiants.

6.3Approche DCSP basée sur backtrack des DCSP pour obtenir I 'emploi du temps : (13)

Cette approche proposée parThierry Moyaux, Brahim Chaib-draa, et Sophie D'Amours est basée sur le retour en arrière (backtrack) dans le CSP pour obtenir I 'emploi du temps recherché de façon distribuée.

Nous allons voir comment en est formulé l'algorithme proposé.

Formulation du DCSP : (13)

Le problème a été formalisé par l'affectation des enseignants et des salles à des cours tout en respectant des créneaux horaires prédéfinis. (13)

Ils ont défini au préalable le nombre de jours et le nombre de cours par jours. Ensuite, supposé que les enseignants ne soient pas toujours disponibles, il fallait donc préciser les jours et les créneaux horaires de disponibilité pour chaque enseignant. ils ont défini aussi un nombre de cours hebdomadaire limité dont il est interdit de dépasser. Enfin, chaque cours concerne un ou quelques enseignants et pas tous, il fallait spécifier pour chaque enseignant les cours dont il puisse se charger. (13)

Le problème de génération d'emploi du temps est défini par cinq ensembles :

Problème = {Enseignants, Salles, Cours, Horaires, Contraintes}

- ✓ Enseignants = {El, E2, …, Ec} est l'ensemble des e professeurs ;
- ✓ Salles= {Sl, S2, …, Ss} est l'ensemble des s salles ;
- ✓ Cours = {Cl, C2, Cc} est l'ensemble des c cours à donner durant la session considérée ;
- ✓ Horaires= {HI, H2, …, Hh} est l'ensemble des h créneaux horaires disponibles dans une semaine ;
- ✓ Contraintes : est l'ensemble des contraintes entre les variables des quatre ensembles précédents.

Les auteurs de ce travail ont vu que le problème d'emploi du temps peut être défini par une matrice binaire, mais un problème se pose lorsque la notion de distribution entre enjeux : I 'accès concurrentiel aux variables critiques (les locaux par exemple). C'est pour cela qu'ils ont proposé de définir pour chaque agent une matrice qui ne porte que les informations auquel sont relatives. Ils ont défini trois types de matrices : (13)

$XE_i[S_j][H_k][C_l] = X[E_i][S_j][H_k][C_l]$: L'enseignant $E_i$ne connait que le contenu XE_i extrait de la matrice X (le contenu qui le concerne).

$XC_l[E_i][S_j][H_k] = X[E_i][S_j][H_k][C_l]$: Le cours C_l connait le contenu de XC_l.

$XS_j[P_i][H_k][C_l] = X[E_i][S_j][H_k][C_l]$: La S_j connait le contenu de XS_j.

Algorithme :

L'algorithme proposé parThierry Moyaux, Brahim Chaib-draa, Sophie D'Amours). En raison qu'un enseignant puisse être prioritaire aux autres enseignants, et que certains cours n'étant affectés qu'a cet enseignant puissent être négligés, on a évité que les agents- enseignants se charger de la résolution du DCSP. Par contre en choisissant les agents-cours comme les résolveurs du DCSP, aucun cours ne sera évitable. Même s'il y a des enseignants qui ne sont pas pris, c'est un problème moins difficile que ce des cours, vu que l'obligation est de planifier tous les cours prévus. (13)

La figure ci-dessous décrit le processus suivi par les agents pour générer l'emploi du temps.

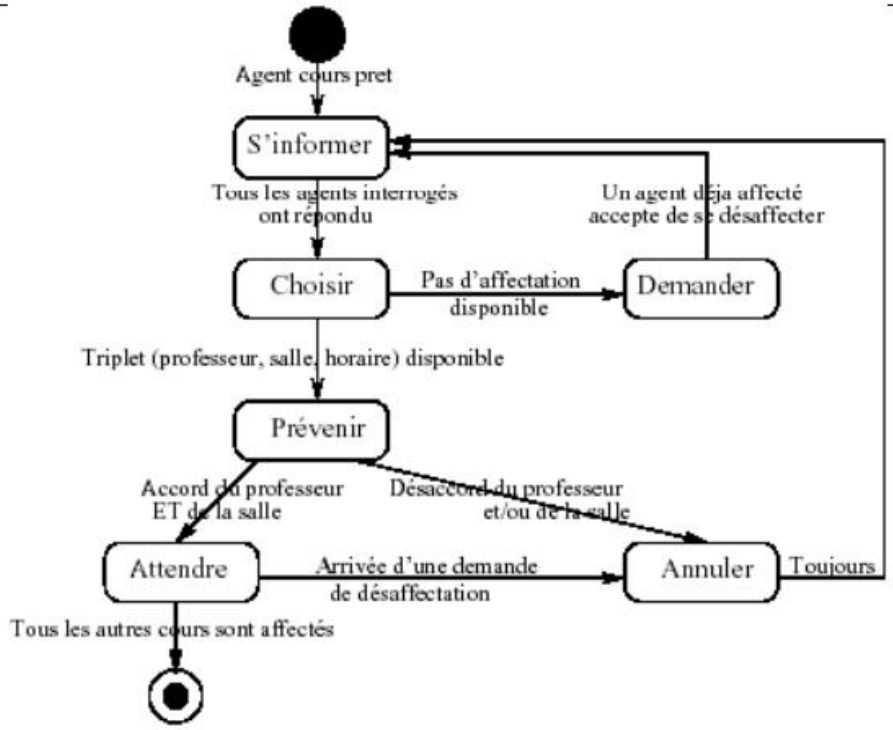

Figure 10: génération d'EDT par les agents cours

Pour chaque agent q, il s'agit de « noircir » (mettre à la valeur « vrai ») les cases de la matrice XCI en fonction de la négociation avec les autres agents-cours et des données provenant des agents-professeur et agents-salle, pour finalement y choisir une case « blanche » (dont la valeur est « faux »). Chaque agent-cours Cl commence par récolter les données des agents-professeur et -salle pour mettre à jour sa matrice XCI (état « S'informer »), puis il cherche dans cette matrice une affectation {professeur, salle, horaire} libre (état « Choisir »). S'il n'en trouve pas, il demande à un autre agent-cours qui a lui-aussi trouvé une affectation de renoncer à cette

affectation (état « Demander »), puis recommence I 'algorithme au début. Si au contraire il trouve une affectation libre, il informe les agents-professeur et agents-salle concernés de I'horaire qu'il aimerait les retenir (état « Prévenir »). Si l'un de ces deux agents lui répond (ou les deux) qu'il ne peut pas, l'agent C1 prévient l'autre agent qu'il ne veut plus de cet horaire (état « Annuler ») et recommence l'algorithme au début. Si au contraire ces deux agents sont d'accord, l'agent C1 note que cette affectation est la sienne, puis il informe les autres agents-cours qu'il s'est trouvé une affectation et se met enfin en attente d'un message provenant d'un autre agent-cours qui lui demanderait de se désaffecter (état « Attendre »). Une fois que tous les agents cours ont averti I 'agent C1 qu'ils ont eux-aussi trouvé leurs affectations, celui-ci termine son exécution (état final). (13)

Conclusion :

Nous avons vu dans ce chapitre la définition planification et on a présenté le planning et leurs différents types pour arriver au problème d'emploi du temps selon la programmation par contraintes. Ensuite nous avons présenté quelques travaux liés à la résolution de ce problème bases sur la programmation par contraintes par différentes méthodes des résolutions pour prendre une sur la résolution de ce problème de génération d'emploi du temps à travers les méthodes de résolution de CSP.

Chapitre 3 : Conception

Introduction :

Afin de bien réaliser notre application il faut réussir à avoir une bonne conception pour faciliter la phase du codage pour ce là il faut d'abord préciser les fonctionnalités que doit fournir notre système, donc nous allons présenter la conception générale et détaillée de tout le système.

Dans ce chapitre nous allons mentionner tous les besoins que nous devrons prendre en considération afin de les satisfaire tous les utilisateurs de l'application, Nous allons présenter par la suite une conception générale et détaillée du système.

I. Présentation générale de notre système :

On doit spécifier ce que doit offrir notre système. Le système doit être capable de réaliser un emploi du temps pour les groupes d'une année universitaire précise qui repend aux besoins.

Pour les séances il y'a trois types qui nous doit réaliser : les séances des cours et les séances TD ou TP.

Les locaux se divisent on deux types : les amphithéâtres pour les séances des cours et les salles pour les TD et TP.

Notre système comporte 3 parties essentiels présentées dans schéma suivant :

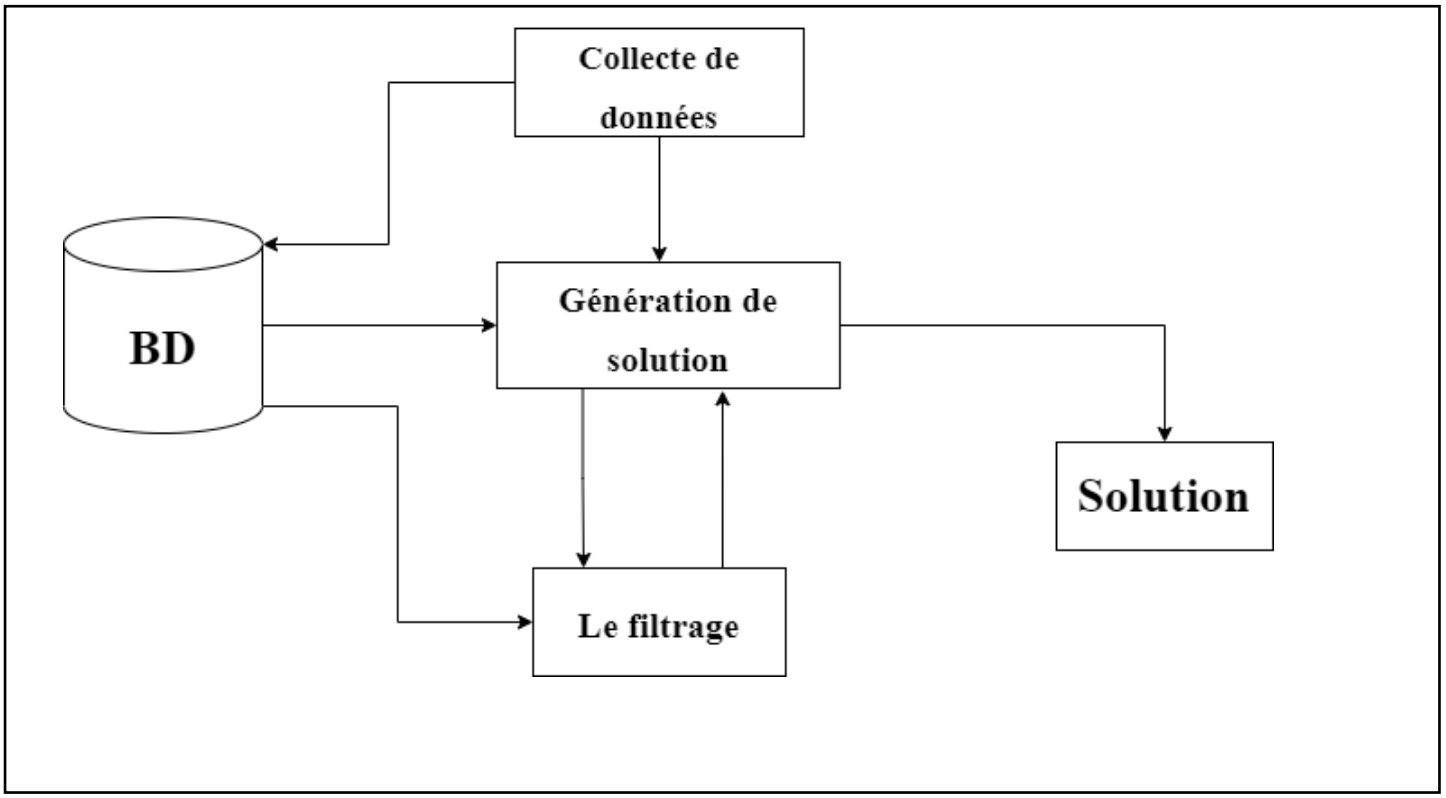

Figure 11: schéma de système

II. Génération de solution :

Notre système Contient un système d'inférence sur un ensemble de règles du système qui sont représenté sous forme de clause de Horne.

Il génère des solutions partielles qui vont être vérifiées par la fonction de filtrage à chaque étape.

1.La base de données :

On a utilisé un ensemble de fichiers texte, chaque fichier représente une table de notre base de données :

- La table enseignant de l'année qui contient : le matricule, nom et prénom, les modules
- La table module de l'année précisé : contient le module, nombre de séance.
- La table salle qui contient la liste des salles.
- La table amphithéâtre qui contient la liste des amphis.
- La table groupe qui contient la liste des groupes.

2.Système d'inférence :

Il s'agit d'un ensemble de règles que nous appliquons sur les tables de données crées pour obtenir un emploi de temps. Donc la fonction d'inférence raisonne-en utilisant les règles suivantes :

- On a cinq jours par semaine pour programmer des séances.
- On a cinq séances par jour.
- Tous les groupes étudient tous les modules du semestre.
- Tous les groupes de la section étudient les séances des cours tous ensemble.
- Les séances des cours dans les amphis.
- On ne peut pas programmer deux même séance pour le même groupe ou la même section l'un après l'autre.
- La troisième séance c'est une séance libre.
- Un enseignant peut enseigner une seule séance à la fois.
- La salle peut être occupée par un seul groupe à la fois.

III. Fonctionnement du système :

Dans le processus de conception du notre système, nous nous somme servi du langage standard de modélisation UML (Unified Modeling Language) ; ce qui offre une vue multidimensionnelle précise sur les différentes parties composant les systèmes.

1.Diagramme de cas d'utilisation :

Nous présentons le digramme de cas d'utilisation qui donne une description de notre application et ces différentes fonctionnalités.

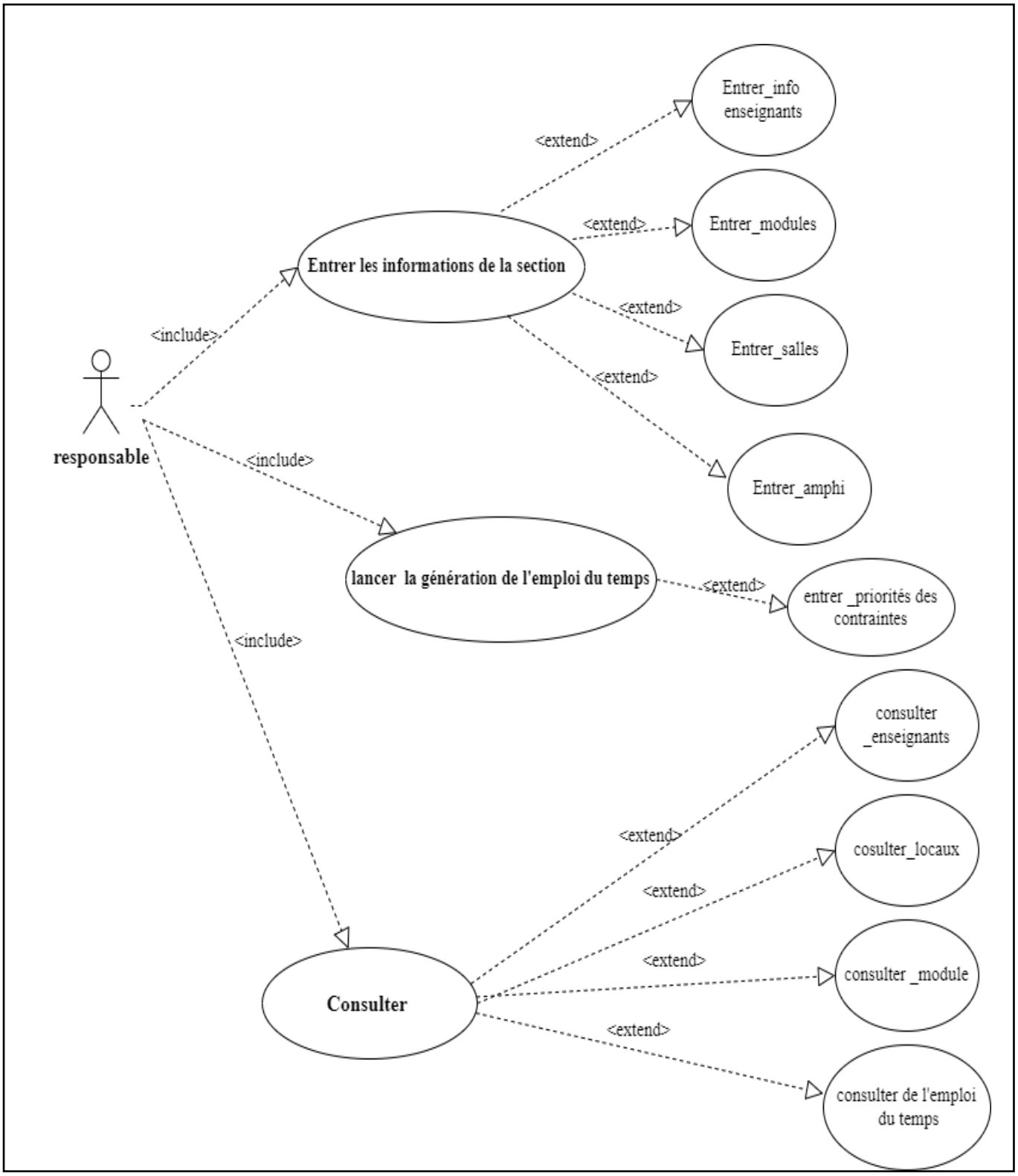

Figure 12:: Diagramme cas d'utilisation

A partir de ce diagramme, on peut résumer les différentes fonctionnalistes fournies à l'utilisateur de notre système.

Nous présentons l'acteur principale de notre système nommé le responsable administratif qui entre les données dont le système a besoin pour générer l'emploi du temps.

2.Diagramme de classes :

Apres l'analyse des cas d'utilisation, nous avons réalisé qu'il y a eu deux types d'objets qui entrent en jeux dans le système : des objets actifs et des objets passifs.

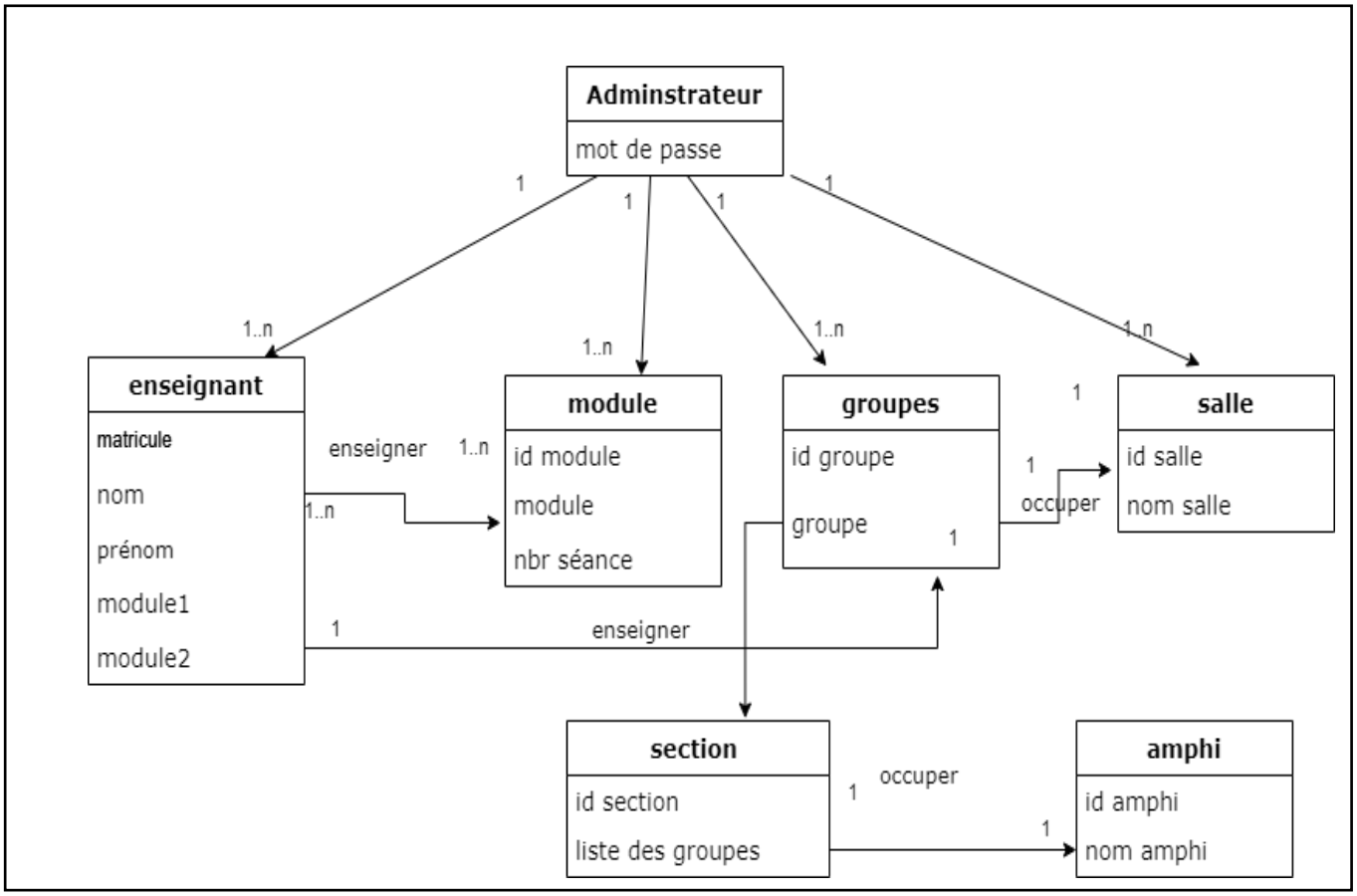

Figure 13: diagramme de class

Les objets actifs sont les objets qui agissent sur l'état du système, Les objets passifs ils ne représentent que des données à utiliser.

Dans notre cas il y a un seul objet actif est le responsable, et toutes les autres classes représentent Les objets passif.

3.Diagramme d'activités :

Les diagrammes d'activités décrivent les actions des différents objets et les flux des processus.

A partir du diagramme de cas d'utilisation, nous avons pu définir les activités de l'acteur.

Ajouter enseignant :

Le responsable doit :

- Se connecter.
- Faire le Choix ajouter un enseignant.
- Remplir le formulaire des informations de l'enseignant.
- L'enseignant sera ajouté.

Le diagramme ci-dessous modélise la description textuelle du processus d'ajout d'un enseignant. De la même façon conçue les autres diagrammes d'activités.

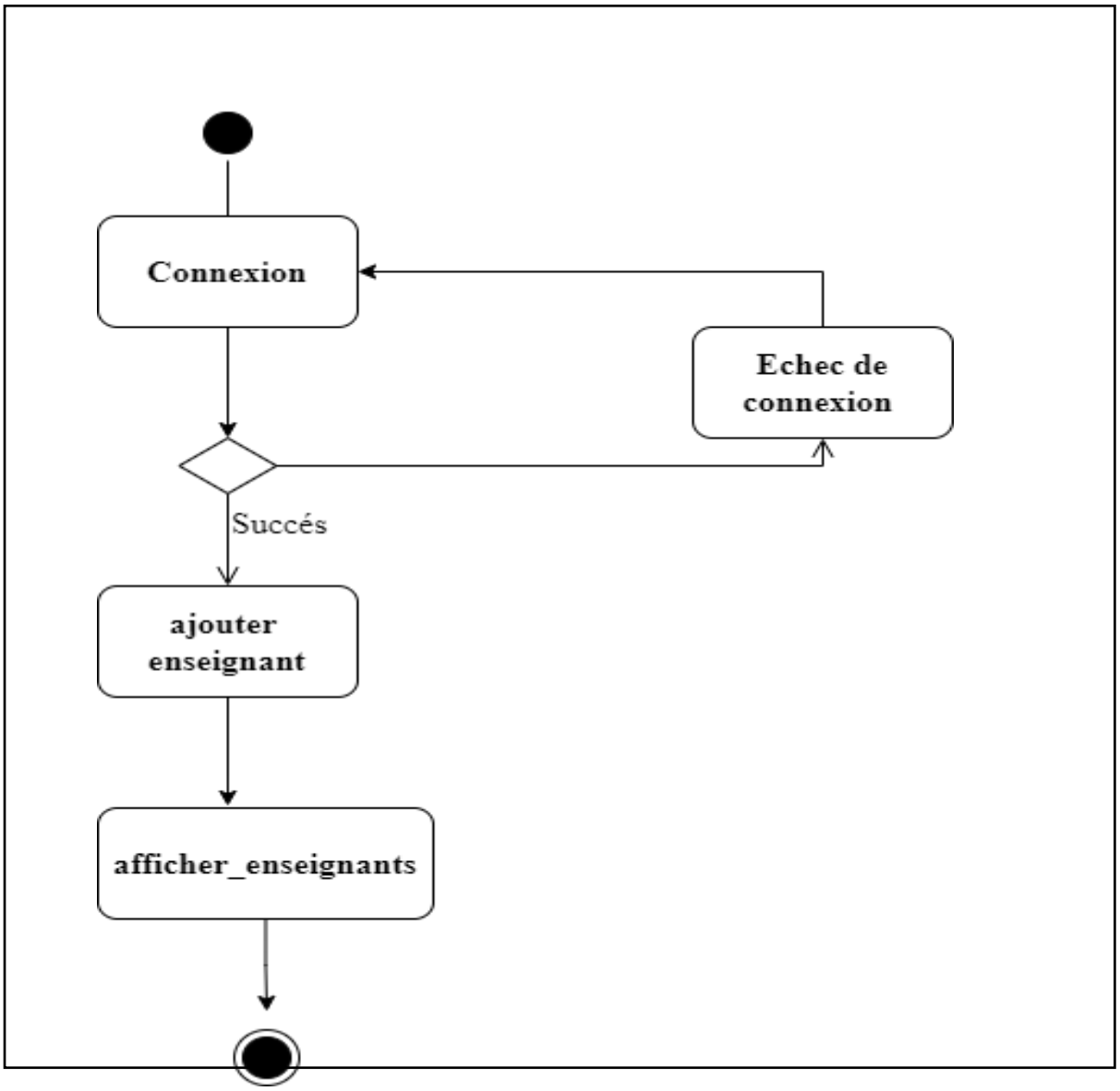

Figure 14: diagramme d'activités : ajouter enseignant

Générer l'emploi du temps :

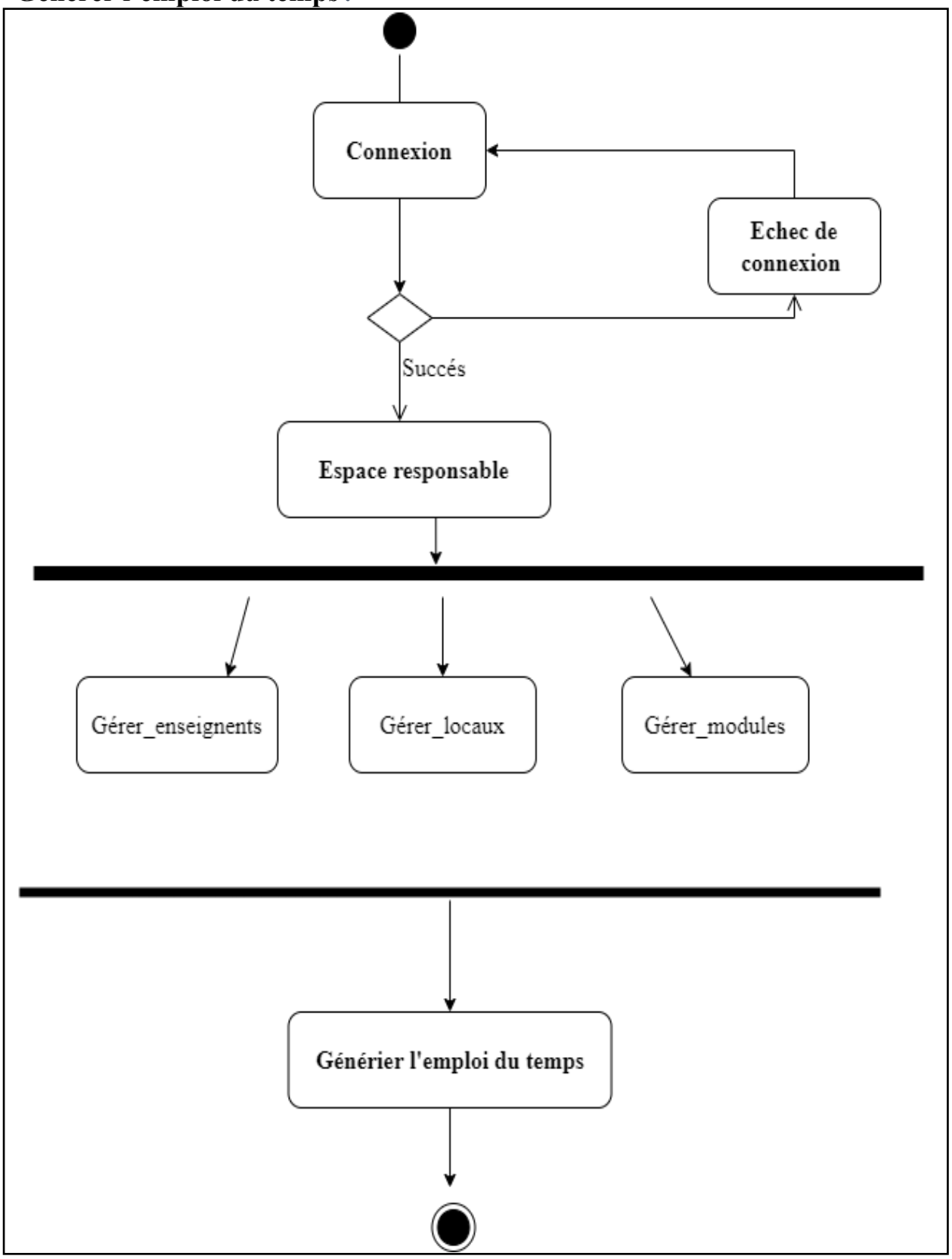

Figure 15: Diagramme d'activités : Générer emploi du temps

Diagramme de séquence :

Les Diagramme de séquence décrit les chemins d'exécution des processus en séquence ; c'est -à-dire : l'ordre des opérations et les interactions entre les différents objets. On peut dire que ce type des diagrammes sert à raffiner toutes les fonctionnalités du système afin de donner une vue un peu proche du codage.

Connexion :

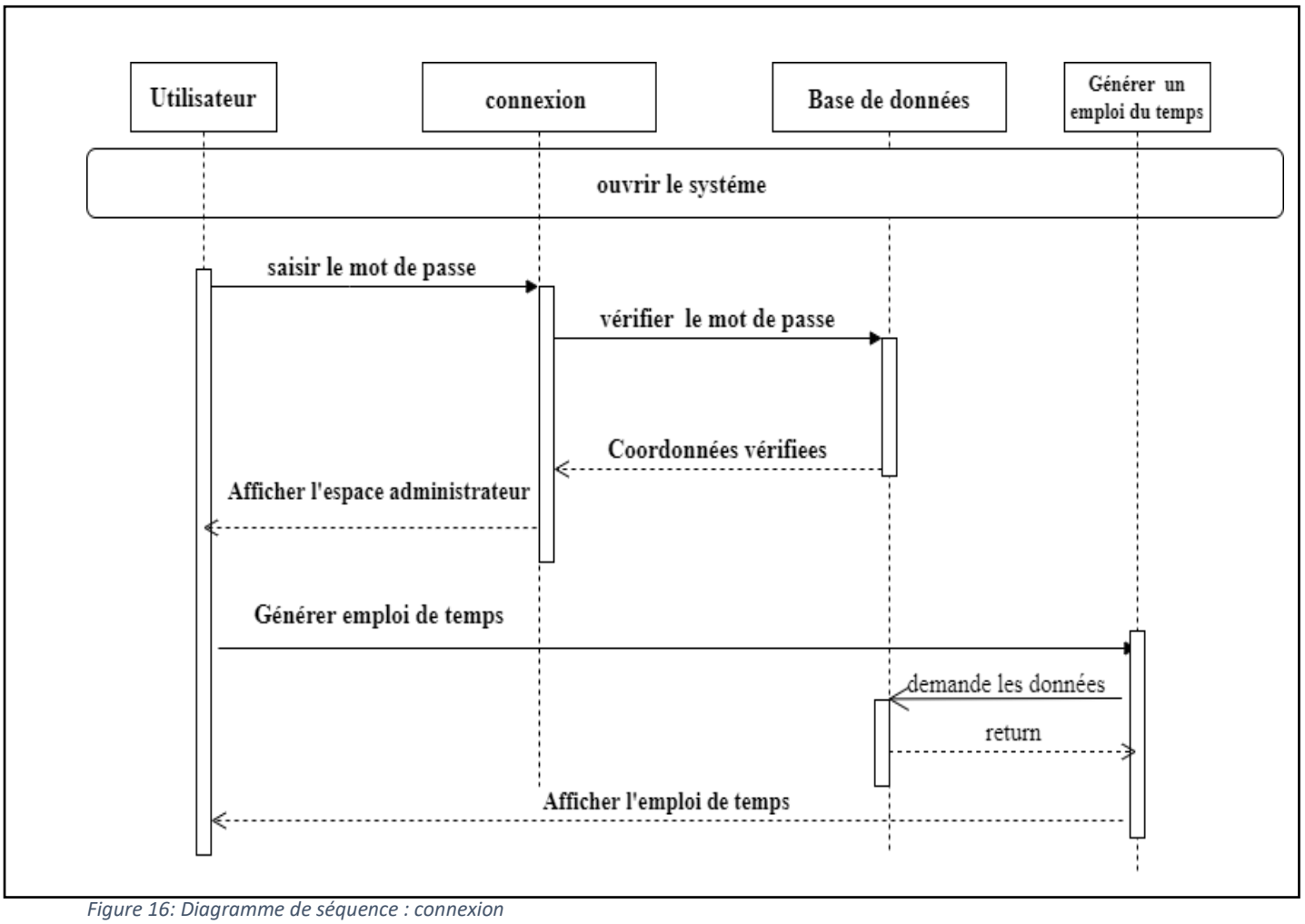

Figure 16: Diagramme de séquence : connexion

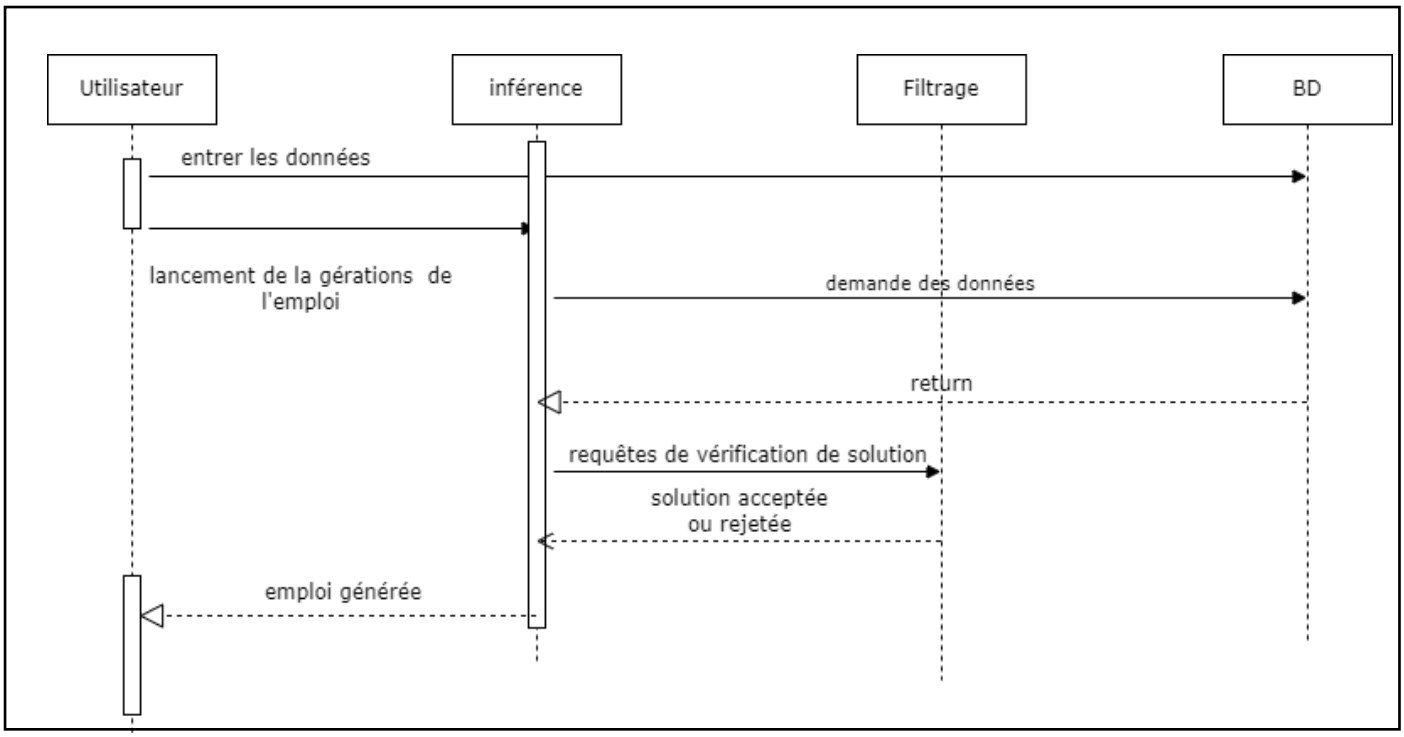

Figure 17: schéma de fonctionnement

Conclusion :

Dans ce chapitre nous avons fait une conception qui va simplifier a nous la phase de codage et exprimer les différents besoins qui nous allons traiter.

Après la partie conception on peut passer à la réalise de notre système qui nous doit présenter dans le chapitre suivant.

Chapitre 4 : implémentation

Introduction

L'implémentation d'un système informatique, désigne sa réalisation et sa mise en œuvre, pour passer à l'interaction avec les utilisateurs. L'objectif de ce chapitre est de présenter les outils utilisés tel que : le langage de programmation et l'environnement de développement utilisé, Nous décrivons le processus de fonctionnement de l'application réalisée.

I. Outils utilisés :

Pour réaliser notre application, nous avons utilisé : le langage de programmation C et les fichiers pour les bases de données avec une interopérabilité entre eux.

I.1 langage de programmation :

C est un langage de programmation impératif généraliste qui offre au développeur une marge de contrôle importante sur la machine notamment sur la gestion de la mémoire et est de ce fait utilisé pour réaliser les « fondations » tel que les compilateurs et les interpréteurs de ces langages plus modernes. (15)

C est un langage de programmation procédural et généraliste. Il est qualifié de langage de bas niveau dans le sens où chaque instruction du langage est conçue pour être compilée en un nombre d'instructions machine assez prévisible en termes d'occupation mémoire et de charge de calcul. En outre, il propose un éventail de types entiers et flottants conçus pour pouvoir correspondre directement aux types de donnée supportés par le processeur. (15)

I.2 Environnement de développement :

Pour développer en C, Nous avons utilisé l'environnement CODEBLOCKS.

Code Blocks : est une excellente option pour la programmation en C. Il s'agit d'un environnement de développement multiplateforme open source intégrée qui supporte l'utilisation de compilateurs multiples comme : GCC. (16)

Pour modéliser notre base de données nous avons utilisé des fichiers textes écrits en utilisant Bloc Note

Bloc Note : Un éditeur de texte, contrairement à un traitement de texte, génère des fichiers ne comportant pas d'informations de formatage ou de styles. Il convient notamment à l'édition de fichiers de configuration, de scripts ou de code source. (17)

II. Description de l'application :

Notre application contient un espace administrateur. Pour qu'un utilisateur accède à l'espace administrateur il doit d'abord se connecter en entrant un mot de passe ce qui garantira la sécurité du système.

```
********************************************** BEINVENUE ********************************
--------un adminstrateur entrez le mot de passe:
---->1234
**************************************************
|pour ajouter un enseignant taper **1**.           |
|pour afficher la liste des enseignants taper **2**.  |
|pour entrer les modules taper **3**.              |
|pour afficher la liste des modules**4**.          |
|pour entrer les groupes taper **5**.              |
|pour afficher la liste des groupes**6**.          |
**************************************************
**********entrez votre choix***********
-->
```

Figure 18:la partie administrateur

Pour ajouter un enseignant il faut remplir les champs de l'enseignant :

```
********************************************************** BEINVENUE ********
--------un adminstrateur entrez le mot de passe:
---->1234
************************************************************
|pour ajouter un enseignant taper **1**.                   |
|pour afficher la liste des enseignants taper **2**.       |
|pour entrer les modules taper **3**.                      |
|pour afficher la liste des modules**4**.                  |
|pour entrer les groupes taper **5**.                      |
|pour afficher la liste des groupes**6**.                  |
************************************************************
**********entrez votre choix***********
-->1
Entrez le nombre des enseignants pour l'ajoutez:
-->1
Entrez les informations d'enseignant numero-1-:::
le matricule:
12
le Nom:ACHOURI
le Prenom:Aya
module 1:M1
module 2:M2
****************************************
```

Figure 19:remplir les champs de l'enseignant

```
********************************************** BEINVENUE **********************
--------un adminstrateur entrez le mot de passe:
---->1234
*****************************************************
|pour ajouter un enseignant taper **1**.              |
|pour afficher la liste des enseignants taper **2**.  |
|pour entrer les modules taper **3**.                 |
|pour afficher la liste des modules**4**.             |
|pour entrer les groupes taper **5**.                 |
|pour afficher la liste des groupes**6**.             |
*****************************************************
**********entrez votre choix************
-->2
************La liste des enseignants*************
-------------------------------------------------------------------------
|id: 1 ,le Nom:KAZAR  ,Prenom: Okba  ,module 1:ihm  ,module 2: 0 |
-------------------------------------------------------------------------
|id: 2 ,le Nom:MEADI  ,Prenom: Nadjib  ,module 1:comp  ,module 2: 0 |
-------------------------------------------------------------------------
|id: 3 ,le Nom:HMIDI  ,Prenom: RRRR  ,module 1:tdcomp  ,module 2: tpcomp |
-------------------------------------------------------------------------
|id: 4 ,le Nom:BOUKHLOUF  ,Prenom: Djemaa  ,module 1:se2  ,module 2: 0 |
-------------------------------------------------------------------------
|id: 5 ,le Nom:BAHI  ,Prenom: Naima  ,module 1:tdse2  ,module 2: 0 |
-------------------------------------------------------------------------
|id: 6 ,le Nom:ayach  ,Prenom: ayach  ,module 1:proba  ,module 2: tdproba |
-------------------------------------------------------------------------
|id: 8 ,le Nom:ZERARKA  ,Prenom: Med  ,module 1:tga  ,module 2: tdtga |
-------------------------------------------------------------------------
|id: 7 ,le Nom:djeffal  ,Prenom: hamid  ,module 1:algo  ,module 2: 0 |
-------------------------------------------------------------------------
|id: 12 ,le Nom:ACHOURI  ,Prenom: Aya  ,module 1:M1  ,module 2: M2 |
***********************************
 pour faire une autre operation taper**1**
-->
```

Figure 20:Afficher listes des enseignants

Générer et afficher l'emploi du temps :

Apres que notre raisonneur effectue son inférence sur l'ensemble des règles du système en prenant en considération les paramètres d'entrée d'un côté et en faisant appel à la fonction de filtrage qui prend en charge la satisfaction des contraintes selon leurs priorités d'un autre coté l'emploi du temps résultant est affiché comme suit :

```
****************** Table des Contraintes ************
 Contrainte A | Vrai
 Contrainte B | Vrai
 Contrainte C | Vrai
********************* L'emploi du temp final *************
```

	8:00-9:30	9:40-11:10	11:20_12:50	13:10-14:40	14:50-16:20
Dimanche	ihm KAZAR A1	comp MEADI A2	**********	tdse2 BAHI G1 s5 tdtga ZERARKA G3 s1	se2 BOUKHLOUF A3
Lundi	se2 BOUKHLOUF A3	tdse2 BAHI G2 s5 tdtga ZERARKA G4 s1	**********	tdse2 BAHI G3 s5	tdse2 BAHI G4 s5
Mardi	tdcomp HMIDI G1 s6	tdcomp HMIDI G2 s6	**********	tdcomp HMIDI G3 s6	tdcomp HMIDI G4 s6
Mercredi	proba ayach A3	tdproba ayach G1 s8	**********	tdproba ayach G2 s8	tdproba ayach G3 s8
Jeudi	tdproba ayach G4 s8	tga ZERARKA A3	**********	tdtga ZERARKA G1 s10	tdtga ZERARKA G2 s10

Figure 21:emploi du temp qui satisfait toutes les contraintes

Les priorités des contraintes :

Dans notre système nous allons travailler sur les priorités des contraints, donc nous allons donner le droit de changer les priorités des contraintes par l'administrateur :

```
entrez la priorite de la fonction **succsessif** 1 ou 2
2
entrez la priorite de la fonction **raffinage** 1 ou 2
1
****************** Table des Contraintes ************
 Contrainte A | Vrai
 Contrainte B | Vrai
 Contrainte C | Vrai
********************* L'emploi du temp final *************
```

	8:00-9:30	9:40-11:10	11:20_12:50	13:10-14:40	14:50-16:20
Dimanche	ihm KAZAR A1	comp MEADI A2	**********	tdse2 BAHI G3 s5	se2 BOUKHLOUF A3
Lundi	tdse2 BAHI G1 s5 tdtga ZERARKA G3 s1	tdse2 BAHI G2 s5 tdtga ZERARKA G4 s1	**********	se2 BOUKHLOUF A3	tdse2 BAHI G4 s5
Mardi	tdcomp HMIDI G1 s6	tdcomp HMIDI G2 s6	**********	tdcomp HMIDI G3 s6	tdcomp HMIDI G4 s6
Mercredi	proba ayach A3	tdproba ayach G1 s8	**********	tdproba ayach G2 s8	tdproba ayach G3 s8
Jeudi	tdproba ayach G4 s8	tga ZERARKA A3	**********	tdtga ZERARKA G1 s10	tdtga ZERARKA G2 s10

Figure 22 :exemple qui montre le droit de changer les priorités des contraintes

La fonction « successif » :

Cette fonction maintient la contrainte de successivité des séances, son rôle est de trouver les séances du même module de même groupe pour les TD ou TP et de même section pour les cours qui sont programmés les uns après les autres pendant le même jour, et les changer pour éviter ce genre de situation.

Le résultat avant utiliser la fonction « successif » :

Dimanche	8:00 - 9:30	ihm	KAZAR		A1
Dimanche	9:40 - 11:10	comp	MEADI		A2
Dimanche	13:10 - 14:40	se2	BOUKHLOUF		A3
Dimanche	14:50 - 16:20	se2	BOUKHLOUF		A3
Lundi	8:00 - 9:30	tdse2	BAHI	s5	G1
Lundi	9:40 - 11:10	tdse2	BAHI	s5	G2
Lundi	13:10 - 14:40	tdse2	BAHI	s5	G3
Lundi	14:50 - 16:20	tdse2	BAHI	s5	G4

Figure 23exemple d'emploi avant l'utilisation de la fonction «successif.

Le resultat apres l'utilisation de la fonction « successif» :

Dimanche	8:00 - 9:30	ihm	KAZAR	A1	
Dimanche	9:40 - 11:10	comp	MEADI	A2	
Dimanche	13:10 - 14:40	tdse2	BAHI	s5	G1
Dimanche	14:50 - 16:20	se2	BOUKHLOUF	A3	
Lundi	8:00 - 9:30	se2	BOUKHLOUF	A3	
Lundi	9:40 - 11:10	tdse2	BAHI	s5	G2
Lundi	13:10 - 14:40	tdse2	BAHI	s5	G3
Lundi	14:50 - 16:20	tdse2	BAHI	s5	G4
Mardi	8:00 - 9:30	tdcomp	HMIDI	s6	G1

Figure 24: exemple après l'utilisation de la fonction "successif"

La fonction « raffiner » :

Cette fonction détecte les séances qui sont programmé hors délais et chercher une autre place pour cette séance en utilisant la fonction « position » qui travaille pour trouver une place idéale pour cette séance par rapport aux contraintes déjà précisées :

```
void raffiner(Liste *liste){
    if (liste == NULL)
    {
        exit(EXIT_FAILURE);
    }
    Element *lst = liste->premier;
    lst=lst->suivant;
    Element *lp,*lpp,*ll = NULL;
    int j = 0;

    while(lst != NULL && lst->jour >= j){
        j = lst->jour;
        ll = lst;
        lst = lst->suivant;
    }

    if(lst!= NULL)
    {
        while(lst != NULL)
        {   lp = position(liste,lst->groupe,lst->ensei);
            lpp = lst;
            lst = lst->suivant;
            insertionml(liste, lp,lpp);
        }
    }
    ll->suivant = NULL;
}
```

Figure 25:la fonction raffiner

La fonction « position » :

```
Element *position(Liste *liste,char g[20],char ens[20]){
    if (liste == NULL)
    {
        exit(EXIT_FAILURE);
    }
     Element *actuel = liste->premier;
     actuel=actuel->suivant;
     Element *lst,*ll = NULL;
       if (liste == NULL)
    {
        exit(EXIT_FAILURE);
    }
    while(actuel != NULL)
        {
            while(actuel != NULL &&  (strcmp(actuel->groupe," ")==0 || strcmp(actuel->groupe,g)==0 || strcmp(actuel->ensei,ens)==0
                ll=actuel;
                actuel=actuel->suivant;
            }
            lst = actuel;
            int sn = actuel->sceance;
            actuel=actuel->suivant;
            while(actuel != NULL && actuel->sceance ==sn && strcmp(actuel->groupe,g)!=0 && strcmp(actuel->ensei,ens)!=0)
            {
                lst = actuel;
                actuel=actuel->suivant;
                }
            if(actuel != NULL && actuel->sceance !=sn)
                return lst;
            }
    return actuel;
}
```

Figure 26:la fonction « position »

Conclusion :

Dans ce chapitre nous avons montré comment on a réalisé notre système et les outils qu‘on a utilisé, avec une description de l’application ou on a ajouté des captures pour bien expliquer.

Nous avons présenté l’application logicielle de notre système en respectant la modélisation qui a été établie précédemment et ce, pour répondre, aux objectifs déclarés, et Nous avons clarifié le mécanisme de notre flux de travail.

Conclusion générale :

Le problème des emplois du temps universitaires est l'un des problèmes de planification intensivement étudiés. Parmi les méthodes utilisées pour résoudre ce genre de problème, s'impose le paradigme de la programmation par contraintes comme une approche très prometteuse.

Le problème de l'emploi du temps des cours est un processus complexe, c'est un problème d'optimisation combinatoire très difficile à résoudre.

Dans le monde de la programmation par contraintes il y a un grand ensemble de méthodes qui travail pour résoudre les problèmes de satisfaction des contraintes, et chaque méthode est considérée comme une méthode qui donne la solution la plus optimale pour un type particulier de problème. Malgré cela, on ne peut pas nier que chaque méthode a un ensemble d'avantages et d'inconvénients qui et le principal critère de choix d'une méthode pour résoudre chaque type de problème.

Dans ce travail, nous avons utilisé une solution basée sur les problèmes de méthodes de satisfaction des contraintes pour résoudre Le problème de l'emploi du temps avec un système d'inférence sur un ensemble de règles du système qui sont représenté sous forme de clause de Horne, et en affectant des priorités à ces contraintes pour donner une certaine souplesse dans l'étapes de filtrage ce qui donnera de meilleurs résultats.

Nous avons réalisé un système qui permet de générer un emploi du temps en utilisant le CSP ce système prend on compte un ensemble de contraintes et raisonne pour satisfaire ces contraintes pour obtenir un bon emploi du temps.

Quelques perspectives contenant des voies d'amélioration de notre système :

- Envisager un ensemble plus large de contraintes pour obtenir un emploi du temps optimal.
- Permettre à l'utilisateur de créer ses propres contraintes.
- Faire appel aux techniques d'optimisation mathématiques ou encore heuristiques afin de minimiser l'exploitation du temps ou encore les ressources

Bibliographie

1. **Moyaux1, Thierry.** *Satisfaction distribuée de contraintes et son application à la.*

2. **Dib, Mohammad.** *Tabu-NG : hybridation de programmation par .*

3. ***INF4230 – Intelligence Artificielle.***

4. Lambert, Tony. ***Hybridation de méthodes complètes et incomplètes pour.*** **2007.**

5. recherche locale. ***wikipédia.*** **[En ligne] https://en.wikipedia.org/wiki/Local_search?oldid=270402194.**

6. Khedidja, Melle. AMARI. ***Elaboration d'un Algorithme Génétique Hybride pour l'Optimisation des.*** **2012.**

7. BOUZIDI, MOHAMED WASSIM. ***UN ALGORITHME TABOU STOCHASTIQUE POUR LE PROBLÈME DE.*** **2015.**

8. Riff-Rojas, Maria-Cristina. ***Résolution de problèmes de satisfaction de contraintes.***

9. Branimir Sigl, Marin Golub, Vedran Mornar. ***Solving Timetable Scheduling Problem.***

10. NOËL, SÉBASTIEN. ***MÉTAHEURISTIQUES HYBRIDES POUR LA RÉSOLUTION DU PROBLÈME.*** **2007.**

11. Nicolas Barnier, Pascal Brisset. ***Optimisation par hybridation d'un CSP avec un.***

12. OUSSAMA, LAOUBI. ***Application de la programmation par contraintes au .***

13. sadek, kahil moustafa. ***par réalisation d'une application pour la génération automatique d'emploi du temps.***

14. Houssam, BOUKELMOUNE. ***Métaheuristique à base de populations appliquée à la génération.*** **2021.**

15. C (langage). [En ligne] https://fr.wikipedia.org/wiki/C_(langage).

16. [En ligne] https://code-blocks.fr.uptodown.com/windows.

17. Bloc-notes (Windows). [En ligne] https://fr.wikipedia.org/wiki/Bloc-notes_(Windows).

18. OUSSAMA, LAOUBI. ***Application de la programmation par contraintes au .***