Table des matières

Liste des tableaux

Liste des figures

Liste d'Acronymes et Abréviations

SIA	Systèmes Intelligents et Applications
OT (Operating theatre)	Bloc opératoire
OR (Operating Room)	Salle d'opération
TS (Taboo Search)	Recherche Tabou
MILP (Mixed Integer Linear Programming)	Programmation linéaire avec nombre entier
SMA	Système multi agents
UML (Unified Modeling Language)	Langage de Modélisation Unifié

Introduction général

La planification et l'ordonnancement sont deux phases importantes dans la gestion du bloc opératoire, elles permettent non seulement d'améliorer la qualité des soins des patients mais aussi de contrôler le budget et maximiser la qualité des services tout en maximisant l'occupation des salles d'opérations, minimisant les coûts prévus des opérations chirurgicales et équilibrant la charge de travail des différents chirurgiens, infirmiers, brancardiers, etc.

Elles permettent ainsi de résister aux différents évènements qui peuvent perturber le déroulement des autres opérations tels que : la panne des matériels, indisponibilité des chirurgiens, personnels, etc.

L'arrivé de tels événements peut influencer l'ordonnancement des opérations et éventuellement les décaler, ce qui en résulte l'annulation ou la réalisation de celles-ci pendant les périodes supplémentaires, cela signifie évidemment une surconsommation des ressources que ce soit humaines ou matérielles.

Divers travaux ont été élaborés dans la littérature pour résoudre ce problème. Ces travaux peuvent être classés selon les ressources prisent en compte : salle d'opérations, salle d'hospitalisation, l'équipe d'aide soignants, etc. Ou selon la méthode de résolution qui a été mis en place : Algorithme génétique, Extension de la méthode Hongroise, Branch and Bound, etc. Ou ainsi selon la fonction objectif à minimiser ou maximiser : maximiser l'utilisation des salles d'opérations, minimiser le temps d'attente des patients, minimiser les coûts d'heures supplémentaires dans la salle d'opérations, minimiser le temps d'inactivité entre les interventions chirurgicales, etc.

Dans ce travail, nous nous focalisons sur l'occupation des salles d'opérations en minimisant le temps inexploité de celles-ci, car une salle qui n'est pas utilisé dans une période régulière consomme ainsi du budget. Et minimiser le temps supplémentaire des salles d'opérations vu qu'une heure de temps supplémentaire coûte plus chère qu'une heure régulière.

Nous considérons certaines contraintes telles que : la capacité limite des salles d'opérations, la durée maximale de travail de chaque chirurgien par jour et les spécialités des chirurgiens et salles.

La démarche proposée consiste en deux phases :

- La phase proactive : permettant la construction d'un planning de la semaine en prenant en compte la fonction objectif et les contraintes citées juste en dessus. Ce planning va être généré une fois par semaine à l'aide de la méta heuristique Recherche Tabou.
- La phase réactive : permettant la prise en charge des imprévus quotidien, parmi lesquels, nous citons l'indisponibilité des chirurgiens. Dans cette phase nous nous servirons des systèmes multi-agents pour une communication entre les chirurgiens.

Ce document est décomposé en cinq chapitres :

- Le premier chapitre est consacré à la description du bloc opératoire et les outils permettant sa gestion, il contient ainsi une brève description de quelques méthodes de résolution du problème de planification et d'ordonnancement dans le bloc opératoire et quelques travaux antérieurs.
- Le deuxième chapitre est consacré aux systèmes multi-agents, leurs définitions, leurs types et architectures, les plateformes permettant le développement des agents, etc.
- Le troisième chapitre présente notre problématique et sa formulation mathématique, il contient ainsi la description globale du système.
- Le quatrième chapitre est consacré à la description technique du système, il contient une partie sur la conception et une autre sur l'implémentation.
- Le cinquième chapitre est le dernier, il est consacré aux expérimentations permettant de tester les performances de notre système et les résultats obtenus.

Chapitre 1 : Gestion du bloc opératoire

Introduction

Les établissements hospitaliers représentent un environnement incertain, chaque jour ils reçoivent des cas urgents ou des demandes des chirurgies urgents ou électives, ils gèrent un nombre important des ressources humaines et ils possèdent des matériels d'une haute technicité. Cela s'applique plus dans le bloc opératoire qui est le cœur de l'hôpital et qui reçoit des patients en cas critique chaque jour. La gestion du bloc opératoire doit être efficace pour pouvoir supporter ces situations.

Dans ce chapitre, nous allons commencer par décrire le bloc opératoire, la trajectoire qu'un patient doit suivre pour une intervention chirurgicale et les outils proposés dans la littérature pour une gestion efficace du bloc. Puis, nous allons parler de quelques méthodes qui ont été appliquer sur ce problème de gestion et d'optimisation au niveau du bloc opératoire et nous allons présenter quelques travaux antérieurs qui s'inscrivent dans la même problématique.

1. Bloc opératoire

1.1. Introduction

Le bloc opératoire est le cœur de l'hôpital, il représente l'élément le plus important de l'hôpital que ce soit au niveau du budget qu'il consomme, plus de 10% selon [3], ou au niveau des équipements qu'il possède ainsi que les ressources humaines, ou de sa relation directe ou indirecte avec de nombreuses services d'hospitalisation et spécialités médicales [4].

Pour cette raison, une bonne gestion du bloc opératoire influence directement sur la gestion des autres services et augmente leur qualité.

Ces dernières années, plusieurs travaux ont abordé ce problème de gestion du bloc opératoire afin de gérer les ressources humaines (infirmiers, chirurgiens, personnels, etc.), gérer les salles d'opération, ou définir le séquencement des opérations chirurgicales, etc.

Tous cela pour optimiser l'utilisation des ressources (matérielles, humaines), maximiser l'utilisation des salles d'opération pour accueillir plus des cas chirurgicaux, minimiser les coûts liés à l'utilisation de ses salles, ou tenir compte des événements qui surviennent et qui peuvent influencer le déroulement des opérations dans le bloc opératoire comme : la panne d'un équipement, l'indisponibilité d'un chirurgien, etc. Et donc une gestion efficace du bloc opératoire doit tenir en compte d'une multitude de contraintes [5] : contraintes humaines, contraintes de préférences et contraintes matérielles.

Cependant, la plupart de ces travaux se sont concentré sur la gestion de la salle d'opérations sans la prise en compte des autres phases du processus opératoire dans le bloc.

1.2. Processus opératoire

Avant d'entamer la description du processus opératoire, voyons tout d'abord la composition du bloc opératoire.

Le bloc opératoire est constitué généralement de deux principaux types de salles :

- Salle d'opérations : ce sont des salles servant à pratiquer une intervention chirurgicale, elles doivent posséder de tous les matériels chirurgicales et anesthésique nécessaires pour réaliser ces interventions chirurgicales.
- Salle de surveillance post-interventionnelle(SSPI) : est appelé aussi la salle de réveil, c'est la salle dans laquelle séjournent les patients qui viennent d'être opérés. Elle contient des lits pour accueillir ces patients et chaque lit doit être doté des dispositifs médicaux permettant la surveillance et le traitement de la douleur post-opératoire du patient.

Ces salles présentent les principaux nœuds du trajet opératoire où le patient s'y installe pour une période plus ou moins grande.

Au sein d'un hôpital, un patient passe par un trajet qui peut être soit chirurgical ou non chirurgical [4]. Nous nous concentrons dans ce qui suit sur le trajet chirurgical qui passe obligatoirement par le bloc opératoire.

Ce trajet chirurgical ou processus opératoire est décomposé en trois étapes :

- La phase pré-opératoire : Correspond à la prise en charge du patient jusqu'à son transfert à la salle d'opération le jour de l'intervention, c'est une phase qui a pour objectif de préparer physiquement et psychiquement le patient, ce dernier subit des consultations chirurgicales et anesthésiques ainsi que des examens.

- La phase per-opératoire : Correspond au jour de l'intervention, dès que le patient est transféré par les brancardiers de la salle de réveil à la salle d'opérations jusqu'à ce qu'il sorte de la salle d'opération. Durant cette période le patient sera anesthésié et ensuite opéré par une équipe chirurgicale.

- La phase post-opératoire : Commence dès que le patient sort du bloc opératoire et est transféré soit vers son service d'hospitalisation ou vers l'unité des soins intensifs et réanimation si son état est critique. Cette phase recouvre l'ensemble des soins nécessaires à l'issue de l'intervention.

1.3. Programmation opératoire

La gestion du bloc opératoire n'est plus une tâche facile, elle est très complexe et sa complexité réside dans l'interaction du bloc opératoire avec d'autres services, dans les ressources humaines qui interviennent à savoir les chirurgiens, brancardiers, infirmières, etc.

Le schéma ci-dessous représente la relation du bloc opératoire avec les autres services présenté dans [4]:

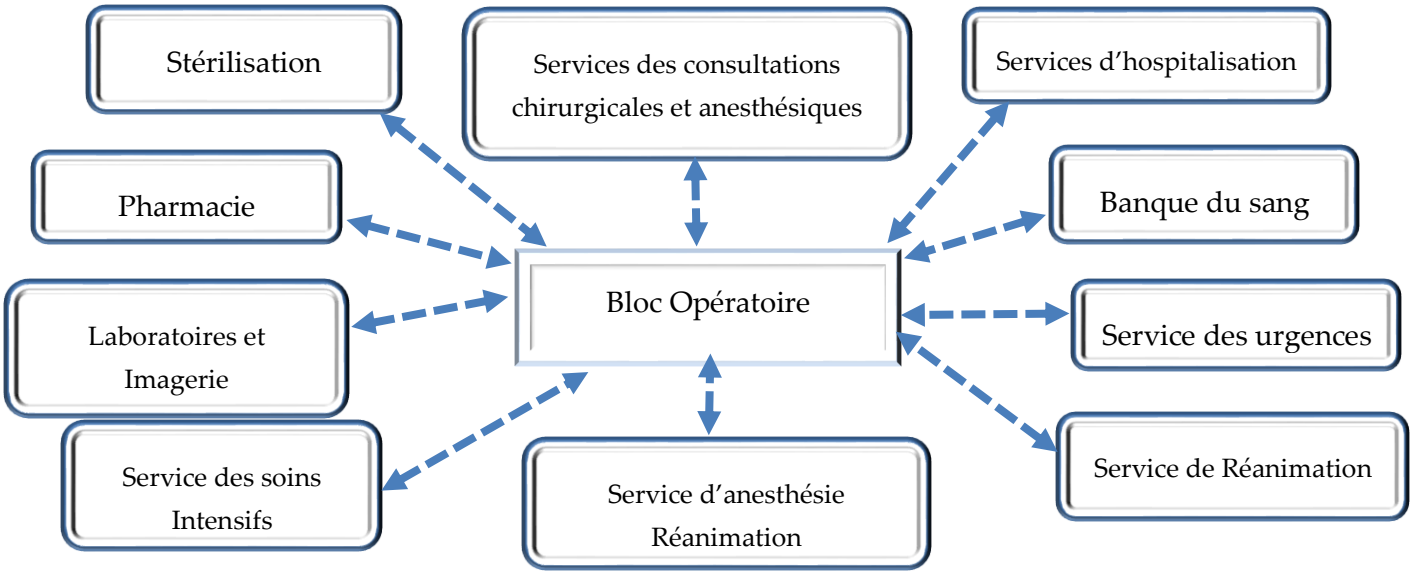


Figure 1: Relation du bloc opératoire avec les autres service

Afin de gérer le bloc opératoire d'une manière efficace, les hôpitaux utilisent la technique de programmation opératoire qui permet d'établir un planning prévisionnel des interventions à réaliser en affectant à chaque patient, qui va subir une intervention chirurgicale, une date d'intervention, la salle d'opérations où il sera opéré, le groupe des chirurgiens qui vont réaliser l'opération, les infirmiers, les matériels nécessaires, etc.

La programmation opératoire se décompose en deux sous problèmes [6] :

- **Planification à l'avance**

Le sujet de la planification à l'avance selon [6] est divisé en deux catégories sur la base de la contrainte primaire considérée lors de la planification des patients :

- Dans la méthode la plus courante de la planification à l'avance, la seule contrainte de la ressource considérée est la durée totale d'ouverture des salles d'opérations.

  Deux approches peuvent être identifiées dans ces méthodes :

  - **Réservation bloquée**

    Consiste à réserver de blocs de temps de salle d'opérations pour des chirurgiens individuels ou des spécialités chirurgicales.

    La durée de chaque bloc de temps est généralement déterminée par le personnel médical de l'hôpital sur la base d'une utilisation passée.

    Le bloc est réservé pour un usage exclusif du propriétaire du bloc jusqu'à un point prédéfini, quand un temps de salle d'opérations inutilisé est mis à la

disposition d'autres chirurgiens, généralement sur la base de la FCFS (premier arrivé, premier servi) [6].

- **Réservation non bloquée**

  Fonctionne selon le principe du premier arrivé, premier servi jusqu'à atteindre le nombre maximum des cas chirurgicales ou bien la capacité de salles d'opérations.

- Dans des méthodes plus complexes de la programmation en avance considèrent en plus de la contrainte précédente plusieurs contraintes de ressources : lits, personnels, infirmiers de la salle d'opérations et équipements.

  Ces méthodes consistent en un ensemble de règles de décision facilement applicables pour contrôler le flux de patients dans et à travers un hôpital.

  Son objectif principal est de maximiser l'occupation du lit de l'hôpital soumis à des contraintes sur le nombre des annulations, le nombre de renvois d'urgence et le nombre des patients programmés en avance.

- **Ordonnancement**

Consiste à définir le séquencement des interventions chirurgicales à réaliser dans les salles d'opérations.

Le programme opératoire selon [7] est l'aboutissement d'une planification et d'un ordonnancement des interventions chirurgicales.

Ces deux tâches peuvent être réalisées de manière séquentielle (hiérarchique) ou de manière simultanée, selon la politique utilisée par l'hôpital [7].

Dans la littérature quatre approches peuvent être identifiées pour construire un programme opératoire :

1.3.1. Programmation ouverte (Open Scheduling)

Consiste à proposer un programme à priori indemne de toute décision antérieure [4]; sans tenir compte des préférences des chirurgiens ou contraintes de placement.

Cette technique a l'avantage d'être simple à organiser. Mais elle peut engendrer des dysfonctionnements [4].

1.3.2. Programmation par pré-allocation de plages horaires (Block scheduling)

Afin d'équilibrer la charge de travail des salles d'opérations et maximiser l'utilisation de celles-ci, il faut sélectionner les jours où les personnels de la salle d'opérations travaillent à plein temps pour réaliser les cas électifs.

Selon [8] une méthode commune pour faire correspondre les cas électifs à la disponibilité du personnel est d'allouer des plages horaires pour les chirurgiens.

Cela revient à réserver une salle d'opérations pour un chirurgien donné de manière exclusive.

Ces plages horaires forment un plan directeur de l'activité chirurgicale ou PDA (Master Surgical Schedule) pour la période d'une semaine ou plus en tenant compte des durées des interventions à réaliser jusqu'à une nouvelle mise à jour [9].

1.3.3. Programmation par pré-allocation avec ajustement des plages horaires (Modified Block Scheduling)

Consiste à allouer les plages non affectées à d'autres chirurgiens en combinant les deux méthodes précédentes (programmation ouverte et programmation par pré-allocation de plages horaires).

Deux techniques sont utilisées :

- Faire une programmation hebdomadaire par pré-allocation d'une partie des plages horaires, et l'autre partie reste commune est gérée par une programmation à l'avance selon le principe premier arrivé, premier servi.

- Si les chirurgiens ou les unités chirurgicales n'utilisent pas suffisamment leurs plages horaires réservées dans une période convenue, ils n'ont plus de priorité sur ces plages horaires et le responsable du bloc peut alors les ajuster pour en faire bénéficier d'autres chirurgiens ou unités chirurgicales [10].

1.3.4. Programmation opératoire basée sur une allocation du bloc opératoire central aux différentes spécialités médicales

Selon [11] il s'agit du même principe que la programmation par pré-allocation, elle consiste à allouer des plages horaires aux différentes spécialités chirurgicales en se basant sur des modèles de prévision de la charge sur des horizons de temps importants (une année).

2. Méthodes de résolution

Dans cette partie, nous allons citer les différentes méthodes qui sont utilisées pour la résolution du problème de gestion du bloc opératoire.

Dans la littérature plusieurs méthodes ont été mis en place pour résoudre ce problème d'optimisation, ces méthodes peuvent être classifiées selon deux grandes catégories : (1) méthodes exactes qui permettent d'avoir une solution optimale mais dans un délai qui peut être beaucoup plus grand selon la difficulté du problème. C'est pour cela que plusieurs

chercheurs ont migré vers d'autres méthodes qui s'adaptent au mieux à ce genre de problèmes en obtenant une solution approchée qui n'est pas forcément optimale mais qui donne un résultat dans un temps minimal, ces méthodes sont appelées des heuristiques (2) ou méthodes approchées.

La figure ci-dessous illustre la classification des méthodes d'optimisation et quelques algorithmes.

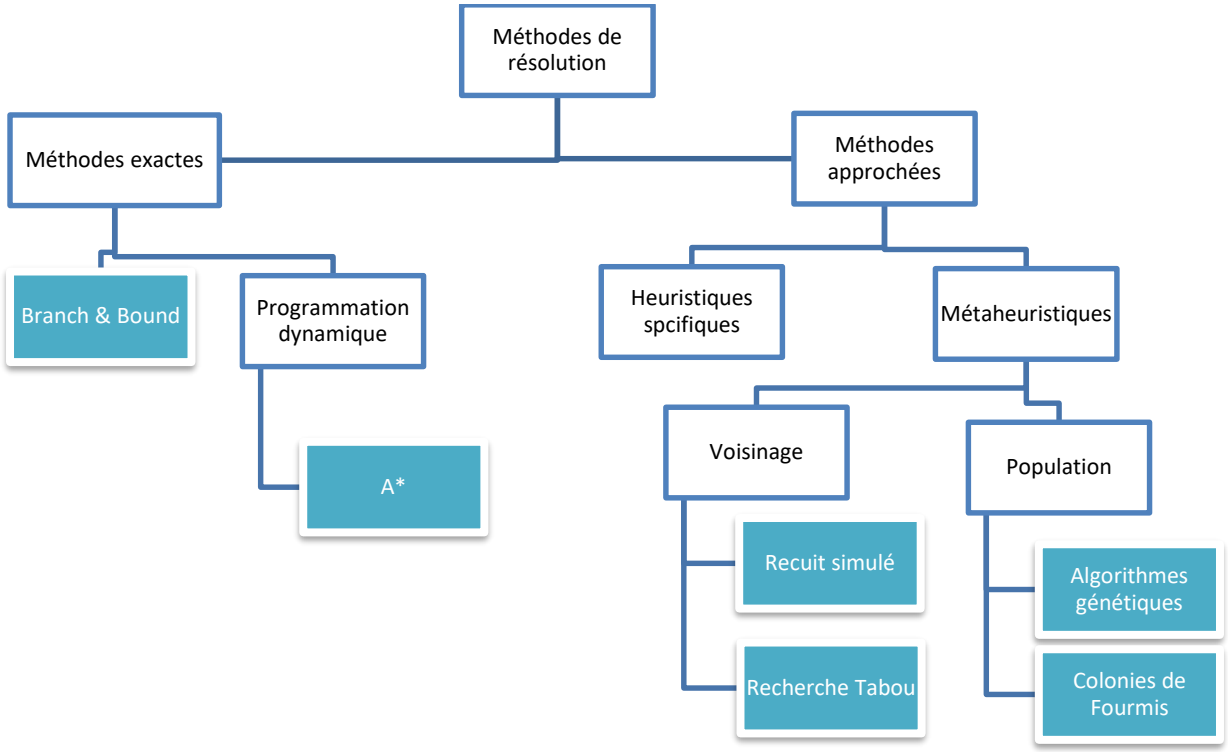


Figure 2: Classification des méthodes d'optimisation

2.1. Méthode exacte (Branch-And-Bound)

Les méthodes exactes sont des méthodes qui garantissent l'optimalité de la solution. Elles demandent un temps de calcul beaucoup plus grand, qui augmente en général exponentiellement avec la taille du problème à résoudre. Et donc, elles peuvent devenir rapidement couteuses en temps d'exécution.

Dans cette catégorie de méthodes nous nous intéressons à un exemple d'algorithme nommé procédure par séparation et évaluation (PSE), ou en anglais branch and bound.

L'algorithme de séparation et évaluation consiste à découper l'espace de recherche en sous-ensembles, chaque ensemble est évalué de façon à déterminer sa borne inférieure. Cette évaluation permettra par la suite d'éliminer des sous-ensembles qui ne contiennent pas des solutions optimales, et garder la meilleure solution. Puis cette solution va être découpée en sous-ensembles et le même processus se répète.

2.2. Méthodes approchées

Contrairement aux méthodes exactes, les méthodes approchées (ou méta-heuristiques) permettent de trouver une solution réalisable qui n'est pas nécessairement optimale dans un temps d'exécution rapide.

L'objectif d'une heuristique [12] est donc de trouver une solution la plus proche possible de celle d'une méthode exacte en se basant sur l'expérience, l'analogie et des résultats déjà obtenus pour optimiser les recherches suivantes.

La qualité d'une méthode approchée va donc se calculer par rapport à l'écart obtenu entre sa solution et l'optimale.

Ces méthodes approchées sont généralement inspirées par des systèmes naturels (la biologie, la physique et l'éthologie). Et elles peuvent être classées en trois catégories : méthodes constructives, méthodes à base de voisinage et méthodes à base de population.

2.2.1. Méthodes constructives

Ce sont des méthodes itératives qui construisent pas à pas une solution. Partant d'une solution partielle initialement vide, ils cherchent à étendre à chaque étape la solution partielle de l'étape précédente, et ce processus se répète jusqu'à l'obtention d'une solution complète.

2.2.1.1. Méthodes gloutonnes (greedy)

Les algorithmes gloutonnes permettent de construire une solution pas à pas à partir d'une solution incomplète. A chaque étape la méthode fait un choix qui semble le meilleur en estimant qu'il conduit à une solution optimale, sans revenir aux décisions déjà prises et les mettre en question.

Les deux méthodes gloutonnes et la programmation dynamique peuvent se représenter sous forme d'un arbre où une solution peut être construite en effectuant une série de choix.

La différence est que pour la programmation dynamique, toutes les solutions sont parcourues et la méthode mémorise les nœuds de l'arbre qui correspondent aux mêmes sous problèmes.

Cependant, pour les méthodes gloutonnes sélectionnent un seul choix qui apparaît le meilleur et traite le sous-problème correspondant et une fois ce choix effectué il n'y a plus de retour en arrière.

2.2.1.2. Algorithme de retour arrière

Un algorithme de retour arrière (*backtracking*) à la différence des méthodes gloutonnes consiste à revenir légèrement en arrière sur des décisions prises afin de sortir d'un blocage.

La résolution d'un problème par cette méthode repose sur la construction d'une solution partielle qu'il va être améliorée afin de s'approcher de plus en plus de la solution finale. Si une

solution partielle ne peut pas être améliorée, elle est abandonnée et l'on revient en arrière pour examiner d'autres solutions possibles.

2.2.2. Méthodes à base de voisinage

Les méthodes de voisinage sont des algorithmes de recherche itératifs qui partent d'une solution initiale qui est obtenue soit de façon exacte ou par un tirage aléatoire et se déplacent pas à pas d'une solution à une autre. Et donc il est nécessaire de définir une relation de voisinage, Selon [12] une relation de voisinage est une application qui associe à toute solution de l'espace de recherche un ensemble de solutions réalisables appelées voisins.

2.2.2.1. Méthode Tabou

C'est une méthode qui consiste à se déplacer d'une solution à une autre en évitant d'explorer des solutions dernièrement visitées, Cela peut se faire à l'aide d'une liste qui garde les dernières configurations réalisées et les marque comme étant des objets tabous et cela pour éviter d'être piégé dans un minimum local. Cette liste est appelée « *la liste Tabou* ».

La méthode Tabou comme étant une méthode à base de voisinage, elle nécessite une solution initiale à partir de laquelle d'autres solutions peuvent être construites pas à pas et qui permettent de minimiser la fonction objectif.

L'algorithme générale de la recherche Tabou est représenté comme suit :

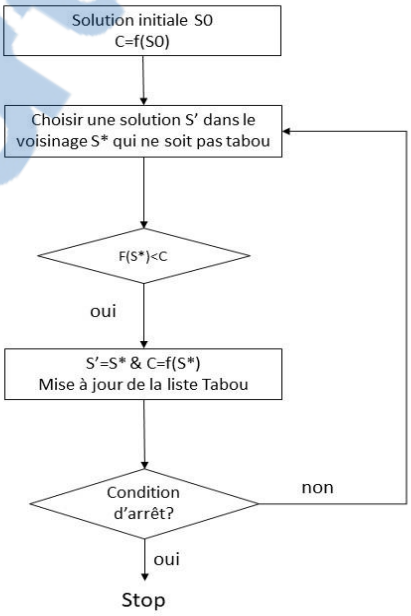


Figure 3: Algorithme générale de recherche Tabou

Avec :

- S0 : la solution initiale
- C : le meilleur coût

- S* : les voisinages de la solution courante
- S' : le meilleur voisin

2.2.2.2. Méthode de recuit simulé

La méthode de recuit simulé est issue d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide, de basse énergie [13].

Elle part d'une solution initiale S0 avec une énergie initiale f(S0) et une température initiale T0 qui doit être élevé pour laisser accepter les mauvaises solutions au départ et elle est diminué au fur et à mesure pour ne pas accepter que les bonnes solutions, à chaque itération la méthode choisie un voisin et calcule la variation de l'énergie, si cette variation est négative cela veut dire que l'énergie se baisse est donc ce voisin est accepté, sinon si la variation est positive ce voisin est accepté avec une probabilité calculé à l'aide de la fonction appelée « *dynamique de Metropolis* ».

L'algorithme de recuit simulé peut être généralisé [14] selon les étapes ci-dessous :

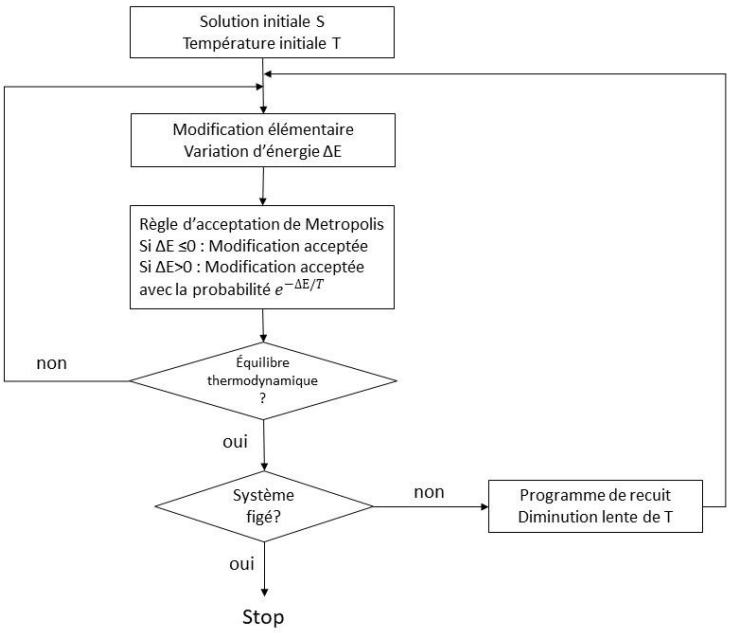


Figure 4: Algorithme de recuit simulé

2.2.3. Méthodes à base de population

Contrairement aux méthodes constructives et de recherche locale qui font intervenir une solution unique les méthodes évolutives (à base de population) manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche.

L'idée centrale [15] consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions distinguables, appelé population, dans le but de guider efficacement la recherche vers de bonnes solutions dans l'espace.

2.2.3.1. **Algorithmes génétiques**

Les algorithmes génétiques sont des méthodes d'optimisation stochastique qui s'inspirent fortement des mécanismes biologiques liés aux principes de sélection et d'évolution naturelle [15] c.-à-d. que la population évolue de génération à une autre pour qu'elle s'adapte au mieux aux contraintes environnementales.

Le principe de l'algorithme est que, à partir d'une population initiale composée de N individus représentant chacun une solution, on construit une nouvelle population composée des individus mieux adaptés. Cela peut se faire à l'aide de certains opérateurs tels que : la sélection, la mutation et le croisement.

La figure ci-dessous représente l'algorithme génétique général :

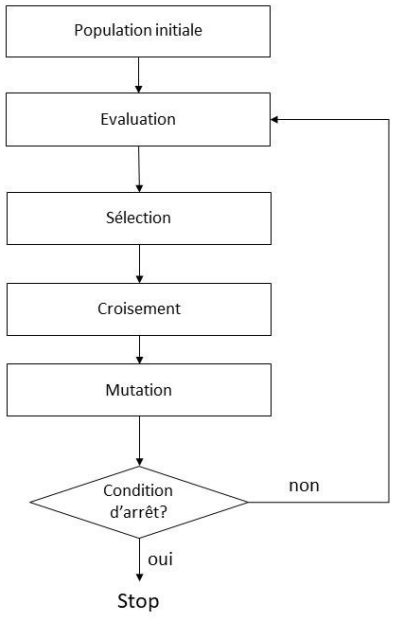


Figure 5: Algorithme génétique

- L'évaluation : cette étape permet de mesurer la qualité de chaque individu à l'aide d'une fonction appelée « *fitness* ».
- Sélection : permet de sélectionner les meilleurs individus et éliminer les autres qui sont moins bons ou mauvais selon leur fitness.
- Croisement : cet opérateur a comme objectif d'enrichir la diversité de la population en combinant les chromosomes des individus parents pour créer des individus enfants.

- Mutation : un autre opérateur qui permet d'enrichir la diversité de la population est la mutation, cet opérateur permet d'appliquer un changement mineur à un chromosome de l'individu.

3. Gestion du bloc opératoire : recueil de la littérature

Dans la littérature, le problème de planification et d'ordonnancement des opérations chirurgicales a pris une grande attention de la part de la communauté scientifique.

Nous constatons dans la plupart de ces travaux que la ressource qui est souvent pris en compte est la salle d'opérations.

[8] utilisent une simulation par ordinateur incluant des différentes méthodes de sac-à-dos pour maximiser l'utilisation de la salle d'opérations, qui est égale à la durée de temps pendant laquelle la salle est occupée sur la durée totale que la salle est disponible et dotée.

[16] utilisent une méthode de Branch-and-Price pour résoudre le problème de planification des interventions sur une semaine en visant à minimiser le coût total d'opération.

Cependant, d'autres auteurs se sont intéressés en plus de la salle d'opérations à la salle de réveil / d'hospitalisation ou la disponibilité des anesthésistes et infirmiers.

[17] Une approche en deux étapes a été définie pour résoudre le problème de la planification du bloc opératoire : les interventions des patients sont affectées aux salles d'opérations à moyen terme en visant à optimiser le coût du temps d'attente des patients, puis ces interventions sont ordonnancées quotidiennement afin d'intégrer plus de caractéristiques (salle de réveil, opérations de nettoyage, etc.) pour la synchronisation des ressources humaines et matérielles.

[18] proposent une nouvelle heuristique *Nouv* qui utilise une variation de l'indice de Palmer appliquée avec les règles NEH et LBM pour l'ordonnancement du bloc opératoire en prenant en compte la disponibilité des lits dans la salle de réveil sans temps d'attente entre la fin de l'intervention et le début de réveil et la disponibilité des brancardiers.

[19] visent à maximiser l'utilisation des salles d'opérations et minimiser les coûts du temps supplémentaires des salles d'opérations ainsi que la salle de réveil selon deux étapes : Tout d'abord, ils ont construit un plan hebdomadaire à l'aide de la méthode de génération de colonnes pour l'affectation des patients aux salles d'opérations à une date et à des chirurgiens en prenant compte des contraintes telles que : la date limite d'affectation des cas chirurgicaux, la capacité limite des chirurgiens. Et ensuite, ils ont fait un ordonnancement quotidien à partir du plan de la 1ère phase pour déterminer le séquencement des opérations chirurgicales du jour à fin d'optimiser les coûts des salles d'opérations et la salle de réveil à l'aide d'un algorithme génétique hybride.

[20] proposent un modèle qui prend en compte la disponibilité des lits dans la salle de réveil ainsi que la disponibilité des transporteurs et pour le résoudre ils mettent en place l'heuristique de relaxation lagrangienne.

[21] proposent un nouvel algorithme pour planifier les cas électifs une semaine avant en prenant en compte l'incertitude des durées des opérations avec, comme objectif de maximiser l'utilisation de la salle d'opérations et minimiser le nombre d'annulations des cas électifs.

D'autres ont pris en compte la disponibilité des chirurgiens ou matériels

[5] développent un modèle qui tiennent compte d'un maximum de contraintes telles que : préférences du personnel, disponibilité de matériels renouvelables et non renouvelables. Ce modèle a été résolu en deux étapes : au premier lieu, les auteurs ont utilisé la recherche en profondeur d'abord pour la génération d'une solution réalisable. Et puis, ils ont intégré une fonction objectif pour la génération d'une solution optimale à l'aide de la méthode Branch-and-Bound.

[22] Comparent différentes méthodes : la programmation linéaire en nombre entiers (PLNE), les trois heuristiques classiques pour le Bin-Packing (First Fit Decreasing, Next Fit decreasing, Best Fit) et le couplage de la première variante (FFD) avec la descente stochastique en tenant compte des tâches de maintenance préventive déjà placées dans les salles, avec comme objectifs de minimiser le nombre des interventions non affectées, de minimiser le nombre de salles-jours utilisés et de minimiser la durée minimum d'occupation des salles.

Mais peu de chercheurs qui ont pris en considération la contrainte relative à la priorité des patients ou les cas urgents.

[23] ont proposé un nouvel algorithme génétique hybride permettant la résolution d'un modèle d'optimisation multi-objectif qui vise à équilibrer l'utilisation des salles d'opérations, maximiser le nombre des opérations chirurgicales planifiées en prenant en compte la priorité de l'opération et le temps d'attente des patients, minimiser la sous-utilisation des salles d'opérations et le coût du temps supplémentaire.

[7] proposent un nouveau modèle stochastique de programmation mathématique en prenant en compte les cas électifs ainsi que les cas urgents. Les cas électifs sont planifiés en avance afin de minimiser les coûts qui y sont liés et les coûts du temps supplémentaire. Cependant, les cas urgents arrivent aléatoirement et doivent être traités le même jour de leur arrivé. Pour résoudre ce modèle, les auteurs combinent une simulation Monte Carlo avec la programmation en nombre entier mixte.

[24] proposent un nouveau modèle d'optimisation RRSM qui vise à minimiser le nombre d'annulations des interventions déjà planifiées et maximiser l'addition des cas urgents en prenant en compte la priorité des patients et les évènements aléatoires qui peuvent perturber

le déroulement du bloc opératoire (l'incertitude du temps estimé pour une intervention, l'arrivé des cas urgents, la panne d'un équipement et l'indisponibilité des ressources.

[25] visent à minimiser le temps d'accès pour les patients avec une priorité clinique élevé en prenant en compte la disponibilité des salles d'opérations et chirurgiens ainsi que la période du temps dans laquelle une opération doit être réalisé. Pour cela, les auteurs comparent entre 83 heuristiques (81 constructives, une heuristique composite et une méta-heuristique).

Le tableau ci-dessous résume quelques travaux réalisés dans ce problème de gestion du bloc opératoire y compris les méthodes utilisées, la fonction objectif et les différentes contraintes prises en compte dans ces travaux.

Articles	Objectifs	Contraintes	Méthodes de résolution
[8]	-Maximiser l'utilisation des salles d'opérations.	- La durée qu'un patient doit attendre pour la réalisation de l'intervention chirurgicale.	Algorithme hybride basé sur la méthode Best Fit Descending with fuzzy constraints (glouton)
[17]	-Minimiser le coût du temps d'attente des patients.	- Chaque chirurgien ne peut réaliser qu'un nombre limité des heures d'interventions par jour. - La date limite des interventions. - La disponibilité de l'équipement chirurgical nécessaire pour chaque intervention. - Interdire l'attribution d'une durée de période plus petite à la durée de l'intervention (pas d'opération majeure pendant les périodes de surtemps).	Extension de la méthode Hongroise.
[18]	-Minimiser la plus grande date d'achèvement.	-Disponibilité des ressources. -Contraintes de précédences.	Nouvelle heuristique *Nouv* identique à l'heuristique NEHLDBM en changeant l'indice de Palmer.
[7]	-Minimiser les coûts du temps supplémentaire prévus ainsi que les coûts liés aux cas électifs (coût d'hospitalisation, coûts du temps d'attente …)	-Un cas électif est affecté une et une seule fois. -Estimer le temps supplémentaire pendant chaque période.	Le problème est formulé par un modèle stochastique. La solution proposée combine la méthode d'optimisation Monte Carlo et la programmation linéaire en nombres entiers.
[24]	-Minimiser le coût de la nouvelle planification. -Minimiser le nombre d'annulations des patients déjà affectées et maximiser le nombre des cas urgents.	- Le nombre de patients planifiés ne doit pas être supérieur au nombre disponible. - Le temps requis pour compléter les interventions chirurgicales affectées est inférieur au temps disponible dans la salle d'opérations.	Modèle RRSM (Robust Reactive Scheduling Model) qui est un modèle d'optimisation

Articles	Objectifs	Contraintes	Méthodes de résolution
[26]	-Minimiser le temps inexploité des plages horaires. -Minimiser les durées d'ouvertures des salles d'opérations.	-Disponibilité des lits d'hospitalisation. -Disponibilité des plages horaires. -Disponibilité des salles d'opération et des lits de réveil.	-Quatre heuristiques : Ordonner les interventions des patients suivant : * la date limite de l'intervention * les règles de gestions SPT (Shortest Processing Time). * la règle LPT (Longest Processing Time) appliquées sur les durées d'interventions. * la règle LPT appliquées sur les durées d'hospitalisation. -Treize heuristiques : Ordonner les interventions des patients suivant les règles de gestions SPT, LPT, MWR (Most Work Remaining) et la règle de Johnson appliquées sur les durées d'interventions et sur les durées de réveil des interventions et des plages horaires.
[5]	-Minimiser la date d'achèvement de la dernière opération (makespan).	- deux opérations ne peuvent pas avoir lieu en même temps dans une même salle d'opérations. - Un chirurgien ne peut pas opérer en même temps dans deux salles différentes. -Chaque opération doit être affectée à une salle et une seule.	Backtracking + Branch-and-bound.
[20]	- Minimiser la date d'achèvement de la dernière opération (le makespan).	-La capacité de l'équipe porteur. -Les contraintes de précédences. - Le temps d'achèvement d'une opération.	Méthode de relaxation lagrangienne.

Articles	Objectifs	Contraintes	Méthodes de résolution
[19]		-Chaque cas est traité exactement une fois avant la fin de la période de planification. -Chaque salle d'opérations peut être occupée par au plus un plan réalisable accepté en un jour. -La durée totale de travail assignée à chaque chirurgien par jour ne peut excéder son temps de travail maximum ce jour. -La durée totale des opérations pour chaque plan ne dépassera pas les heures d'ouverture maximales de la salle d'opérations. -Chaque plan correspond à une seule salle d'opérations disponible pendant la période de planification.	Méthode column-generated-based Et pour la construction de la solution initiale l'utilisation de la méthode Best Fit Descending with fuzzy constraints . Algorithme génétique hybride. (Algorithme génétique+ recherche Tabou).
[22]	- Minimiser le nombre d'interventions chirurgicales non affectées. - Minimiser le nombre de triplets (salles, jour, période) utilisés sur l'horizon. - Minimiser la durée d'occupation de salles.	-Respect de la durée disponible des périodes. -Une intervention chirurgicale n'est affectée qu'à une seule période dans un seul jour.	Descente stochastique DS + First Fit Decreasing.
[25]	- Maximiser le niveau de service d'une spécialité chirurgicale en priorisant les patients ayant des grandes valeurs de poids w.	-Chaque opération est traité une et une seule fois pendant la période de planification. - Date au plus tôt et au plus tard dans laquelle un patient doit être planifié. -S'assurer que la durée totale des opérations de chaque jour ne dépasse pas la capacité régulière de la salle d'opérations ce jour-là. -S'assurer que la durée totale des opérations affecté à un chirurgien dans chaque jour ne dépasse pas sa capacité dans ce jour-là.	83 heuristiques : - 81 heuristiques constructives, - Une heuristique composée, - Une meta-heuristique.
[21]	-Maximiser l'utilisation des salles d'opérations. -Minimiser le nombre d'annulation des patients.	-Chaque patient est planifié une et une seule fois. -Le nombre de patients qui nécessitent des lits pendant le week-end est inférieur au nombre maximum des lits disponibles dans le week-end.	-Algorithme hybride combine des techniques de recherche de voisinage avec la simulation de Monte Carlo, utilisée pour gérer la variabilité des temps de chirurgie. De plus, des algorithmes gloutons sont développés pour l'allocation des heures supplémentaires

Articles	Objectifs	Contraintes	Méthodes de résolution
[23]	-Maximiser le nombre des patients planifiés. -Maximiser la priorité totale des patients planifiés et les patients qui doivent être planifiés durant la période de planification. -Minimiser la sous-utilisation des plages horaires de la salle d'opérations.	-Chaque patient est assigné au plus à un bloc de la salle d'opérations pendant l'horizon de planification. -Chaque groupe chirurgical est assigné au plus à un seul bloc de la salle d'opérations. -Chaque bloc est assigné au plus à un seul group chirurgical. -Equilibrer la charge de travail entre les spécialités chirurgicales. -La durée totale d'une opération ne doit pas dépasser la capacité du bloc de la salle d'opérations plus le temps supplémentaire du bloc.	Algorithme génétique hybride.

Tableau 1: Etat de l'art sur la planification et l'ordonnancement du bloc opératoire

Conclusion

Au cours de ce chapitre, nous avons décrit le bloc opératoire comme étant la ressource la plus importante du milieu hospitalier, et le processus opératoire. Puis, nous avons parlé d'un outil appelé « Programmation opératoire » qui est utilisé pour gérer le bloc opératoire, et nous avons mis le point sur la différence entre la planification et l'ordonnancement. Ensuite, nous avons parlé des méthodes exactes et des méta-heuristiques qui sont utilisés souvent dans ce problème d'optimisation au niveau du bloc opératoire. Après, nous avons présenté quelques travaux de recherche qui ont été élaborés pour la résolution de ce problème, accompagné par un tableau récapitulatif.

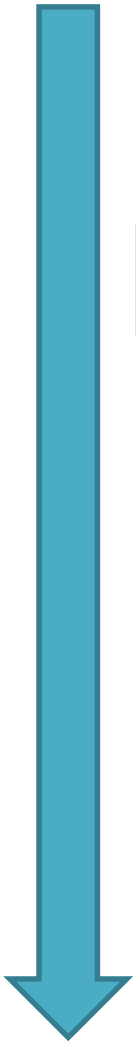
Chapitre 2 : Systèmes multi-agents

Introduction

L'intelligence artificielle distribuée a eu lieu dans les années 80 pour enrichir les approches classiques de l'I.A. En effet, elle s'agit d'un ensemble d'entités distribuées ou agents qui interagissent entre eux pour atteindre un but, ces agents peuvent être non intelligents dans ce cas-là l'intelligence émerge de leurs interactions. Ou ils peuvent être intelligents.

Dans ce chapitre, nous allons voir quelques définitions sur le concept d'agent car de plus en plus, il subit une évolution dans sa définition et cela est dû à son intégration dans des différents domaines de recherche, nous allons voir ainsi ses types et ses caractéristiques. Puis, nous allons parler des systèmes multi agents, les mécanismes d'interactions entre les agents et les modes de communication. Ensuite, nous allons présenter les architectures SMA et leurs domaines d'application. Finalement, nous allons parler des travaux de recherche qui ont été élaborés dans la littérature pour la résolution du problème de gestion d'emploi du temps en général.

1. Historique sur les méthodes de programmation



Langage Machine	Suite de bits qui est interprétée par le processeur de l'ordinateur lors de l'exécution d'un programme. C'est le langage natif du processeur.
Assembleur	Langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
Programmation procédurale	Un paradigme qui se fonde sur le concept d'appel procédural. Une procédure ou fonction, contient simplement une série d'étapes à réaliser (**C, Fortran**, etc.)
Programmation orientée objet	Un modèle de langage de programmation qui s'articule autour d'objets et de données, plutôt que d'actions et de logique (**C++, JAVA, C#**, etc.).
Programmation orientée objet distribués	Système formé de composants matériels ou logiciels localisés sur des ordinateurs en réseau qui communiquent et coordonnent leurs actions uniquement par l'envoi de messages, le tout en vue de répondre à un besoin donné (**RMI, CORBA**, etc.).
Programmation orientée composants	Un paradigme de programmation qui vise le développement de logiciel par agrégation de briques logicielles existantes. Il permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance (**EJB**).
Programmation orientée services	Une forme d'architecture de médiation qui est un modèle d'interaction applicative qui met en œuvre des services.
Programmation orientée agents	Un paradigme de programmation dans lequel, les agents sont les éléments centraux du langage, de la même façon que les objets sont centraux pour les langages orientés objets. Ils sont caractérisés par des notions mentales comme leurs croyances, leurs décisions et leurs obligations.

2. Les agents

2.1. Définitions

Le concept « d'agent » a connu une grande évolution dans sa définition surtout qu'il s'applique, de plus en plus, dans des domaines différents. Et donc il n'existe pas une définition unifié pour le concept d'agent. Nous allons présenter par la suite quelques-unes proposées dans la littérature.

2.1.1. Définition 1

[27] ''L'agent est une entité physique ou virtuelle qui est capable d'agir et de percevoir son environnement, qui peut communiquer directement avec d'autres agents et qui possède des ressources propres et des compétences lui permettant de satisfaire ses objectives''.

2.1.2. Définition 2

[28] ''Un agent est un système informatique situé dans un environnement donné et capable d'agir de manière autonome dans cet environnement pour atteindre ses objectifs pour lesquels il a été conçu. Le système devrait pouvoir agir sans l'intervention directe des humains (ou d'autres agents), et devrait avoir le contrôle de ses propres actions et de son état interne''.

2.1.3. Définition 3 :

[29] ''Les Agents intelligents exécutent sans interruption trois fonctions : perception des conditions dynamiques dans l'environnement ; action pour affecter des conditions dans l'environnement, et le raisonnement pour interpréter des perceptions, résoudre des problèmes, mener des inférences et déterminer des actions''.

2.2. Caractéristiques d'un agent

En se basant sur quelques définitions du concept d'agent dans la littérature, on pourra extraire certaines caractéristiques communes de l'agent.

2.2.1. L'autonomie

C'est la caractéristique principale d'un agent, elle caractérise l'agent par sa capacité d'effectuer certaines tâches et de prendre des initiatives sans l'intervention d'un tiers que ce soit un humain ou un autre agent afin de satisfaire ses objectives.

2.2.2. La communication et la coopération

L'agent doit pouvoir communiquer avec les autres agents ou utilisateurs humaines afin d'échanger des informations lui permettant d'accomplir ses objectifs. Elle doit être capable ainsi d'établir des relations avec les autres agents pour effectuer des actions conduisant à un objectif en commun.

2.2.3. La rationalité ou l'intelligence

Un agent doit être capable, à partir de sa base de connaissances et des mécanismes de raisonnement, de réaliser des actions autonomes et flexibles pour atteindre ses objectifs et éventuellement d'apprendre et s'adapter dans le temps.

2.2.4. La mobilité

Un agent doit être capable de se déplacer d'une machine à une autre et doit être multi plateformes et multi-architecture.

2.3. Types d'agent

2.3.1. Agent réactif

Ces agents sont considérés comme non ou peu intelligents, ils n'ont pas une représentation symbolique de leur environnement, ni du monde auquel ils appartiennent. Ils réagissent par stimulus-réponse à l'état courant de l'environnement. L'intelligence émerge de leur association et communication.

2.3.2. Agent cognitif

Contrairement aux agents réactifs, les agents cognitifs sont dotés d'une forme d'intelligence, ils ont une représentation symbolique de leur environnement et une base de connaissances leurs permettant de prendre des décisions pour atteindre leurs buts.

Exemple [27]:

Supposons qu'un robot veuille franchir une porte et que celle-ci soit fermée à clef. Pour un agent **cognitif**, il pourra construire un plan "dans sa tête" tel que :

Plan ouvrirPorte

Aller jusqu'à l'endroit où se trouve la clef

Prendre la clef

Aller jusqu'`a la porte ouvrir la porte avec la clef.

Et pour un agent **réactif**, dispose du comportement suivant :

R1 : si je suis devant la porte et que j'ai une clef, alors l'ouvrir

R2 : si je suis devant la porte et sans clef, alors essayer de l'ouvrir

R3 : si la porte ne s'ouvre pas et que je n'ai pas la clef, alors aller chercher la clef

R4 : si je cherche une clef et qu'il y a une clef devant moi, alors prendre la clef et aller vers la porte.

3. Système Multi Agents

3.1. Définition

[30] définissent un système multi-agents (SMA) comme étant un système distribué, composé d'un ensemble d'agents interagissant entre eux selon des modes de coopération, de concurrence ou de coexistence.

Un SMA est composé généralement [27]:

- Un environnement Env, c'est-à-dire, un espace disposant généralement d'une métrique ;
- Un ensemble d'objets O, ces objets sont situés, c'est- à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans Env. les objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, crées, détruits et modifiés par les agents ;
- Un ensemble A d'agents, qui sont des objets particuliers, lesquels représentent les entités actives du système ;
- Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux ; − Un ensemble d'opérations Op. Permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O ;
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

3.2. Liens du SMA avec les autres disciplines

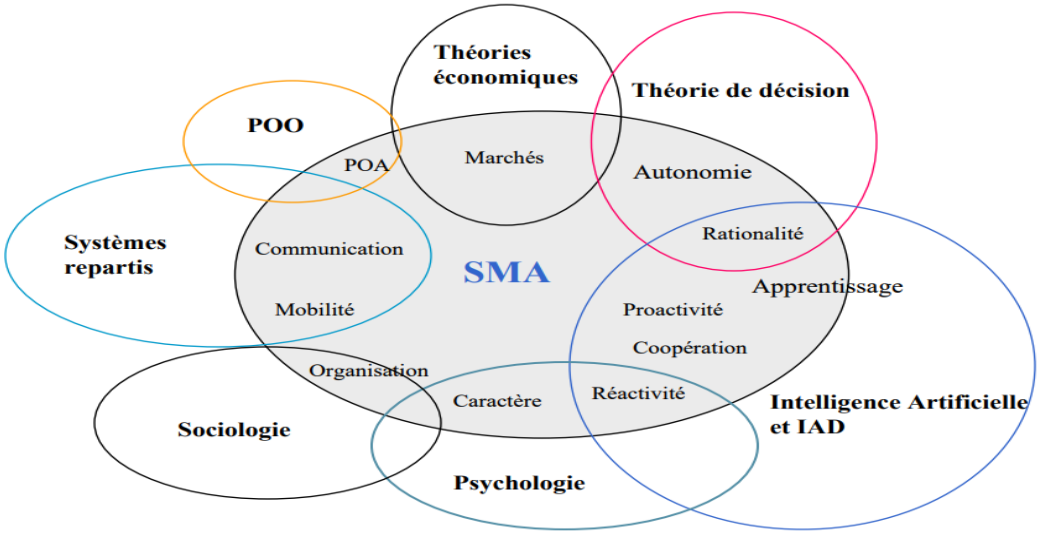


Figure 6: liens du SMA avec les autres disciplines [31]

3.3. Mécanismes d'interaction entre agents

Parmi les notions fondamentales des SMA, il y a l'interaction entre les agents, en effet l'environnement étant vaste et ouvert, et il n'est pas possible qu'un agent stocke toute la représentation du monde. Par contre, un agent peut se déplacer ou encore interagir avec d'autres agents qui sont dans son voisinage pour explorer d'autres agents qui sont dans son voisinage pour acquérir, partager et échanger des connaissances distribuées.

3.3.1. Coopération

La coopération consiste en l'interaction ou la participation des agents afin de réaliser conjointement une tâche ou d'atteindre un objectif commun. En effet chaque agent est caractérisé par : ses buts, ses capacités à réaliser certaines tâches et les ressources dont il dispose. L'interaction est motivée par le fait qu'un agent peut désirer des ressources que les autres possèdent. [30]

La coopération peut se faire à travers deux modèles [32]: coopération par partages de tâches ou coopération par partages de résultats. La coopération par partage de tâches consiste en une décomposition du problème principal en un ensemble de sous-problèmes moins complexes, puis en la répartition de ces sous-problèmes entre les différents agents. Cette allocation se fait généralement par des mécanismes d'appels d'offres tel que le Contract Net ou directement par un agent supérieur qui maîtrise les compétences de chacun des agents du système. Par ailleurs, la coopération par partage de résultats implique un partage des informations liées aux sous-problèmes. Ce partage d'information peut se faire d'une manière proactive (un agent qui envoie des informations à un autre en croyant que cela peut lui être utile) ou d'une manière réactive (un agent qui envoie des informations en réponse à une requête qui lui a été adressée).

3.3.2. Coordination

Dans un système coopératif, les agents possèdent un modèle de leur environnement et ce des autres, ils perçoivent les effets des actions des autres agents présents dans l'environnement pour mettre à jour les modèles des autres agents dont dispose chacun.

Selon [32] parmi les principales approches utilisées pour coordonner les activités d'agents, il y la coordination par « planification partielle globale » (PGP : Partial Global Planning) et la coordination par intentions communes. Dans la coordination par planification partielle globale, chaque agent génère des plans à court terme qui indiquent les actions futures à entreprendre pour atteindre ses objectifs. Ces plans sont partagés entre les agents dans le but d'identifier les améliorations potentielles à apporter à la coordination et ainsi d'éviter des actions inconsistantes ou contradictoires. En cas de nécessité, ces plans locaux peuvent être modifiés. Dans le cas de la coordination par intentions communes, les SMA sont considérés comme une équipe de travail dont les membres peuvent générer une intention commune d'exécuter une activité commune.

3.3.3. Négociation

La négociation est une solution pour surmonter les conflits entre les agents. C'est un processus qui permet de faire évoluer la satisfaction de chaque agent lors de la discussion d'une proposition dans le but d'aboutir à un arrangement mutuellement acceptable.

Un conflit se définit comme un désaccord entre au moins deux parties interdépendantes dont les intérêts paraissent incompatibles concernant la répartition d'un ensemble de ressources multilatérales nécessaires pour l'exécution d'un processus de gestion [33].

3.3.4. Communication

Les agents peuvent interagir soit en accomplissant des actions linguistiques (en communiquant entre eux), soit en accomplissant des actions non-linguistiques qui modifient leur environnement. En communiquant, les agents peuvent échanger des informations et coordonner leurs activités. Dans les SMA deux stratégies principales ont été utilisées pour supporter la communication entre agents : les agents peuvent échanger des messages directement ou ils peuvent accéder à une base de données partagées (appelée tableau noir ou "blackboard") dans laquelle les informations sont postées. Les communications sont à la base des interactions et de l'organisation sociale d'un SMA. [30]

3.4. Les modes de communication entre les agents

Nous distinguons deux modes de communication :

- Par envoi de messages : Chaque agent a un point de vue local du problème et communique directement avec les autres agents qu'il connaît. Ce type de communication peut se

résumer à des échanges asynchrones. Ainsi, ce mode de communication se base sur deux principales variantes à savoir une communication point à point (c'est au cas où on connait le destinataire) et une communication par diffusion (c'est au cas où on ne connait pas le destinataire).[29]

- Par mémoire partagée (communication indirecte) : sont des systèmes à base de tableaux noirs (blackboard), Le modèle de tableau noir est fondé sur un découpage en modules indépendants qui ne communiquent directement aucune information, mais qui interagissent indirectement en partageant des informations. Ces modules, appelés sources de connaissance ou KS (pour Knowledge Sources), travaillent sur un espace qui comprend tous les éléments nécessaires à la résolution d'un problème. [27]

3.5. Architecture SMA

3.5.1. Architecture BDI

L'architecture BDI est une approche utilisé dans la conception des agents délibératifs, elle part du modèle « Croyance-Désir-Intention » sur lequel les agents se basent pour choisir leurs actions.

- Croyance : correspond aux informations qu'un agent possède sur l'environnement, ces informations peuvent être changé au fur et à mesure à cause de l'interaction de l'agent avec d'autres agents ce qui implique le recueil de plus d'informations
- Désir : représente les buts que l'agent aimerait voir réalisés, c'est la description du comportement désiré de l'agent en termes de satisfaction.

Intention : sont actions que l'agent a décidé de faire afin d'accomplir ses désirs.

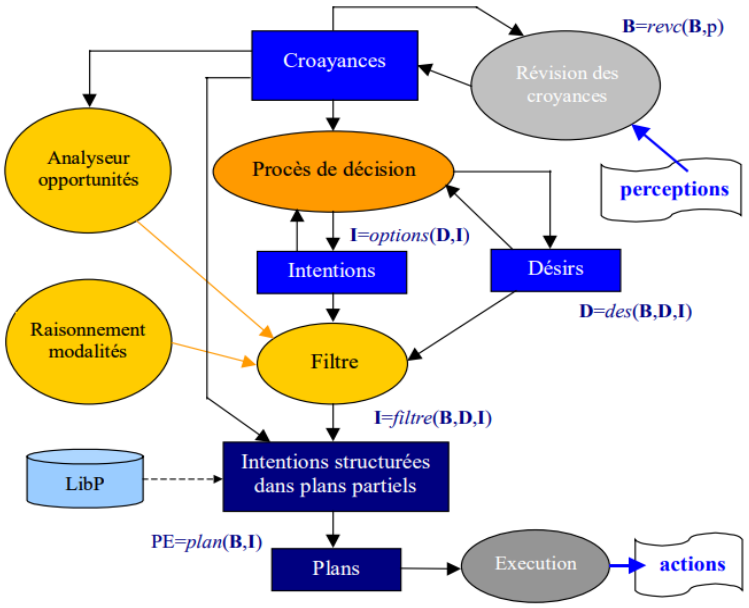


Figure 7: Architecture BDI [34]

3.5.2. Architecture réactive

Les architectures réactives représentent le fonctionnement de l'agent au moyen de composantes avec une structure de contrôle simple, et sans représentation évoluée des connaissances de l'agent. L'intelligence de l'agent est vue comme étant le résultat des interactions entre ces composantes et l'environnement. Cela veut dire qu'une telle architecture peut résoudre des problèmes complexes, qui normalement demandent un comportement intelligent, sans traiter l'intelligence du point de vue classique de l'intelligence artificielle [34].

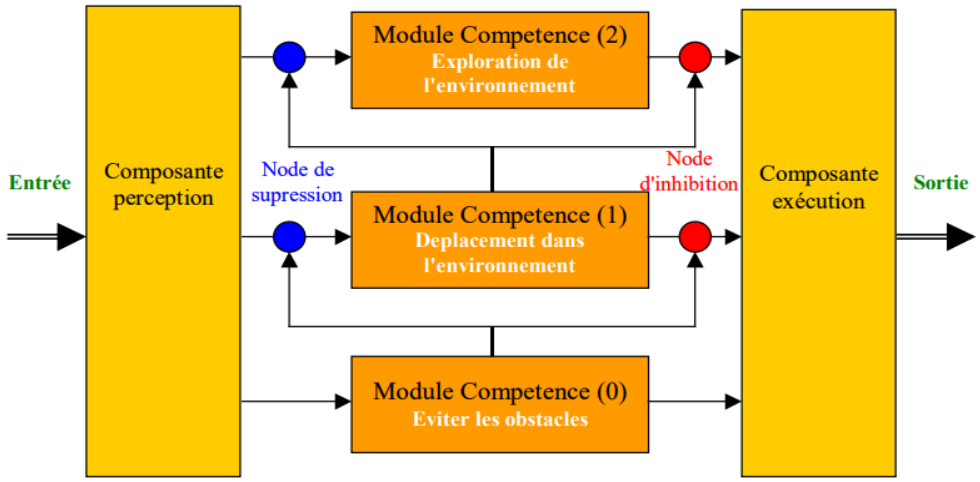


Figure 8: Architecture réactive [34]

3.5.3. Architecture hybride

[29] Cette architecture combine à la fois des aspects réactifs et délibératifs, elle contient généralement plusieurs couches logicielles. Les couches peuvent être arrangées verticalement dont seulement une couche a accès aux capteurs et aux effecteurs, ou horizontalement dont toutes les couches ont accès aux entrées et aux sorties.

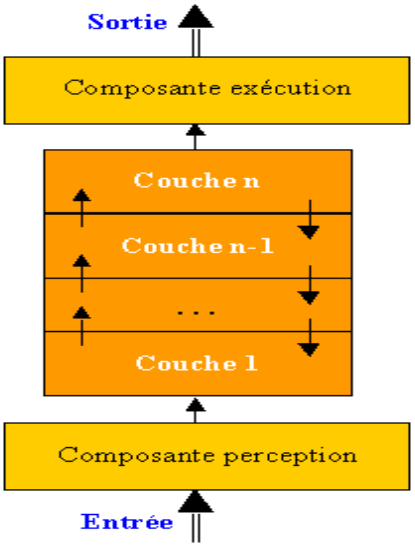


Figure 9: Architecture hybride [34]

3.6. Domaines d'application des SMA

Le concept d'agent a été l'objet d'études dans plusieurs disciplines, c'est pour cela que les domaines d'application des SMA sont vastes, nous allons citer par la suite les grandes catégories d'application des SMA selon [27].

3.6.1. La résolution de problèmes

La résolution de problèmes au sens large concerne en fait toutes les situations dans lesquelles des agents logiciels accomplissent des tâches utiles aux êtres humains. Cette catégorie s'oppose aux applications de robotique distribuée en ce sens que les agents sont purement informatiques et n'ont pas de structure physique réelle.

3.6.2. La robotique distribuée :

La robotique distribuée porte sur la réalisation non pas d'un seul robot, mais d'un ensemble de robots qui coopèrent pour accomplir une mission. Elle utilise des agents concrets qui se déplacent dans un environnement réel.

3.6.3. La simulation multi-agents :

La simulation multi-agents est fondée sur l'idée qu'il est possible de représenter sous forme informatique le comportement des entités qui agissent dans le monde et qu'il est ainsi possible de représenter un phénomène comme le fruit des interactions d'un ensemble d'agents disposant de leur propre autonomie opératoire. L'intérêt des simulations est de pouvoir considérer aussi bien des paramètres quantitatifs que qualitatifs.

3.7. Plateformes SMA

Il existe plusieurs plateformes permettant la simulation, le développement et l'exécution des systèmes multi-agents. Nous allons citer quelques-uns.

3.7.1. AgentBuilder

''AgentBuilder est un environnement de développement complet. Une modélisation orientée-objet avec OMT constitue la base de la conception des systèmes à laquelle on ajoute une partie « ontologie ». L'élaboration du comportement des agents se fait à partir du modèle BDI et du langage AGENT-0. KQML est utilisé comme langage de communication entre les agents. L'exécution du système se fait à partir de l'engin d'exécution d'AgentBuilder. Par contre, on peut créer des fichier « .class » et les exécuter sur une JVM standard. AgentBuilder est un outil complexe qui demande des efforts d'apprentissage importants et de bonnes connaissances dans le domaine des systèmes multi-agents pour être utilisé de façon performante. Il est limité au niveau de l'extensibilité, du déploiement et de la réutilisabilité.''[35]

3.7.2. Madkit

''MadKit est une plateforme de développement de systèmes multi-agents destinée au développement et à l'exécution de systèmes multi-agents et plus particulièrement à des systèmes multi-agents fondés sur des critères organisationnels groupes et rôles.

MadKit est écrit en Java et fonctionne en mode distribué de manière transparente à partir d'une architecture "peer to peer" sans nécessiter de serveur dédié. Il est ainsi possible de faire communiquer des agents à distance sans avoir à se préoccuper des problèmes de communication qui sont gérés par la plateforme.'' [36]

3.7.3. Zeus

''Zeus est un environnement complet qui utilise une méthodologie appelée « role modeling » pour le développement de systèmes collaboratifs. Les agents possèdent trois couches. La première couche est celle de la définition où l'agent est vu comme une entité autonome capable de raisonner en termes de ses croyances, ses ressources et de ses préférences. La seconde couche est celle de l'organisation. Dans celle-ci, il faut déterminer les relations entre les agents. La dernière couche est celle de la coordination. Dans celle-ci, on décide des modes de communication entre les agents, protocoles, coordination et autres mécanismes d'interactions. L'outil est un des plus complets. Les différentes étapes du développement se font à l'intérieur de plusieurs éditeurs : ontologie, description des tâches, organisation, définition des agents, coordination, faits et variables ainsi que les contraintes. Le développement de SMA avec Zeus est cependant conditionnel à l'utilisation de l'approche « role modeling ». L'outil est assez complexe et sa maîtrise nécessite beaucoup de temps.''[35]

3.7.4. JACK

JACK est un langage de programmation et un environnement de développement pour agents cognitifs, développé par la société Agent OrientedSoftware comme une extension orientée agent du langage Java.

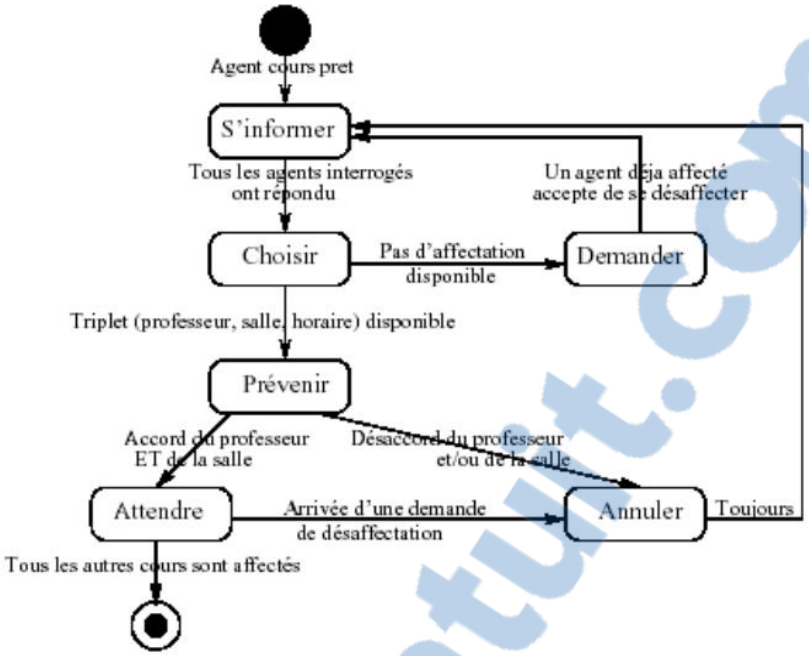
3.7.5. JADE

''JADE (Java Agent Development Framework) est un environnement logiciel permettant de créer des systèmes d'agent pour la gestion des ressources d'informations dans le réseau conformément aux spécifications FIPA pour les systèmes inter-agents inter-agents. JADE fournit un middleware pour le développement et l'exécution d'applications basées sur des agents qui peuvent fonctionner et interagir de manière transparente aussi bien dans un environnement filaire que sans fil. En outre, JADE prend en charge le développement de systèmes multi-agents via le modèle d'agent programmable et extensible prédéfini et un ensemble d'outils de gestion et de test. Actuellement, JADE est l'un des cadres de développement d'agents les plus utilisés et prometteurs'' [37].

4. Systèmes multi-agents et le problème de gestion d'emploi du temps

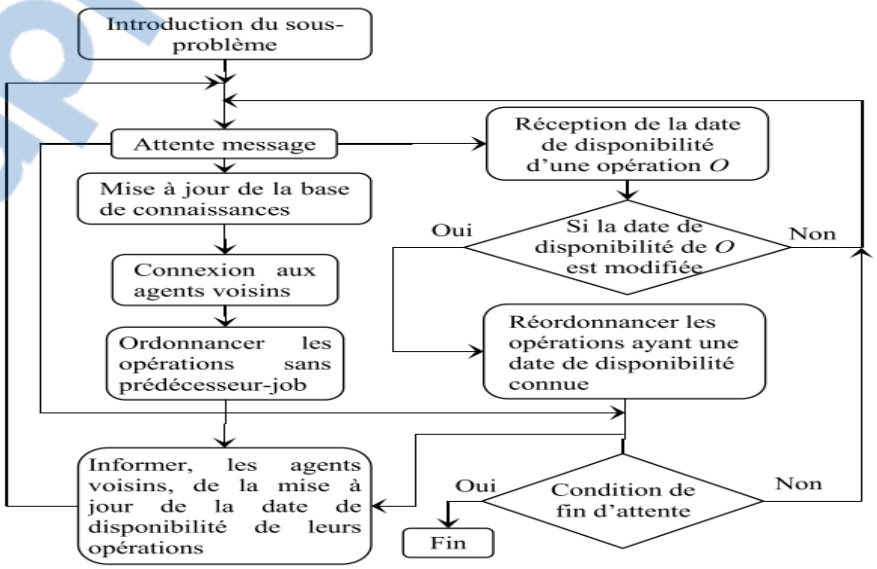
Dans la littérature, plusieurs approches ont été proposé pour résoudre le problème d'emploi du temps en général. Nous allons présenter par la suite quelques travaux qui utilisent les SMA pour les problèmes de planification et d'ordonnancement.

[38] Dans leur travail, ils ont abordé le problème de génération distribuée de l'emploi du temps des cours d'un établissement scolaire, cela a été résolu d'un point de vue multi-agents afin de donner un cadre d'étude à la coordination distribuée soutenue par la communication. Le problème de départ a été formalisé sous la forme d'un problème de satisfaction de contraintes distribué DCSP (Distributed Constraints Satisfaction Problem), puis un algorithme à retour arrière a été proposé pour le résoudre et enfin cet algorithme a été implémenté et testé dans un système multi-agent.



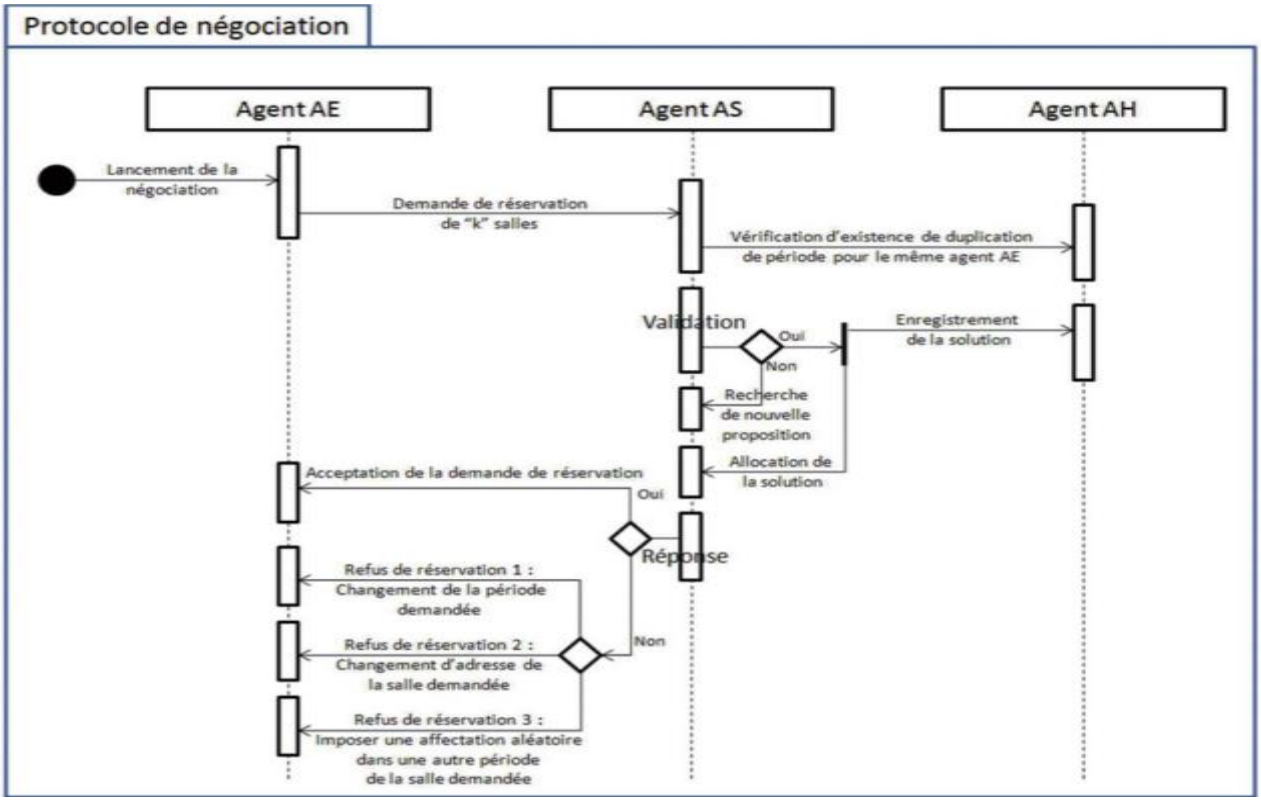
[39] Ils s'intéressent à un problème d'ordonnancement de type job shop où les travaux sont supposés définis au fur et à mesure que les commandes apparaissent.

Contrairement aux approches classiques centralisées de résolution, une approche distribuée basée sur les systèmes multi-agents est proposée, dans laquelle, des agents sont associés aux différentes machines constituant l'atelier, et possédant chacun leur propre autonomie décisionnelle. Chaque agent est responsable de la résolution du sous problème d'ordonnancement relatif à sa machine. Les différents agents du système coopèrent et s'échangent des messages pour produire, au niveau local un ordonnancement réalisable, et ce dans le but de converger au mieux vers une solution globale qui minimise le makespan. En considérant à la fois deux cas de problèmes d'ordonnancement, statique et dynamique.

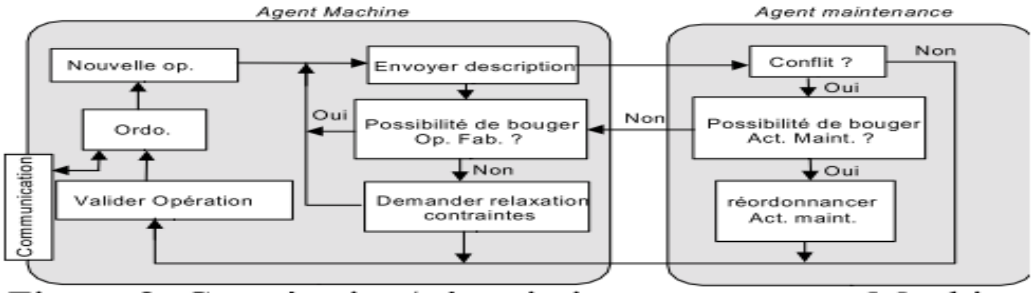


[29] Ils ont abordé le problème d'emploi du temps des cours dans les établissements universitaires. L'approche proposée consiste en l'utilisation des SMA pour un traitement distribué et fortement parallèle du problème et l'intégration d'un comportement coopératif aux agents du système afin d'améliorer l'efficacité des résultats, en satisfaisant plus de préférences et de contraintes qui semble être plus difficile du problème.

Le modèle est nommé M.A.T.P (Multi-Agent model for Timetabling Problem).



[40] Proposent de modéliser la fonction d'ordonnancement de la production et celle d'ordonnancement de la maintenance par deux systèmes multi-agents coopérants. Ils ont mis en place une nouvelle version de RAMSES II permettant de traiter le cas d'activités de maintenance conditionnelle, c'est-à-dire le cas où c'est la séquence d'opérations de production planifiée sur une machine qui induit l'activité de maintenance. Elle permet aussi de tenir compte des ressources de maintenance, par la description des ressources matérielles et des compétences d'opérateurs associées à une activité de maintenance, et par la prise en compte de leur capacité.



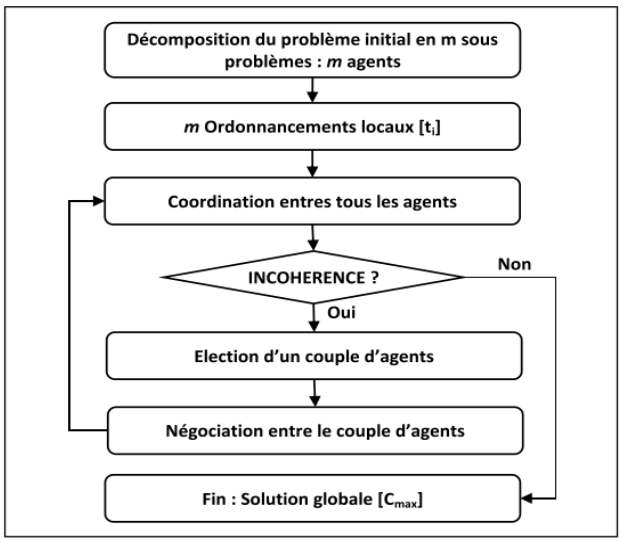
[41] Ils ont proposé une nouvelle approche proactive-réactive basée agents pour l'ordonnancement conjoint des tâches de production et de maintenance

L'approche proposée prend en compte les incertitudes des données du problème. Ces incertitudes peuvent être des imprécisions relatives aux estimations des durées opératoires des tâches ou bien des incertitudes concernant l'apparition ou non d'événements pouvant affecter le déroulement du processus de production.

Deux phases ont été proposées : (1) Une phase proactive où un ordonnancement proactif est calculé. (2) Une phase réactive où ce dernier est adapté aux changements pouvant surgir dans l'atelier suite à l'apparition d'événements perturbateurs.

[42] L'article s'intéresse à la résolution distribuée d'un problème d'ordonnancement de type job shop, les auteurs ont proposé une approche coopérative qui fournit une famille de solutions flexibles, au lieu d'une solution unique, afin de tenir compte des perturbations. La modélisation distribuée proposée au problème d'ordonnancement job shop considère que

chaque machine possède une autonomie décisionnelle lui permettant de gérer son propre ordonnancement local, et qu'elle coopère avec les autres machines pour aboutir à une solution globale réalisable. L'utilisation des SMA est principalement motivée par les moyens de communication entre agents (les machines dans ce cas) qu'ils offrent à travers les standards tels que les langages FIPA-ACL ou bien KQML ainsi que la multitude d'outils existants facilitant la mise en œuvre d'une solution distribuée.



Conclusion

Au cours de ce chapitre, nous avons défini le concept d'agent et les systèmes multi agents, nous avons vu quelques caractéristiques des agents et leurs types, puis les mécanismes d'interactions et les modes de communication entre eux. Par la suite, nous avons parlé des architectures des SMA et quelques domaines d'application de ceux-ci. Finalement, nous avons présenté quelques travaux de la littérature résolvant le problème d'emploi du temps à l'aide des SMA.

Chapitre 3 : Modélisation et description du système

Introduction

Nous avons parlé dans les chapitres précédents du contexte du projet, à savoir une description du bloc opératoire et le processus opératoire, et nous avons montré comment sa gestion pourrait guider à une augmentation des soins des patients et aussi une meilleure gestion du budget du bloc. Puis dans le chapitre qui suit, nous avons parlé des SMA comme étant une évolution des approches classiques de l'I.A. et une sorte de l'I.A distribuée. Nous avons vu comment ces systèmes ont connu un grand intérêt dans différents domaines de recherche.

Dans ce chapitre, nous allons commencer par la reformulation de notre problème en un modèle mathématique, puis, nous allons décrire les deux phases qui constituent notre système.

1. Modélisation du problème

1.1. Introduction

Nous présentons dans cette partie le modèle mathématique que nous avons proposé pour la résolution du problème de planification et d'ordonnancement des salles d'opérations.

Comme c'est indiqué dans l'introduction générale nous visons à minimiser la sous-utilisation des salles d'opérations ainsi que la sur-utilisation de celles-ci, en prenant en compte la durée maximale de travail de chaque chirurgien par jour, le nombre d'heures maximum d'ouverture des salles d'opérations, les spécialités des salles d'opérations et les spécialités des chirurgiens.

Nous commençons par la description des différents paramètres du problème et les variables de décisions.

Paramètres :

N_{HR}^{j} : Nombre des heures régulières du jour *j*.

N_{HS}^{j}: Nombre des heures supplémentaires du jour *j*.

N_O : Nombre des opérations.

N_{CH} : Nombre des chirurgiens.

N_S : Nombre des salles d'opérations dans le bloc opératoire.

NO : Nombre des jours ouvrables.

D_o : Durée de l'opération *o*.

Max_c^j : Maximum des heures de travail pour le chirurgien *c* le jour *j*.

P : Nombre de périodes par jour.

ω : Ratio coût d'une heure normale de travail à une heure supplémentaire.

SP : Nombre des spécialités chirurgicales.

$SS_{s,sp}$: Spécialité de la salle *s*.

$SC_{c,sp}$: Spécialité du chirurgien *c*.

$SO_{o,sp}$: Spécialité de l'opération *o*.

Variables de décision:

$$x_{o,s,j,p} = \begin{cases} 1 \ si \ l'opération \ o \ est \ affectée \ à \ la \ salle \ s \ le \ jour \ j \ pendant \ la \ période \ p \\ 0 \ sinon \end{cases}$$

$$l_{o,c} = \begin{cases} 1 \ si \ l'opération \ o \ est \ affecté \ au \ chirurgien \ c \\ 0 \ sinon \end{cases}$$

$$e_{s,j} = \begin{cases} 1 \ si \ la \ salle \ s \ est \ utilisé \ le \ jour \ j \\ 0 \end{cases}$$

1.2. Modèle mathématique

Fonction Objectif :

$$Minimiser \ f = \max\left\{\sum_{j=1}^{NO}\sum_{s=1}^{N_s}\left[e_{s,j} * N_{HR} - \sum_{o=1}^{N_O}\sum_{p=1}^{P} x_{o,s,j,p} * D_o\right], \omega\left[\sum_{j=1}^{NO}\sum_{s=1}^{N_s}\left(\left(\sum_{o=1}^{N_O}\sum_{p=1}^{P} x_{o,s,j,p} * D_o\right) - e_{s,j} * N_{HR}\right)\right]\right\} \quad (1)$$

Contraintes :

$$\sum_{o=1}^{N_o} x_{o,s,j,p} \leq 1 \ ; \ \forall s = 1 \dots N_s \ , \ \forall j = 1 \dots NO \ , \ \forall p = 1 \dots P \quad (2)$$

$$\sum_{j=1}^{NO}\sum_{p=1}^{P}\sum_{s=1}^{N_s} x_{o,s,j,p} = 1 \ ; \ \forall \ o = 1 \dots N_O \quad (3)$$

$$\sum_{s=1}^{N_s}\sum_{o=1}^{N_o^j} l_{o,c} * x_{o,s,j,p} \leq 1 \ ; \ \forall p = 1 \dots \ P \ , \forall j = 1 \dots NO \ , \ \forall c = 1 \dots N_{CH} \quad (4)$$

$$\sum_{o=1}^{N_o}\sum_{s=1}^{N_s}\sum_{p=1}^{P} x_{o,s,j,p} * l_{o,c} * D_o \leq Max_c^j \ ; \ \ \forall \ j = 1 \dots NO \ , \forall c = 1 \dots N_{CH} \quad (5)$$

$$\sum_{o=1}^{N_o} \sum_{p=1}^{P} x_{o,s,j,p} * D_o \leq N_{HR}^{j} + N_{HS}^{j} \; ; \; \forall j = 1 \dots NO \, , \forall s = 1 \dots N_s \quad (6)$$

$$\sum_{c=1}^{N_{CH}} \sum_{sp=1}^{SP} l_{o,c} * SC_{c,sp} * SO_{o,sp} \geq 1; \; \forall o = 1 \dots N_O \quad (7)$$

$$(x_{o,s,j,p} = 1) => (e_{s,j} = 1); \; \forall p = 1 \dots P, \; \forall o = 1 \dots N_O \, , \forall s = 1 \dots N_S \, , \forall j = 1 \dots NO \quad (8)$$

$$\sum_{j=1}^{NO} \sum_{p=1}^{P} \sum_{s=1}^{N_S} \sum_{sp=1}^{SP} SO_{o,sp} * SS_{s,sp} * x_{o,s,j,p} = 1 \; ; \; \forall o = 1 \dots N_O \quad (9)$$

(2) permet de s'assurer qu'une salle est utilisée au plus une fois pendant une période du jour.

(3) S'assurer qu'une opération est traitée une et une seule fois.

(4) S'assurer qu'un chirurgien n'effectue pas plus d'une opération dans la même période du jour.

(5) La durée totale des opérations affectées à un chirurgien pendant un jour ne doit pas dépasser son nombre maximum des heures de travail ce jour-là.

(6) La durée totale des opérations affectées à une salle pendant un jour ne doit pas dépasser sa capacité régulière plus son temps supplémentaire ce jour-là.

(7) S'assurer qu'une opération est affectée à au moins un chirurgien de même spécialité.

(8) Une opération est affectée à une salle pendant un jour implique que cette salle est utilisée ce jour-là.

(9) S'assurer qu'une opération de spécialité *sp* est affectée à une salle de même spécialité.

2. Description du système

Le système est composé de deux phases :

- Phase proactive : c'est la phase qui permet de construire un planning des opérations élective qui auront lieu au cours de la semaine, ce planning doit satisfaire toutes les contraintes fortes et essaye de satisfaire au maximum les contraintes faibles en minimisant la fonction objectif décrit ci-dessus.
- Phase réactive : cette phase a pour objectif la prise en compte des aléas du quotidien ; des évènements qui peuvent survenir aléatoirement et qui peuvent perturber le déroulement des opérations dans le bloc opératoire. Vu que le bloc opératoire est un environnement incertain (l'arrivée des cas urgents, panne des matériels, etc.).

La figure ci-dessous représente l'architecture du système :

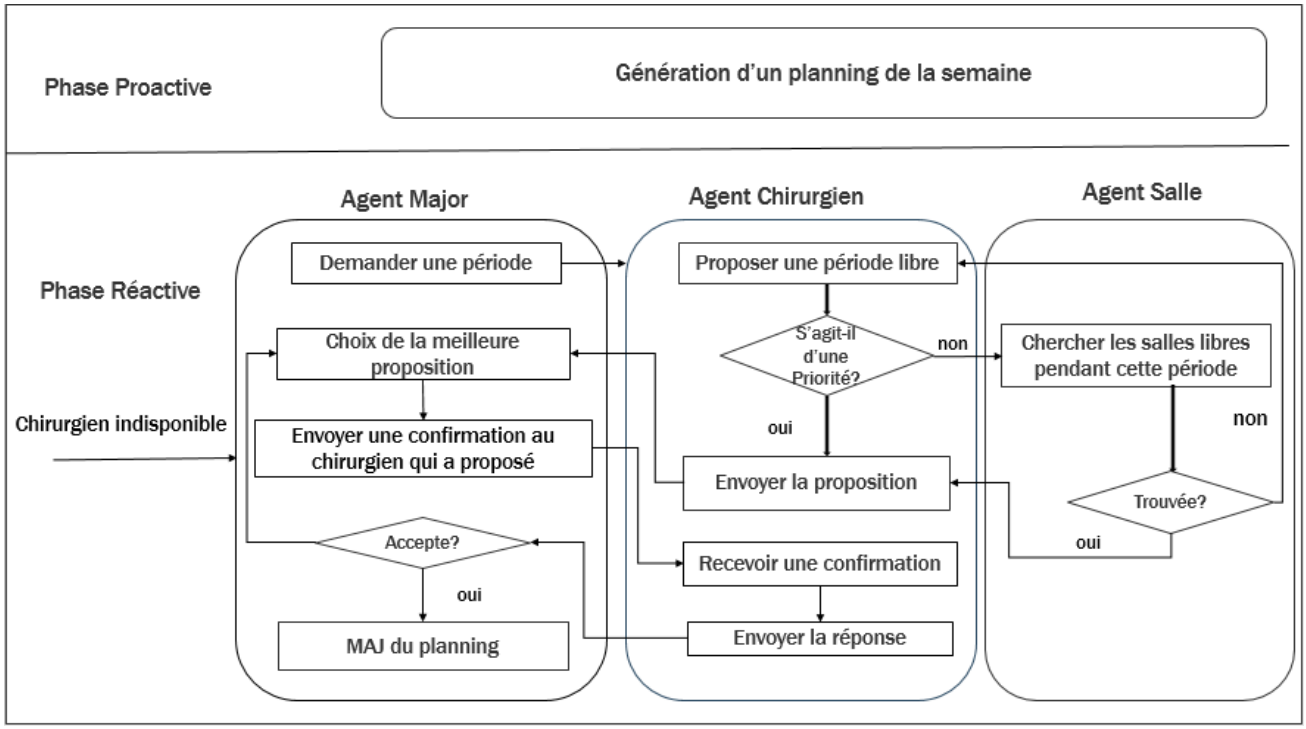


Figure 10: l'architecture global du système

2.1. La phase proactive

Dans cette phase, nous voudrions construire le planning de la semaine qui prend en charge les contraintes citées en-dessous et qui minimise la fonction objectif. Pour cela, nous nous servirons d'une méta-heuristique qui est largement utilisé et qui a prouvé son efficacité dans ces problèmes d'ordonnancement, c'est la méthode de recherche Tabou.

La recherche Tabou comme ci-indiqué dans 1.2.2.2.1 est une méthode basée sur le principe de voisinage, elle permet à partir d'une solution initiale de parcourir la liste des voisinages et choisir un voisin qui minimise la fonction objectif (ou maximiser).

Cette méta-heuristique se distingue des autres méthodes de recherche locale par sa liste taboue qui mémorise les voisins déjà visités afin d'éviter le problème des optima locaux.

Nous allons par la suite décrire les différents composants de l'algorithme RT.

2.1.1. Description des solutions

La solution consiste en une matrice à trois dimensions dont les lignes représentent les salles d'opérations, les colonnes représentent les jours ouvrables et chaque jour contient la liste des périodes. Et un tableau qui représente l'affectation des chirurgiens aux opérations.

	Jour 1					Jour 2					…		Jour 5				
	P=1	P=2	P=3	…	P=12	1	2	3	…	12	1	…	1	2	3	…	12
Salle 1																	
Salle 2																	
…																	

Tableau 2: tableau illustratif de la forme de la solution qui va être généré

Chaque élément de la matrice renvoie l'indice de l'opération si cette dernière est affectée à une période du jour et une salle, sinon la matrice renvoie 0.

Le tableau des affectations des chirurgiens a la même longueur de celle des opérations, pour chaque opération il renvoie l'indice du chirurgien qui lui est affecté.

	1	2	3	…	NbOpérations
Opérations					

Dans ce projet, nous avons défini quatre solutions :

2.1.1.1. **Solution initiale**

Une méthode à base de voisinage doit partir obligatoirement d'une solution initiale qui lui permet d'explorer les voisins.
Dans ce travail, nous nous sommes partis de la méthode *Best Fit Descending* pour la génération de la première solution. Cette méthode appartient à la famille des sacs à dos. Elle consiste à trier les opérations selon l'ordre décroissant et les affecter en premier, cela privilégiera la réalisation des opérations d'une durée longue au début du jour d'où la disponibilité des ressources matérielles et humaines.

2.1.1.2. **Solution courante**

Cette solution est utilisée lors de l'exploration des voisins afin de garder temporairement la nouvelle solution obtenue à partir du voisin courant.

2.1.1.3. **Solution**

Elle consiste à garder la solution courante qui minimise le coût total entre les quatre voisins, sans aucune évaluation des contraintes et ce, afin de poursuivre la recherche aux alentours de cette bonne solution [14] même s'elle est irréalisable.

2.1.1.4. Meilleure solution

Elle consiste à garder la solution qui minimise le coût total et qui satisfait l'ensemble des contraintes.

2.1.2. Mouvements

Ce sont des petits changements que subit une solution. Selon [] distinguent trois principaux mouvements :

- Inversion : consiste à permuter l'objet courant par l'objet qui est juste à côté, il est considéré comme le plus mauvais vue la taille réduite du voisinage qu'il engendre.
- Transposition : consiste à permuter l'objet courant par un autre quelconque, il est plus efficace que le déplacement pour le problème d'affectation quadratique car il ne change que la position des objets transposés.
- Déplacement : consiste à changer la place de l'objet courant. Il est considéré comme le meilleur pour les problèmes d'ordonnancement car c'est le séquencement qui est important pas la position individuelle des tâches.

Dans ce travail, nous avons choisi le déplacement car dans un part, il est efficace pour le problème d'ordonnancement et d'un autre part, il va nous permettre d'accélérer la recherche dans des régions plus loin de la solution courante et qui ne sont pas encore visitées.

2.1.3. Voisinage

Le voisinage consiste à appliquer des changements élémentaires sur la solution courante afin de trouver une solution optimale et qui satisfait toutes les contraintes dures.

Dans ce travail, nous avons défini quatre voisinages :

- Voisin période : consiste à déplacer l'opération courante vers une autre période qui n'est pas encore affectée.
- Voisin salle : consiste à déplacer l'opération vers une autre salle de même spécialité.
- Voisin jour : permet de déplacer l'opération vers un autre jour.
- Voisin chirurgien : consiste à changer le chirurgien de l'opération par un autre de même spécialité.

2.1.4. Liste Taboue

La liste tabou est un élément indispensable dans l'algorithme de recherche Tabou, il permet de mémoriser les déplacements effectués et les solutions visités afin d'interdire de revenir à la même solution plus qu'une fois et tomber dans le problème des optima locaux. Dans ce projet, l'objet Tabou ne garde que l'état courant de l'élément et son état prochain c.-à-d., il mémorise la période courante de l'opération et la période obtenue après le déplacement, même chose pour les salles, jours et chirurgiens.

Une description technique va être présentée dans la partie de conception suivante, pour montrer les différents objets et les relations entre eux.

2.1.5. Algorithme de construction du planning

```
DÉBUT
   solution <- InitialisationSolution()
  affectationChirurgien<- AffectationChirurgien()
   meilleurCout <- CalculCout(solution)
   meilleureSolution <- solution
   listeTabou <- null
   TANTQUE  !ConditionArret  FAIRE
                    POUR CHAQUE operation FAIRE
                        \\ Recherche des voisinages
                        VoisinagePeriode <- ChoisirVoisinagePeriode()
                        VoisinageSalle <- ChoisirVoisinageSalle()
                        VoisinageJour <- ChoisirVoisinageJour()
                  \\ S'assurer que le voisin période n'est pas tabou
               SI Verifier(voisinPeriode) != null ALORS
                        solutionCourante <- Deplacer(solution, voisinPeriode)
                        SI CalculCout(solutionCourante) < CalculCout(solution) ALORS
                             solution <- solutionCourante
                             Ajouter(solution)
                        FINSI
               FINSI
               \\ S'assurer que le voisin jour n'est pas tabou
               SI Verifier(voisinJour) != null ALORS
                        solutionCourante <- Deplacer(solution, voisinJour)
                        SI CalculCout(solutionCourante) < CalculCout(solution) ALORS
                             solution <- solutionCourante
                             Ajouter(solution)
                        FINSI
               FINSI
              \\ S'assurer que le voisin salle n'est pas tabou
               SI Verifier(voisinSalle) != null ALORS
                        solutionCourante <- Deplacer(solution, voisinSalle)
                        SI CalculCout(solutionCourante) < CalculCout(solution) ALORS
                             solution <- solutionCourante
                             Ajouter(solution)
                        FINSI
               FINSI

               SI CalculCout(solution)< meilleureCout ALORS
                        POUR CHAQUE operation FAIRE
                        Conflits <- ExtraireConflits()
                       Si conflits DANS contrainte2 OU Contrainte3 OU Contrainte7
                                VoisinChirurgien <- ChoisirVoisinageChirurgien()
```

```
                    AffectationChirurgien<- DeplacerChirurgien(VoisinChirurgien)
                FINSI
                FINPOUR
                SI aucun conflit Faire
                meilleureSolution <- solution
                meilleureAffectation<-AffectationChirurgien
                meilleurCout <- calculCout(solution)
            FINSI
        FINSI

    FINTANTQUE
FIN
```

Les principales méthodes de l'algorithme :

- *InitialisationSolution()* : retourne une solution initiale, la construction de cette solution se fait selon quatre étapes :
    - Premièrement les opérations sont stockées selon l'ordre décroissant de leurs durées dans une liste, puis l'opération de la plus longue durée sera affectée au jour qui a la plus grande durée totale de travail des chirurgiens de même spécialité, et n'importe quelle salle de même spécialité, cette technique nous a permis d'avoir des bons résultats après plusieurs tests.
    - Deuxième étape consiste à affecter les opérations qui restent, à la salle qui a suffisamment de temps pour la réalisation de l'opération et a le moins de temps régulier restant, principe de « *Best Fit Descending* ». Donc ces opérations seront affectées au période réguliers.
    - Troisième étape : pour les opérations qui restent, elles seront affectées à la salle qui a la plus grande durée totale (temps régulier restant + temps supplémentaire). Cela permet de réaliser les opérations ayant la plus grande durée régulière restante et le minimum de temps supplémentaire possible principe de « *Worst Fit Descending* ».
    - La dernière étape : s'il reste encore des opérations non affectées, elles seront affectées à n'importe quelle salle de même spécialité et qui a une période libre sans prendre en compte les contraintes de capacités.

  L'idée de violer les contraintes douces en premier et puis les contraintes fortes s'il restent encore des tâches non affectées afin de générer une solution initiale, provient de [].

- *AffectationChirurgien()* : l'affectation des chirurgiens se fait en deux étapes :
    - La première étape : consiste à affecter les opérations du jour aux chirurgiens de même spécialité et qui ont la plus grande durée de travail restante ce jour –là.
    - Deuxième étape : consiste à affecter les opérations restantes aux chirurgiens de même spécialité sans tenir compte des contraintes de capacités ni de leur affectation à plus d'une opération à la même période du jour.

- *ChoisirVoisinagePeriode()/Salle/Jour/Chirurgien :* permet de chercher une période du même jour qui n'est pas encore affectée et retourne un objet Tabou décrivant le déplacement qui sera effectué en changeant la periode2, le même principe s'applique aux autres voisins.
- *Verifier()* : prend en paramètres le voisin (période, jour ou salle) et retourne 1 si cet objet est dans la liste Tabou.
- *Deplacer() :* prend en paramètres la solution courante et le voisin, et retourne la nouvelle solution après le déplacement.
- *Ajouter()* : permet d'ajouter un objet de type *Tabou* à *listeTabou.*
- *ExtraireConflits() :* cette méthode permet de retourner une matrice des conflits trouvés dans la solution. Les lignes correspondent aux opérations et les colonnes correspondent aux contraintes (1,2,3,4,5,6 et 7).
- *DeplacerChirurgien() :* retourne le nouveau tableau des *AffectationChirurgiens* en effectuant le déplacement par *voisinChirurgien.*

2.2. La phase réactive

Cette phase a pour objectif la prise en charge des imprévus quotidien, parmi lesquels nous citons l'indisponibilité des chirurgiens.

Un chirurgien peut déclarer son indisponibilité qui est parfois à cause d'une sous-estimation de la durée des opérations. Dans ce cas –là l'opération soit elle va être décalé vers une période qui est libre, ou bien le chirurgien cherche un autre chirurgien qui va lui remplacer.

La phase réactive part de cette problématique, en permettant au chirurgien d'être remplacé en premier temps, s'il n'y a plus de remplaçant, l'opération va être décalé vers la période libre la plus proche.

Notre système multi agents est composé :

- D'un agent Major : Un seul agent dans le système
- Des agents chirurgiens : il y a autant d'agents chirurgiens qu'il y a des chirurgiens dans l'hôpital.
- Des agents salles : il y a autant d'agents salles qu'il y a de salles d'opérations dans le bloc opératoire.

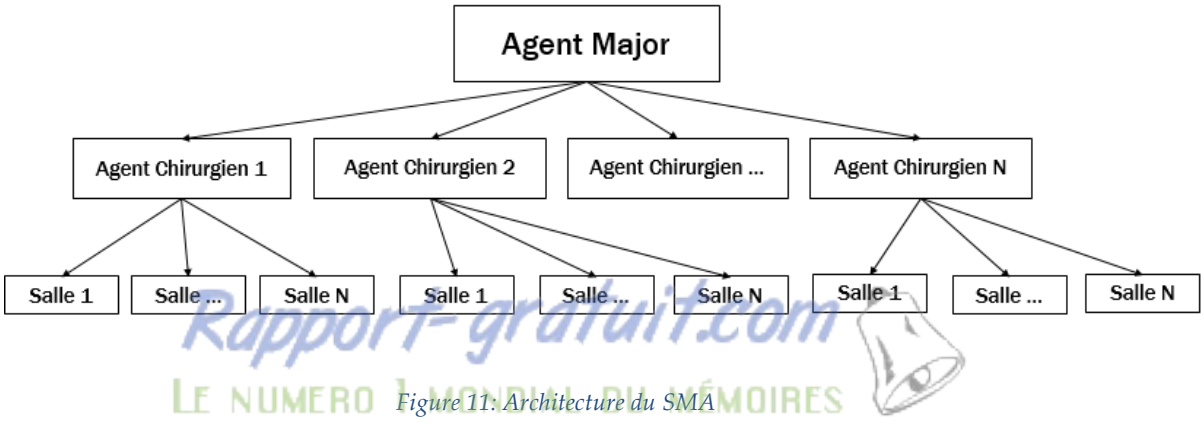


Figure 11: Architecture du SMA

Conclusion

Ce chapitre est divisé en deux grandes parties :

- La modélisation : dans cette partie, nous avons présenté le modèle mathématique, la méta-heuristique « recherche tabou » que nous avons mise en place pour construire un planning qui satisfait les contraintes citées dans le modèle mathématique et qui minimise la fonction objectif.
- La description du système : dans cette partie nous avons décrit l'architecture globale du système qui comprend la phase proactive et la phase réactive.
    - Dans la phase proactive, nous avons parlé de la méta-heuristique que nous avons choisi pour construire un planning de la semaine, ainsi que l'algorithme utilisé.
    - Dans la phase réactive, nous avons parlé de l'architecture du système multi agents que nous avons mis en place pour gérer les imprévus.

Chapitre 4 : Conception et implémentation

Introduction

Ce chapitre est conçu pour une représentation technique du système.

Dans un premier temps, nous allons présenter quelques diagrammes UML (diagramme de classe, de paquetage et de séquence). Puis nous allons présenter la plateforme multi-agents JADE qui va nous permettre de développer et exécuter les agents.

1. Conception

1.1. Diagramme de paquetage

Le projet est composé de trois packages : le premier contient les classes liées au problème, le deuxième contient les classes liées à la résolution du problème (la recherche Tabou) et le troisième contient les classes nécessaires à la mise en place du système multi agents.

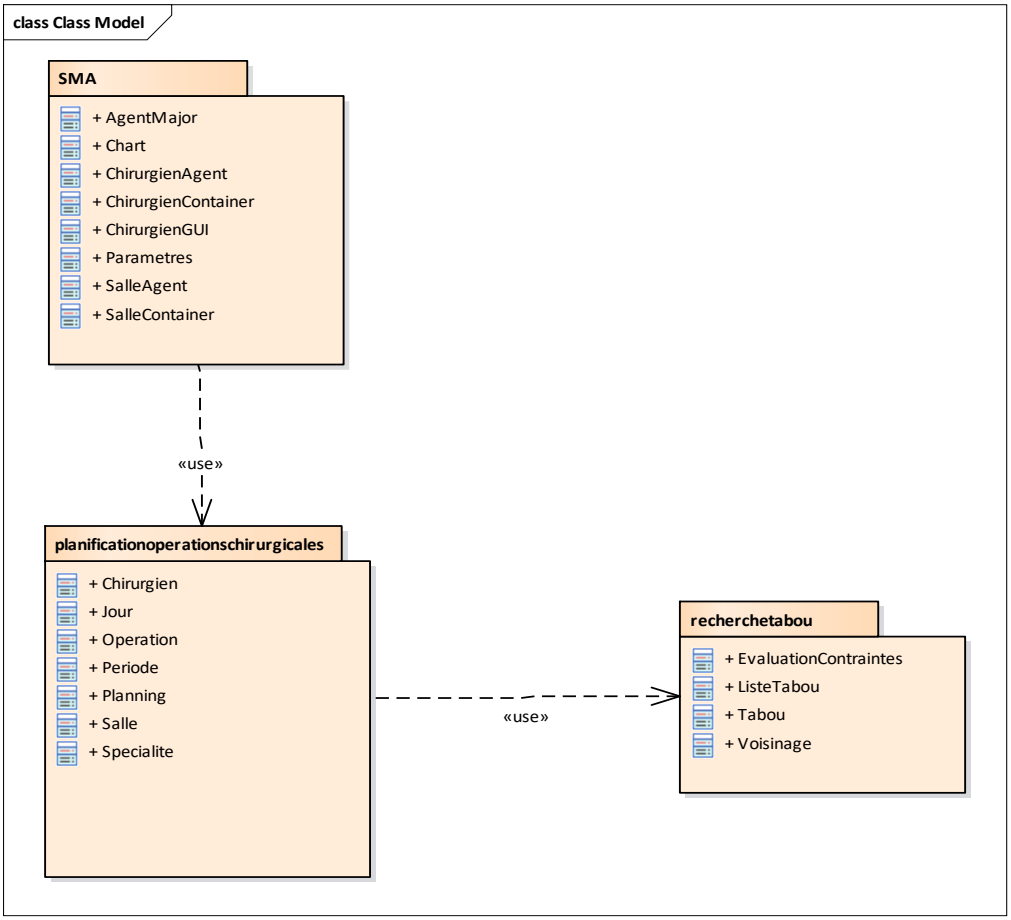


Figure 12: Diagramme de paquetage

Pour le premier package « planificationoperationschirurgicales », sept classes ont été définies :

- Classe *Operation* : contient des informations sur l'opération telles que la durée, spécialité, etc. Elle contient aussi des méthodes permettant de vérifier si l'opération est affectée à un chirurgien, une période/ jour ou une salle.
- Classe *Salle* : cet objet contient la spécialité de la salle, sa durée d'ouverture régulière par jour ainsi que le nombre d'heures supplémentaire, etc.
- Classe *Chirurgien* : cette classe contient en plus de la spécialité et le nombre d'heures de travail du chirurgien par jour, un attribut « *PeriodesDeTravail* » permettant de savoir les périodes du jour où le chirurgien est affecté, cela pourrai être utile si nous voudrions générer des emplois du temps des chirurgiens.
- Classe *Specialite* : contient une liste des salles et une liste des chirurgiens ayant cette spécialité.
- Classe *Periode* : contient des attributs tels que la liste des chirurgiens qui sont affectés à cette période du jour, la liste des opérations ainsi que les salles utilisées pendant cette période.
- Classe *Jour* : est composée d'un tableau des *periodes*.
- Classe *Planning* : c'est la classe principale qui contient toutes les méthodes permettant d'avoir une meilleure solution à partir de l'initialisation de la solution puis la recherche par voisinage et finalement l'obtention de la meilleure solution rencontrée. Cette classe prend en entrée un tableau des chirurgiens, salles, spécialités et opérations. Elle contient aussi un objet de type *EvaluationDesContraintes*, les quatre voisins (période, jour, salle et chirurgien) et un objet de type *listeTabou* pour vérifier si la solution courante a été déjà visité et l'empêcher si c'était le cas.

Pour le deuxième package « rechercheTabou », il y en a quatre classes :

- Classe *Tabou* : cette classe permet de mémoriser les déplacements effectués. Pour *période1*, *chirurgien1*, *salle1* et *jour1* correspondent à l'objet courant, et pour la *période2*, *chirurgien2*, *salle2* et *jour2* correspondent au mouvement qui va être réalisé pour atteindre le voisin. Dans ce travail, nous avons défini quatre voisins afin de diversifier les zones de recherches :
  - Le voisin période : un changement est effectué au niveau de la *periode1* sans modifier les autres.
  - Le voisin jour : un changement est effectué au niveau du *jour2* et peut être effectué aussi au niveau de la *periode2*.
  - Le voisin salle : un changement est effectué au niveau de la *salle2*.
  - Le voisin chirurgien : un changement est effectué au niveau du *chirurgien2*.
- Classe *ListeTabou* : contient une liste des objets Tabou et des méthodes pour ajouter un objet Tabou et vérifier son existence dans la liste.

- Classe *EvaluationDesContraintes*: cette classe permet d'extraire les conflits qui ont eu lieu dans la solution. Nous avons défini sept contraintes.
  - Contrainte1 : compte le nombre de fois où une salle est utilisée dans la même période.
  - Contrainte2 : compte le nombre de fois où un chirurgien est affecté à des opérations dans la même période du jour.
  - Contrainte3 : retourne 1 si la durée totale des opérations affectés à un chirurgien dépasse son nombre d'heures maximum de travail par jour.
  - Contrainte4 : retourne 1 si la durée totale des opérations affectés à une salle dépasse sa capacité (heures régulières et supplémentaires).
  - Contrainte5 : vérifier si l'opération est affectée à une période supplémentaire. Cette contrainte n'influence pas sur la solution, elle est considérée comme une contrainte faible, donc elle peut être violée.
  - Contrainte6 : retourne 1 si la salle et l'opération n'ont pas la même spécialité.
  - Contrainte7 : retourne 1 si le chirurgien et l'opération n'ont pas la même spécialité.

Dans le modèle mathématique que nous avons proposé dans la section 3.1.2 la contrainte *(3)* n'est pas défini dans la classe *EvaluationDesContraintes* car elle est satisfaite lors de la construction de la solution initiale.

Le troisième package « SMA » contient huit classes :

- SalleContainer : C'est la classe responsable de déploiement des agents salles, c'est une classe indispensable dans la plateforme JADE que nous allons en parler dans le chapitre suivant.
- SalleAgent : représente l'agent salle, il contient un objet de type salle contenant toutes les informations sur la salle qui vont permettre à l'agent de répondre aux requêtes envoyées par les autres agents.
- ChirurgienContainer : C'est la classe responsable de déploiement des agents chirurgiens.
- ChirurgienAgent : représente l'agent chirurgien, il contient un objet de type Chirurgien lui permettra de répondre aux requêtes envoyées par d'autres chirurgiens ou major.
- AgentMajor : gère la communication entre les différentes entités. Elle reçoit des requêtes de la part d'un agent et elle les renvoie aux autres agents en attendant qu'il reçoive leurs réponses pour choisir la meilleure, etc. Ce processus va être clarifié dans la partie suivante.
- ChirurgienGUI : représente l'interface graphique lié à l'agent chirurgien, sur laquelle le chirurgien peut déclarer son indisponibilité ou confirme le changement avec un autre chirurgien.
- Parametres : cette classe regroupe les paramètres qu'un agent envoie lors de sa requête. Elle joue un rôle facilitateur.
- Chart : c'est la classe responsable de l'affichage du graphe représentant la variation du coût en fonction du nombre d'itérations.

1.2. Diagramme de classe

1.2.1. Package « rechercheTabou »

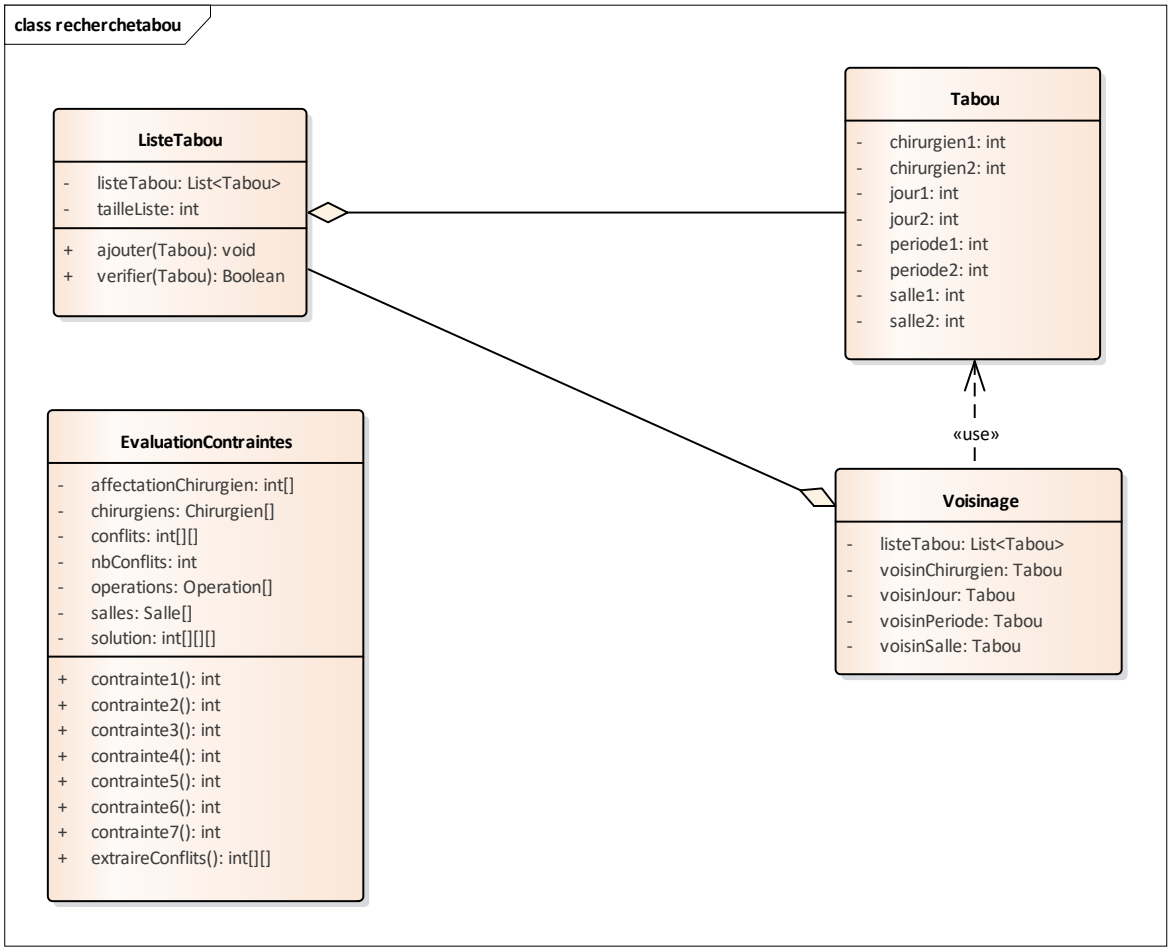


Figure 13: diagramme de classe pour le package "rechercheTabou"

1.2.2. Package « planificationoperationschirurgicales »

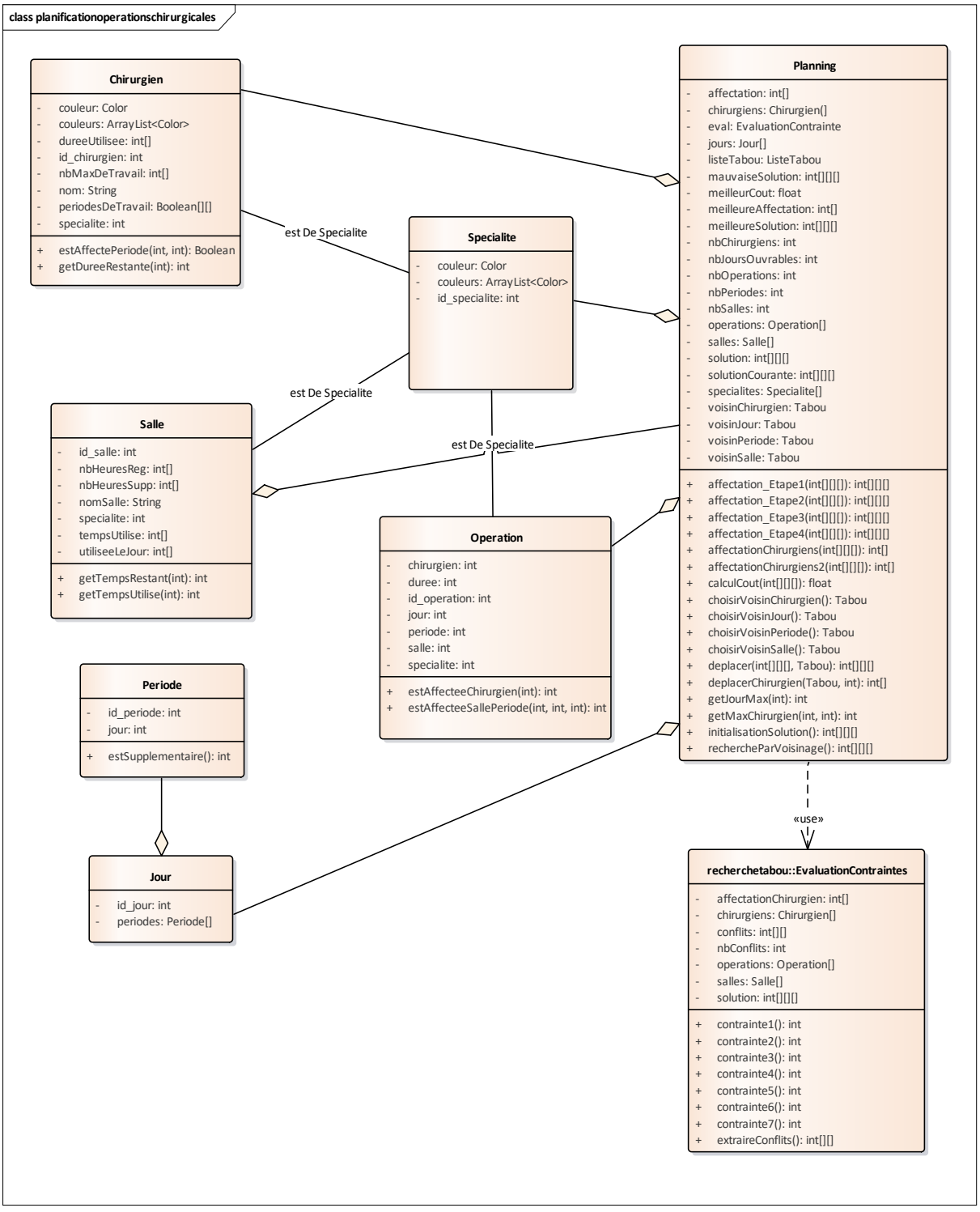


Figure 14: diagramme de classe pour la package "planificationoperationschirurgicales"

1.2.3. Package « SMA »

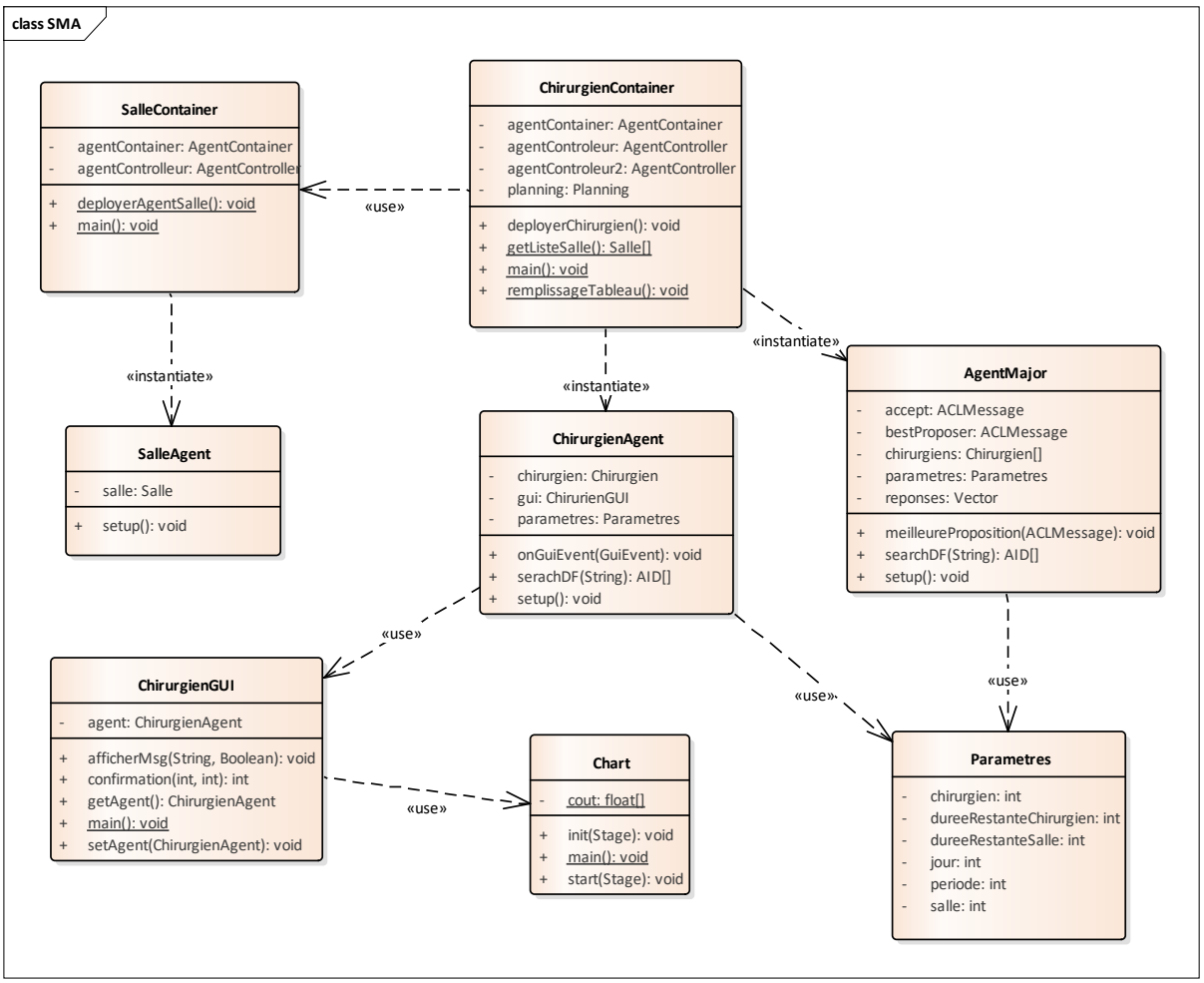


Figure 15: diagramme de classe pour le package "SMA"

1.3. Diagramme de séquence

Ce diagramme présente les interactions entre les différents agents du système.

Dans un premier temps, tous les agents doivent s'enregistrer auprès de l'AMS et publier leurs services auprès de DF.

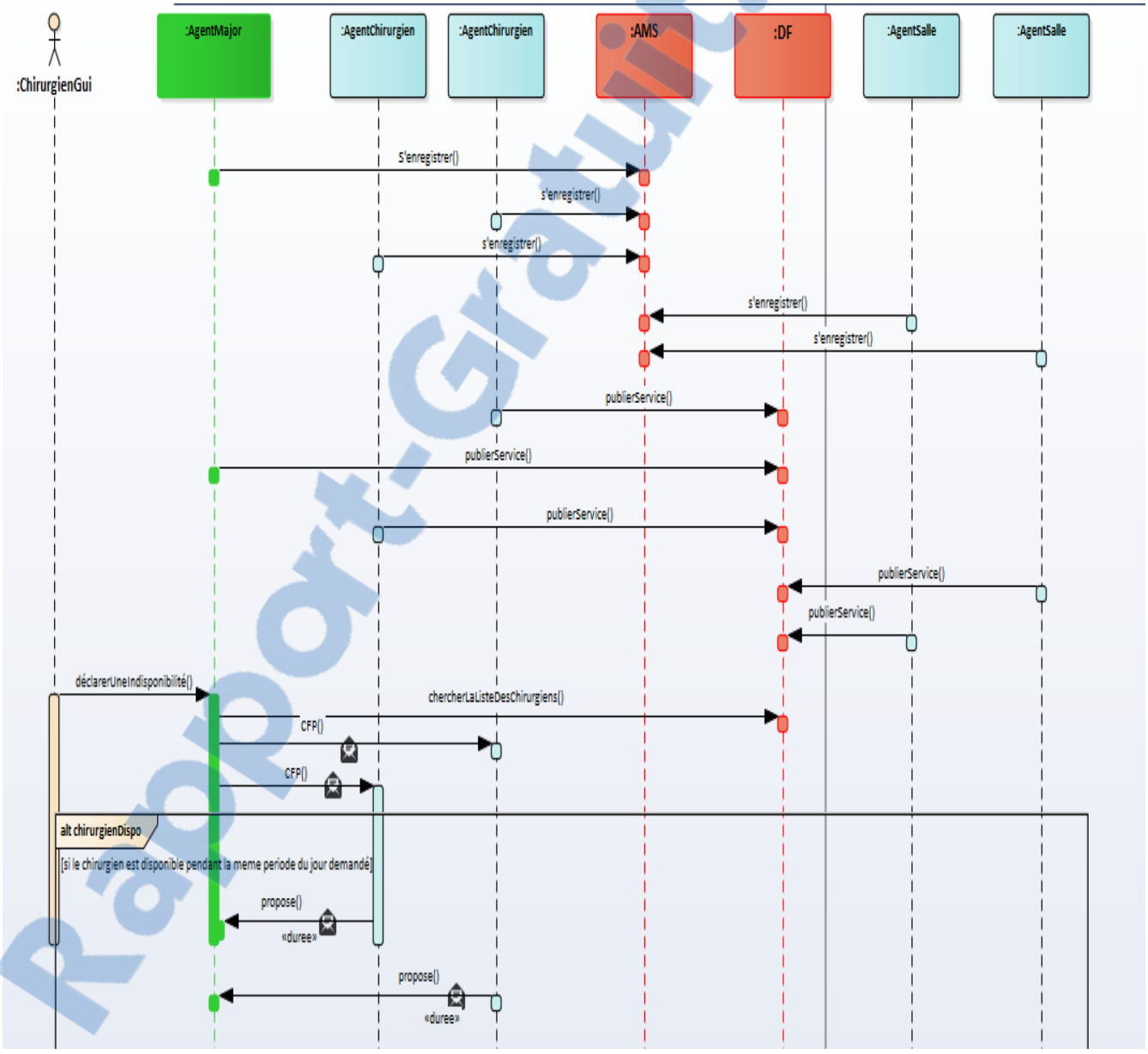
Le scénario se déclenche dès qu'un chirurgien déclare son indisponibilité via l'interface graphique ChirurgienGUI, cette demande est transférée à l'agentMajor qui gère la communication entre les agents.

L'AgentMajor de sa part, il demande la liste de tous les chirurgiens de même spécialité et envoie un CFP (call for proposer) à ces agents par diffusion.

Chaque agent va proposer une période, si cette période correspond à la même période demandée, il envoie cette réponse avec la durée de travail restante ce jour-là. Sinon, il envoie

une requête aux agents salles pour s'assurer qu'il y ait une salle libre pendant la période proposée.

Toutes les réponses des agents chirurgiens sont stockés et l'agentMajor choisira la meilleure proposition ; la priorité est donnée au chirurgien qui propose la même période demandée, s'il y en a beaucoup, il choisit celui qui propose la plus grande durée. Dans le cas échéant, l'AgentMajor choisira celui qui propose la plus proche période.



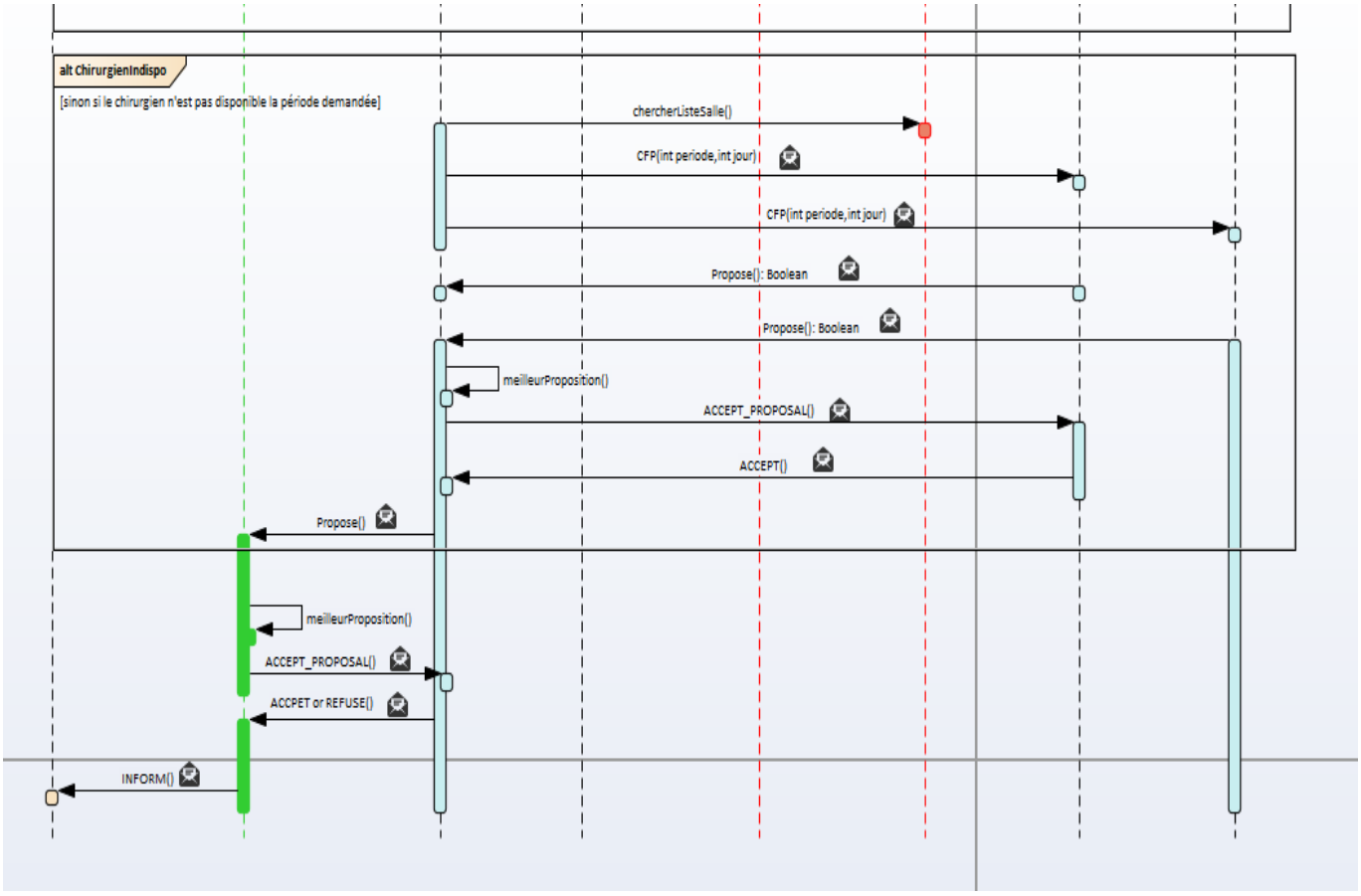


Figure 16: Diagramme de séquence

2. Implémentation

2.1. Présentation de la plateforme JADE

2.1.1. Introduction

JADE (Java Agent DEvelopment Framework) est une plateforme multi agent implémenté entièrement en langage Java et développé par CSELT (Groupe de recherche de Gruppo Telecom, Italie), son but est de simplifier la mise en œuvre des systèmes multi-agents grâce à un middleware conforme aux spécifications FIPA. Parmi les avantages qu'il comporte [43], nous citons:

- Un système basé sur JADE peut être distribué sur plusieurs machines.
- La configuration peut être contrôlée via une interface graphique distante.
- La configuration peut être modifiée au moment de l'exécution en déplaçant les agents d'une machine à l'autre.

La plateforme JADE contient :

- Un runtime environment : l'environnement où les agents peuvent vivre. Il doit être activé avant l'exécution des agents.

- Une librairie de classes : c'est une librairie que les programmeurs doivent utiliser pour développer leurs agents.
- Une suite d'outils graphiques : qui facilitent la gestion et la supervision de la plateforme des agents.

2.1.2. Plateformes et conteneurs

Chaque instance du JADE est appelée conteneur « container en anglais », car elle peut contenir plusieurs agents. L'ensemble des conteneurs constituent une plateforme, cette dernière doit contenir un conteneur spécial appelé main-container.

Un main-container contient deux agents spéciaux :

- AMS (Agent Management System) : est l'agent qui exerce un contrôle sur l'accès et l'utilisation de la plateforme, il fournit le service de nommage ; il possède un répertoire des identificateurs d'agents et leurs états. Chaque agent doit s'enregistrer auprès de l'AMS au démarrage.
- DF (Directory Facilitator) : est l'agent qui fournit un service de Pages Jaunes, au moyen duquel un agent peut trouver d'autres agents fournissant les services dont il a besoin pour atteindre ses objectifs.

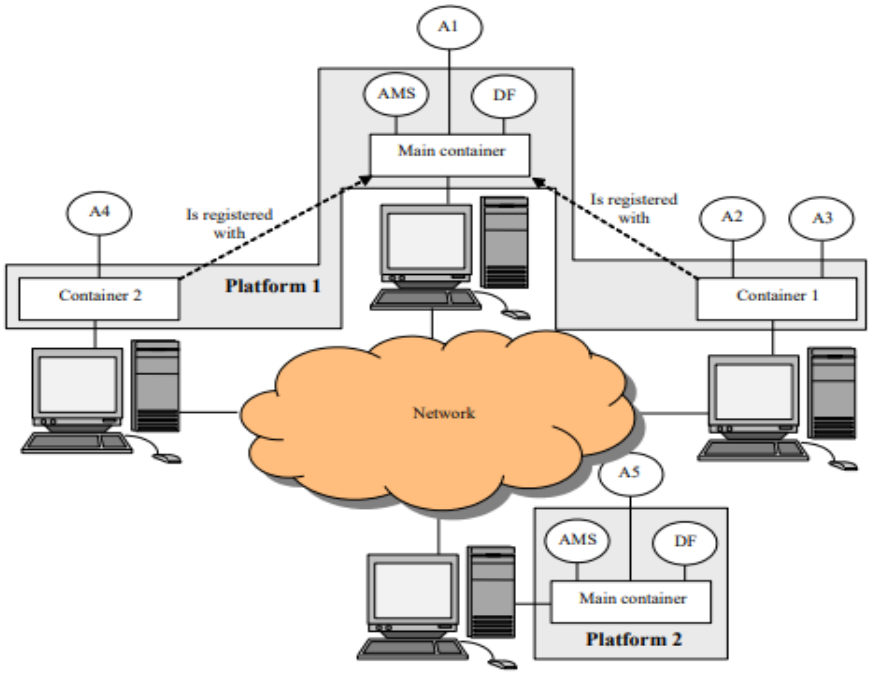


Figure 17: Les conteneurs et les plateformes dans JADE [44]

2.1.3. Le langage de communication entre les agents

JADE utilise le langage FIPA-ACL pour la communication entre les agents. Ce langage est considéré comme le langage standard des messages, et s'appuie sur la définition de deux ensembles [45]:

- Un ensemble d'actes de communication primitifs, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition de ces actes de base.
- Un ensemble de messages prédéfinis que tous les agents peuvent comprendre.

Un message FIPA-ACL est composé de :

- Emetteur : (Sender)
- Destinataires : (Receivers)
- Actes de communication : (Performatives)
- Contenu : (Content)
- Ontologie : (Ontologie)
- L'id de la conversation : (Conversation-id)

FIPA-ACL définit 21 actes de communications (ou performatives), nous citons quelques-uns :

- CFP : appel de proposition pour exécuter une action donnée.
- PROPOSE : Une proposition d'une action conditionnée à certaines préconditions données.
- ACCEPT_PROPOSAL : Accord de l'expéditeur d'effectuer certaines actions présentées à l'avance.
- INFORM : Communication par l'expéditeur d'une proposition, pensée vrai par celui-ci.
- REFUSE : Refus de l'expéditeur d'effectuer certaines actions.
- AGREE : Accord de l'émetteur d'effectuer une action.
- CONFIRM : L'émetteur informe le destinataire de la validité (selon les règles de l'agent) de la proposition préalablement reçue.
- REQUEST : Communication par l'expéditeur d'une demande au destinataire d'effectuer une action.

2.1.4. Comportements d'agents en JADE

Un agent JADE peut être composé de plusieurs comportements (behaviours), ce dernier représente une tâche qu'un agent peut effectuer et est implémenté en tant qu'objet d'une classe qui hérite de la classe Behaviour.

Chaque classe qui étend de Behaviour doit implémenter la méthode action(), qui définit les opérations à effectuer lorsque le comportement est en exécution et la méthode done(), qui spécifie si un comportement est terminé.

- Comportements simples
  - OneShotBehaviour : Ce comportement s'exécute une seule fois et ne peut pas être bloqué c.-à-d. qu'il se termine immédiatement, il implémente la méthode done() qui retourne *true*.
  - CyclicBehaviour : Ce comportement une fois qu'il est lancé ne se termine jamais, il implémente la méthode done() qui retourne *false*.
- Comportements composés : permettent d'implémenter des tâches complexes en JADE, en composant des comportements simples.
  - SequentialBehaviour : permet d'exécuter une série de tâches possèdant chacune un comportement, de manière enchaînée.
  - ParallelBehaviour : permet l'exécution des tâches en parallèle.

2.1.5. Outils JADE

JADE offre une interface graphique GUI permettant de contrôler et superviser les états des agents.

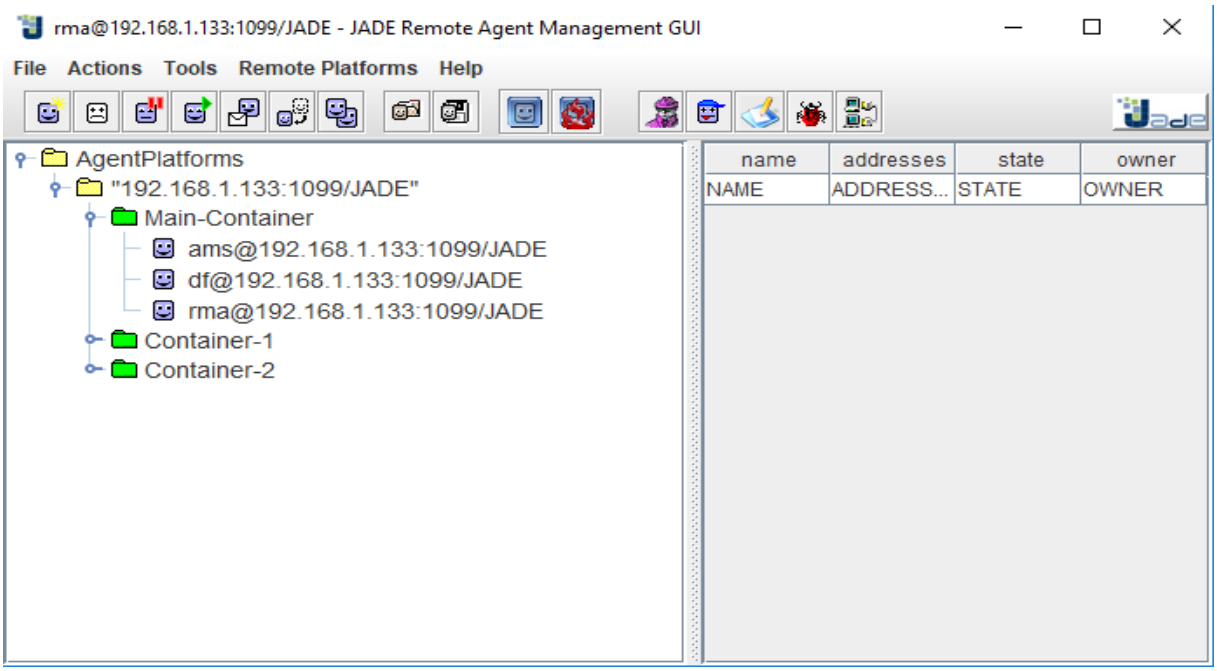


Figure 18: Interface graphique GUI de JADE

Elle propose ainsi quelques outils de gestion :

- RMA : Permet de gérer les agents d'une plateforme.
- Sniffer : Permet de visualiser et d'analyser les échanges de messages entre les agents.

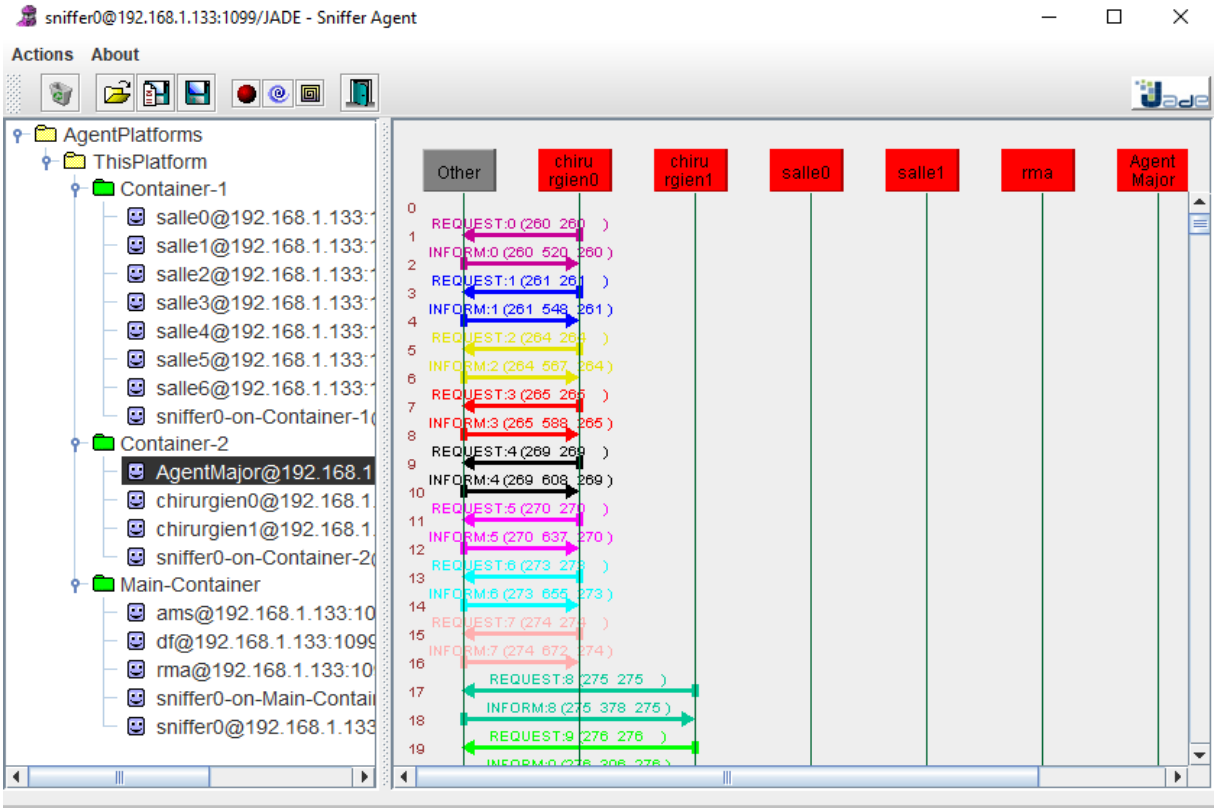


Figure 19: Agent Sniffer de la plateforme JADE

- Dummy : Permet d'envoyer des messages aux agents.

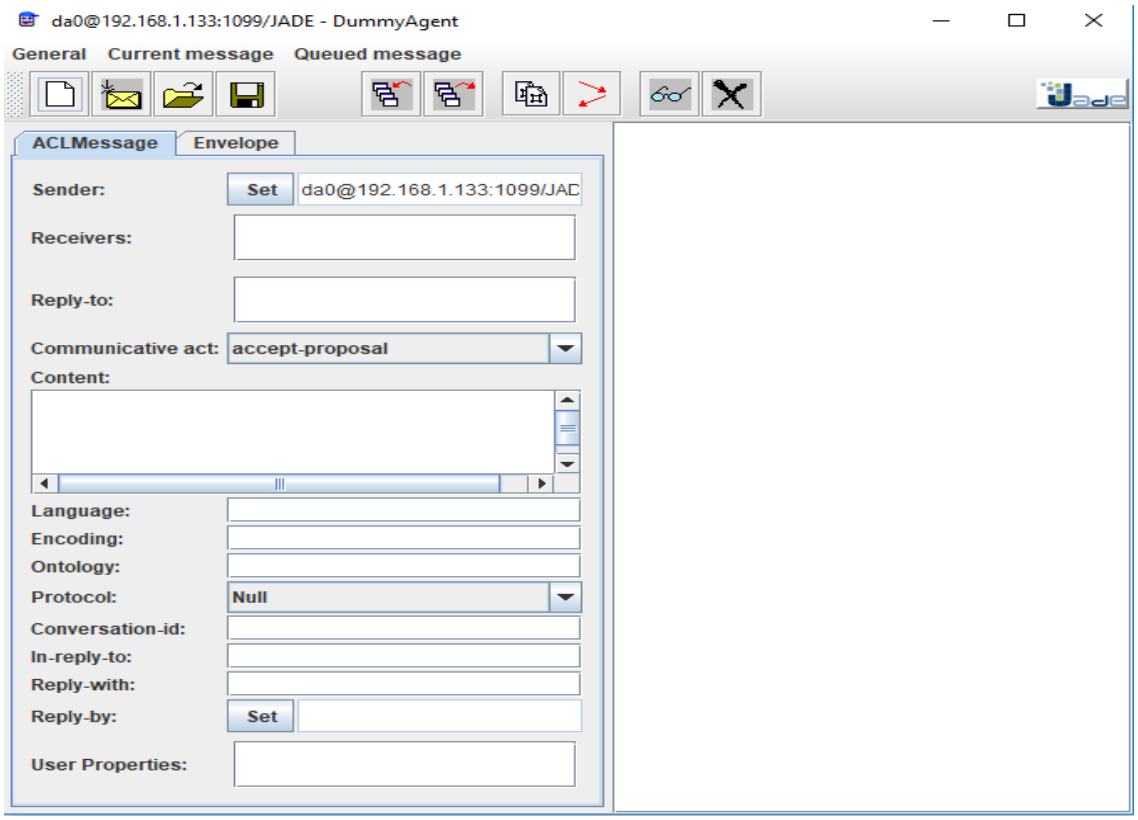


Figure 20: Agent Dummy de la plateforme JADE

2.2. Environnement matériel

Le modèle a été implémenté et exécuté sur un ordinateur portable Toshiba :

- Un processeur Intel (R) Core (TM) i7-2640M CPU vitesse 2.80GHZ.
- Mémoire RAM : 4G.
- Système d'exploitation Windows 10.

Conclusion

Au cours de ce chapitre, nous avons présenté quelques diagrammes UML permettant la conception du projet, il inclut le diagramme de paquetage, de classe pour chaque package et le diagramme de séquence illustrant la communication inter-agents.

Dans un second lieu, nous avons parlé de la plateforme JADE comme étant un environnement de travail logiciel, et donc nous avons parlé des outils qu'elle offre, le langage de communication entre les agents ainsi que leurs comportements dans la plateforme JADE.

Et finalement, nous avons décrit brièvement l'environnement matériel de travail.

Chapitre 5 : Expérimentations et Résultats

Introduction

Dans ce chapitre nous allons présenter les expérimentations que nous avons réalisé afin de tester les performances du système pour la phase proactive. Puis, nous allons présenter les interfaces graphiques qui permettent au chirurgien d'afficher son planning de la semaine et de déclarer son indisponibilité. Et qui permettent au major d'afficher tout le planning de la semaine.

1. Outils de développement

1.1. IBM ILOG CPLEX Optimization Studio

Nous avons utilisé pour faire tourner notre modèle mathématique le logiciel CPLEX Studio [1] qui constitue le moyen le plus rapide pour créer des modèles d'optimisation efficaces et des applications à la pointe de la technologie pour tous les problèmes d'ordonnancement et de planification. Avec son environnement de développement intégré, un langage de modélisation descriptif et des outils intégrés, il prend en charge l'ensemble du processus de développement de modèle.

1.2. Netbeans

Pour le développement de TS et SMA nous avons utilisé le logiciel Netbeans [2] qui un Environnemet de Développement Intégré contient un ensemble d'outils, venant aider les programmeurs. Il est spécialisé dans le domaine Java.

2. Expérimentations

Nous présentons dans cette partie les différents tests effectués afin de tester les performances de l'algorithme. Ainsi, nous allons présenter l'implémentation du SMA pour la phase réactive.

2.1. Phase proactive

Dans ce rapport nous avons utilisé deux catégories des expérimentations :

- Expérimentations avec des petites instances, pour comparer les solutions de MILP avec celles de la TS proposé.
- Expérimentations avec un cas réel, c.-à-d. des grandes instances, dans laquelle le logiciel CPlex se plante et nous n'arriverons pas à avoir des solutions par une méthode exacte.

2.1.1. Expérimentations 1

Nous considérons dans cette catégorie :

- Nombre de salles : 3
- Nombre de chirurgiens : 4
- Nombre de spécialité : 2
- Nombre de jours ouvrables : 5
- Nombre de périodes du jour : 12

Le tableau représente les résultats obtenus par MILP

Nombre total des opérations	Fonction Objectif	Temps d'exécution (s)
25	505	136 ,67
35	295	298,76
45	300	478,84
55	575	1030,67
65	705	1474,39
75	-	-

Tableau 3: Résultats des expérimentations 1 avec MILP

Le tableau représente les résultats obtenus par TS avec une condition d'arrêt à 5000 itérations.

Nombre total des opérations	Fonction Objectif	Temps d'exécution (s)
25	505	4,486
35	295	19,083
45	300	4,611
55	575	10,602
65	705	5,824
75	495	10,311

Tableau 4: Résultats des expérimentations 1 avec TS

2.1.2. Expérimentations 2

Dans cette catégorie d'expérimentations, nous allons nous approcher de la réalité pour tester l'algorithme.

Nous considérons treize spécialités :

- Neurochirurgie
- Chirurgie cardio-vasculaire
- Urologie
- Chirurgie viscérale
- Chirurgie thoracique
- Ophtalmologie
- Chirurgie pédiatrique
- Chirurgie traumatologique
- Chirurgie vasculaire
- Chirurgie maxillo-faciale
- O.R.L
- Gynécologie
- Orthopédie

Avec :

Nombre de salle d'opérations : 13

Nombre de chirurgiens : 26

Nombre de jours ouvrable : 5

Nombre de périodes par jour : 12

Le tableau ci-dessous montre les résultats obtenus :

Nombre des opérations par spécialité	Nombre total des opérations	Fonction Objectif	Temps d'exécution (s)
20	260	1875	123,820
30	390	2800	208,201
40	520	3750	281,497
50	650	4675	246,163
60	780	2460	32,142

Tableau 5: Résultats des expérimentations 2

2.1.3. Résultats

Le figure ci-dessous montre la comparaison du temps d'exécution de la méta-heuristique TS et de MILP.

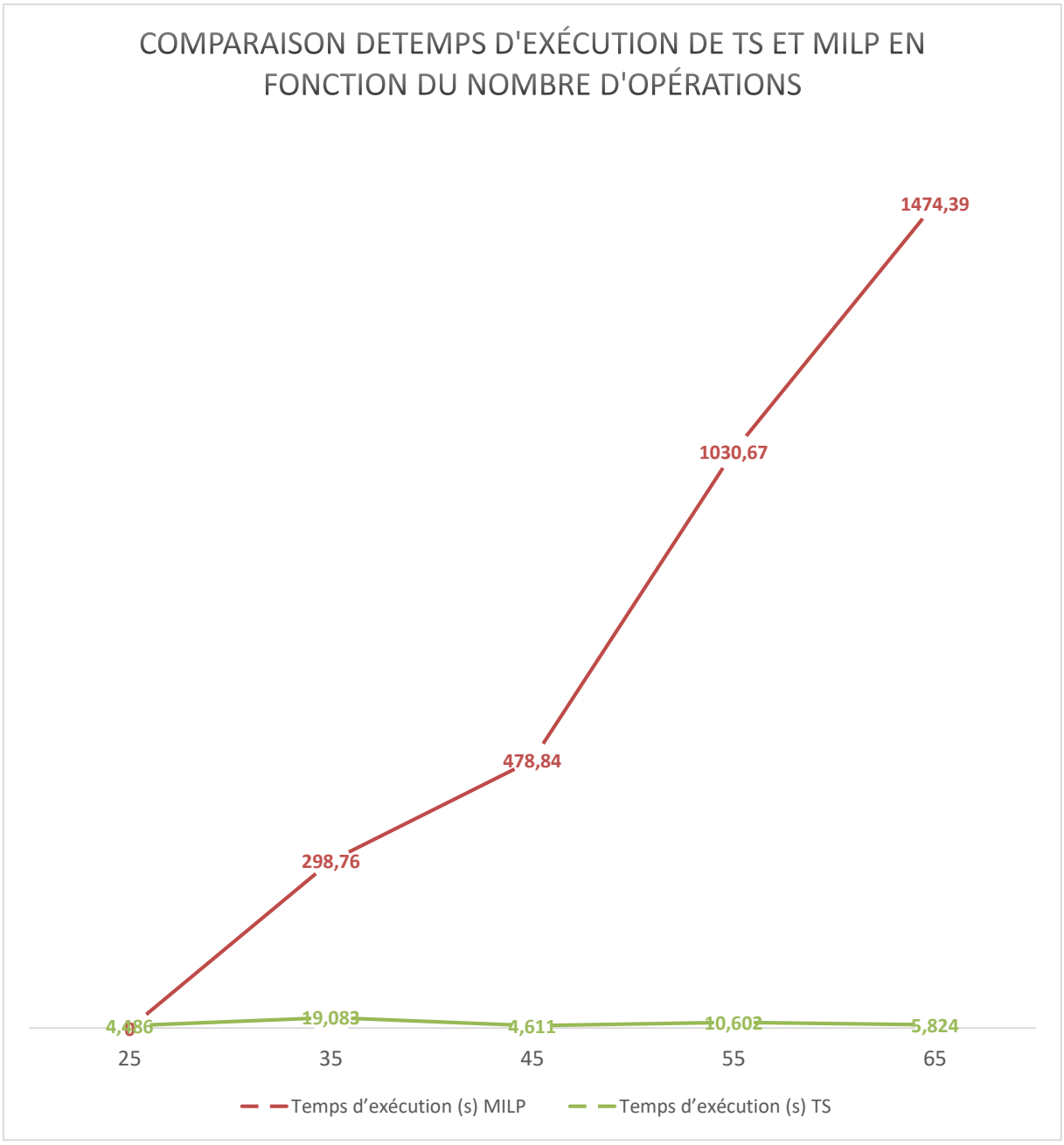


Figure 21: Comparaison entre le temps d'exécution de TS et de MILP

Nous remarquons que le temps d'exécution augmente énormément en fonction du nombre d'opérations pour MILP, par contre pour le TS, il ne dépasse pas les 20 seconds pour des petites instances.

Le figure ci-dessous montre une comparaison entre les valeurs de la fonction objectif obtenues par MILP et celles de TS.

Et nous constatons que TS a pu atteindre les solutions optimales pour les petites instances.

A partir du nombre d'opérations 75, MILP ne peut plus donner de solution à cause d'un problème de mémoire insuffisante.

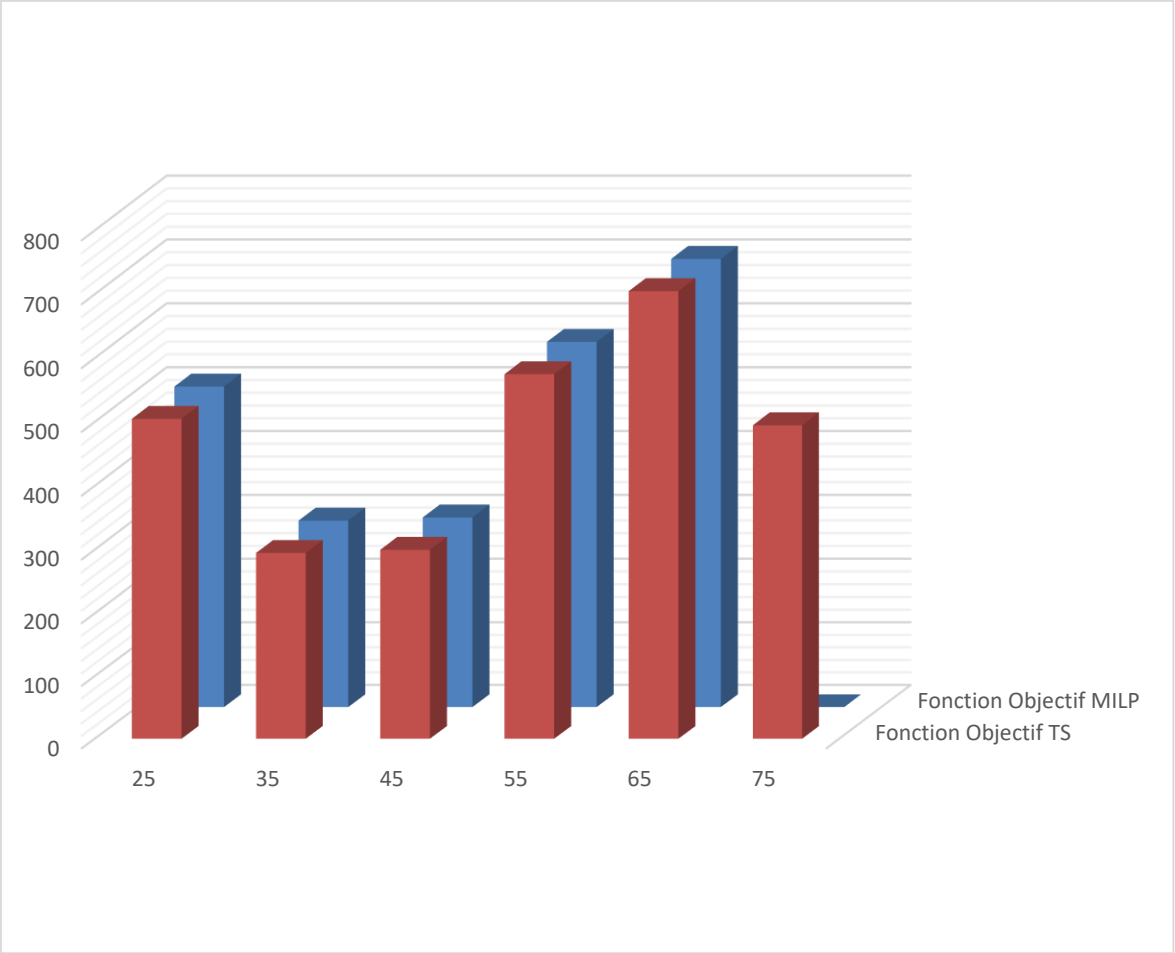


Figure 22: Comparaison entre les solutions obtenues par MILP et TS

D'après les différents tests effectués, nous remarquons que l'algorithme proposé atteint les solutions optimales pour chaque petite instance avec un temps d'exécution minimale par rapport au MILP. Par contre, pour des grandes instances, les méthodes exactes restent ineffectives et ne donnent pas des résultats.

2.2. Phase réactive

Notre application est constituée d'une interface graphique principale permettant l'affichage du planning et les chirurgiens qui y sont assignés, elle permet ainsi d'afficher la courbe qui montre la variation du coût lors de l'exécution de l'algorithme. Elle contient un bouton qui permet de déployer les différents chirurgiens du système. Et un bouton pour rafraîchir l'affichage du planning, si jamais il y a un changement.

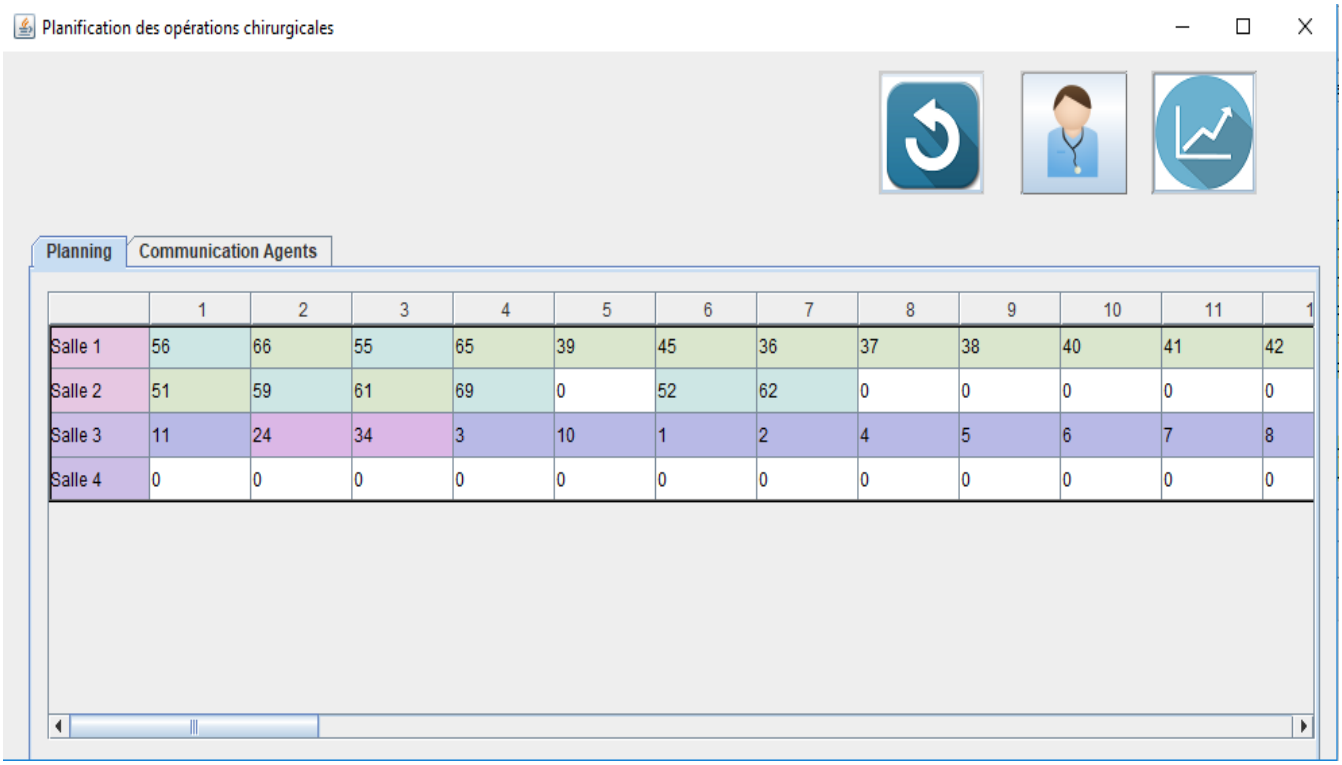
2.2.1. Interface principale

Cette interface est lancée une seule fois, elle affiche le planning qui a été construit dans la phase proactive.

2.2.1.1. Affichage du planning

Pour les salles de même spécialité, elles sont représentées par la même couleur. Les numéros affichés dans les cases correspondent aux indices des opérations, et les opérations de même couleur indiquent que c'est le même chirurgien qui va les réaliser.

Les cases qui ne sont pas colorées indiquent que la période est libre.



	1	2	3	4	5	6	7	8	9	10	11	1
Salle 1	56	66	55	65	39	45	36	37	38	40	41	42
Salle 2	51	59	61	69	0	52	62	0	0	0	0	0
Salle 3	11	24	34	3	10	1	2	4	5	6	7	8
Salle 4	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23: Interface principale du Major

2.2.1.2. Variation du coût

En cliquant sur ce bouton  nous obtiendrons le graphe de la variation des coûts.

Le graphe ci-dessous montre la variation de la fonction objectif en fonction du nombre d'itérations.

Figure 24 : Graphe de la variation du coût

2.2.1.3. **Déploiement d'un chirurgien**

Ce bouton  permet le déploiement des différents chirurgiens de l'hôpital pour communiquer entre eux et échanger des informations concernant leurs disponibilités et la durée de travail restante pour chaque jour.

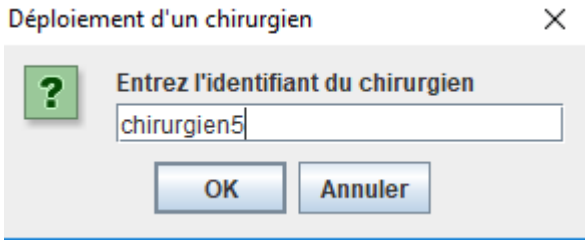
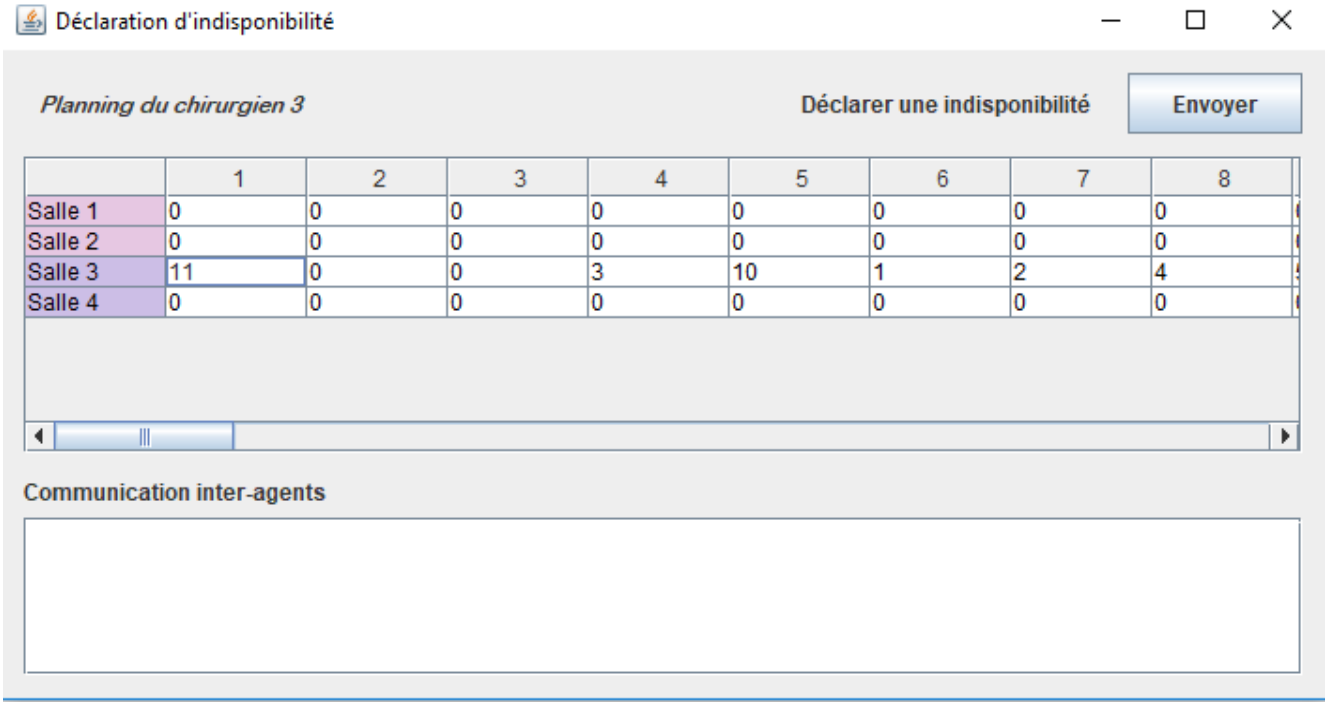


Figure 25: Interface de déploiement du chirurgien

2.2.2. **ChirurgienGUI**

La deuxième interface graphique est liée à l'agent chirurgien. Par laquelle un chirurgien peut déclarer son indisponibilité.

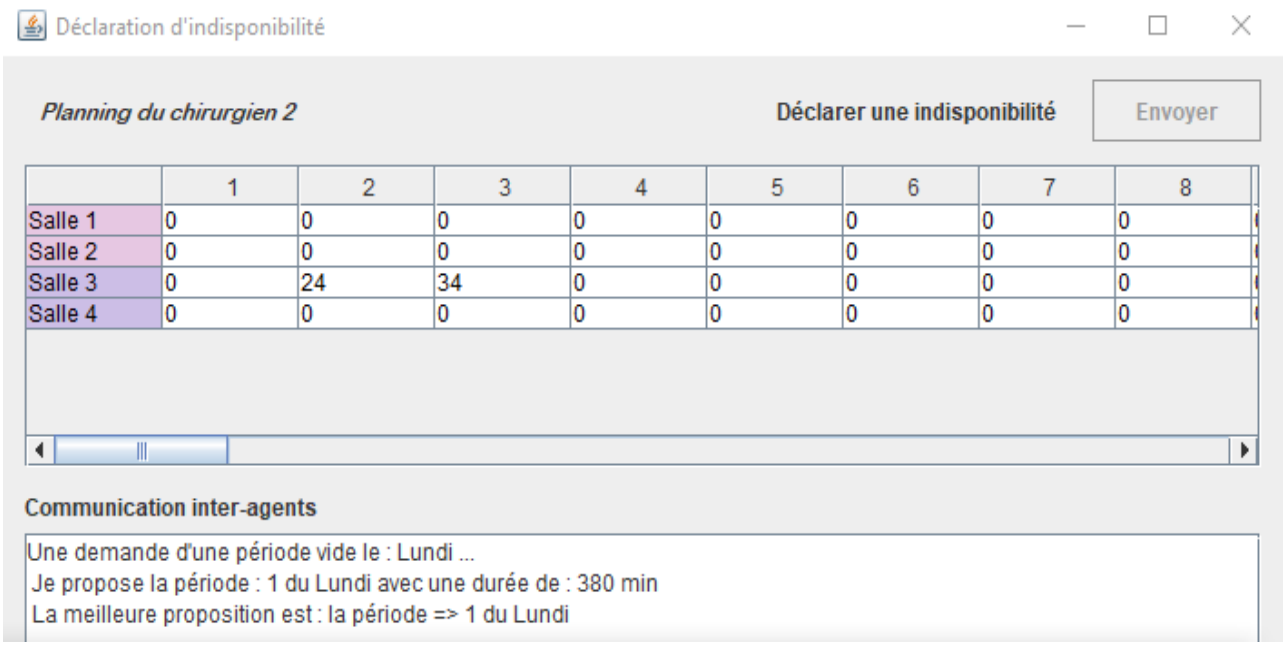
Chaque agent chirurgien est lié à une interface permettant d'afficher son planning de la semaine, c.-à-d. les périodes de travail et les opérations lui sont affectées. Le chirurgien pourra cliquer sur la case dont il est indisponible et cliquer sur envoyer. A ce moment-là, le processus de communication entre agents se déclenche et il sera afficher dans la zone qui est juste en dessous.



	1	2	3	4	5	6	7	8
Salle 1	0	0	0	0	0	0	0	0
Salle 2	0	0	0	0	0	0	0	0
Salle 3	11	0	0	3	10	1	2	4
Salle 4	0	0	0	0	0	0	0	0

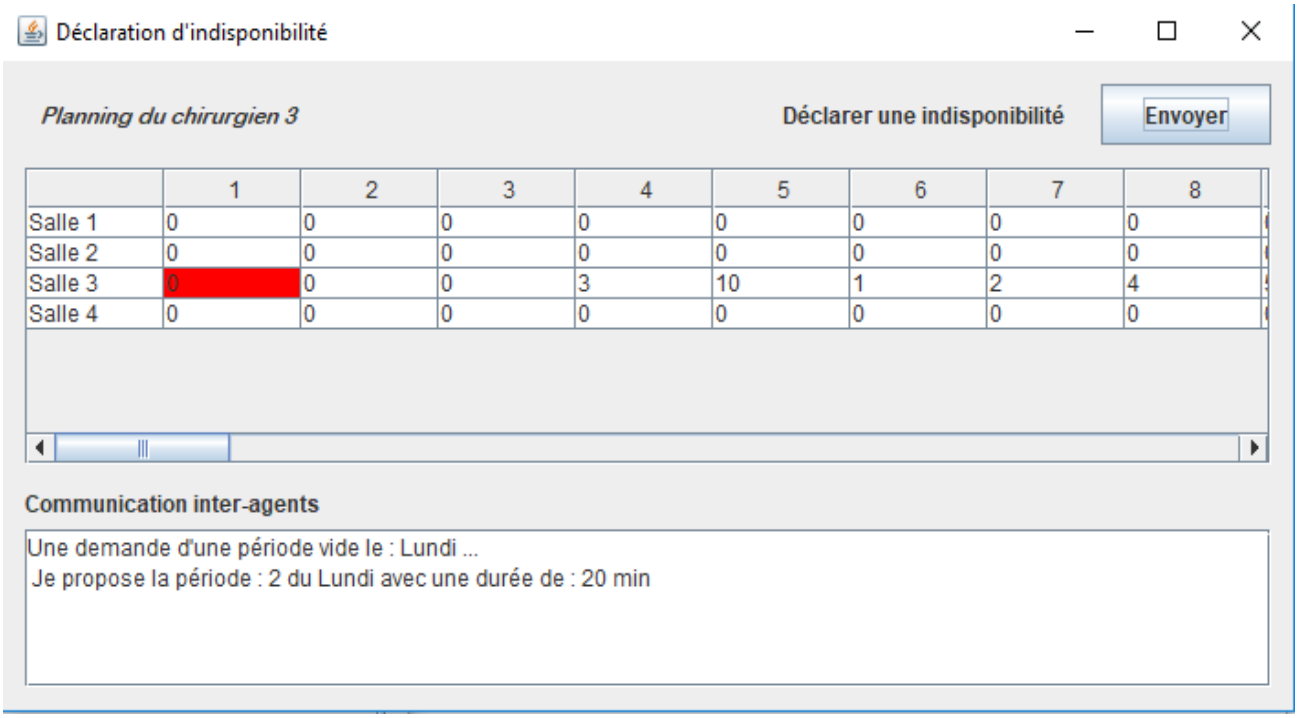
Figure 26: Interface représentant le planning du chirurgien 3

Chaque chirurgien reçoit une demande de période libre, et à son tour répond par une proposition d'une période libre du jour et la durée restante ce jour-là.



	1	2	3	4	5	6	7	8
Salle 1	0	0	0	0	0	0	0	0
Salle 2	0	0	0	0	0	0	0	0
Salle 3	0	24	34	0	0	0	0	0
Salle 4	0	0	0	0	0	0	0	0

Figure 27: Interface représentant le planning du chirurgien 2



	1	2	3	4	5	6	7	8
Salle 1	0	0	0	0	0	0	0	0
Salle 2	0	0	0	0	0	0	0	0
Salle 3	0	0	0	3	10	1	2	4
Salle 4	0	0	0	0	0	0	0	0

Figure 28: Interface représentant le planning du chirurgien 3 après changement

Le chirurgien qui a la meilleure proposition, va être informé et un message de confirmation va être apparu.

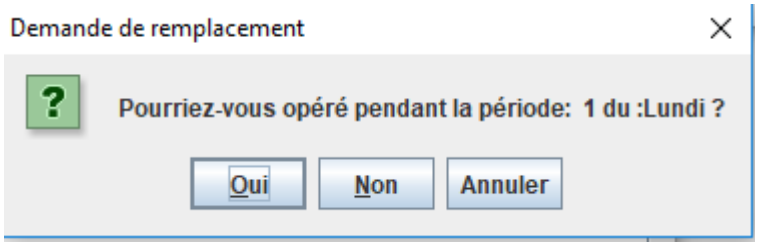


Figure 29: Message de confirmation

Si le chirurgien clique sur oui, un changement de planning aura eu lieu, dans sa propre interface et évidemment sur l'interface principale.

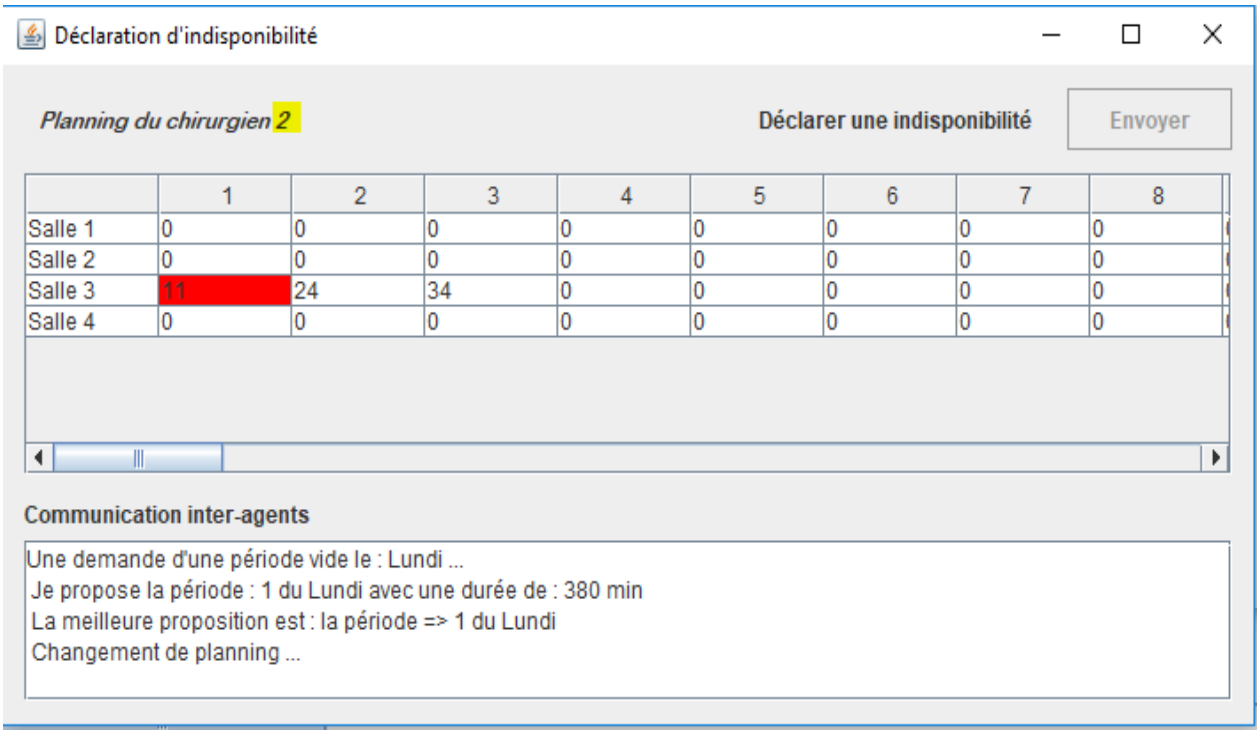
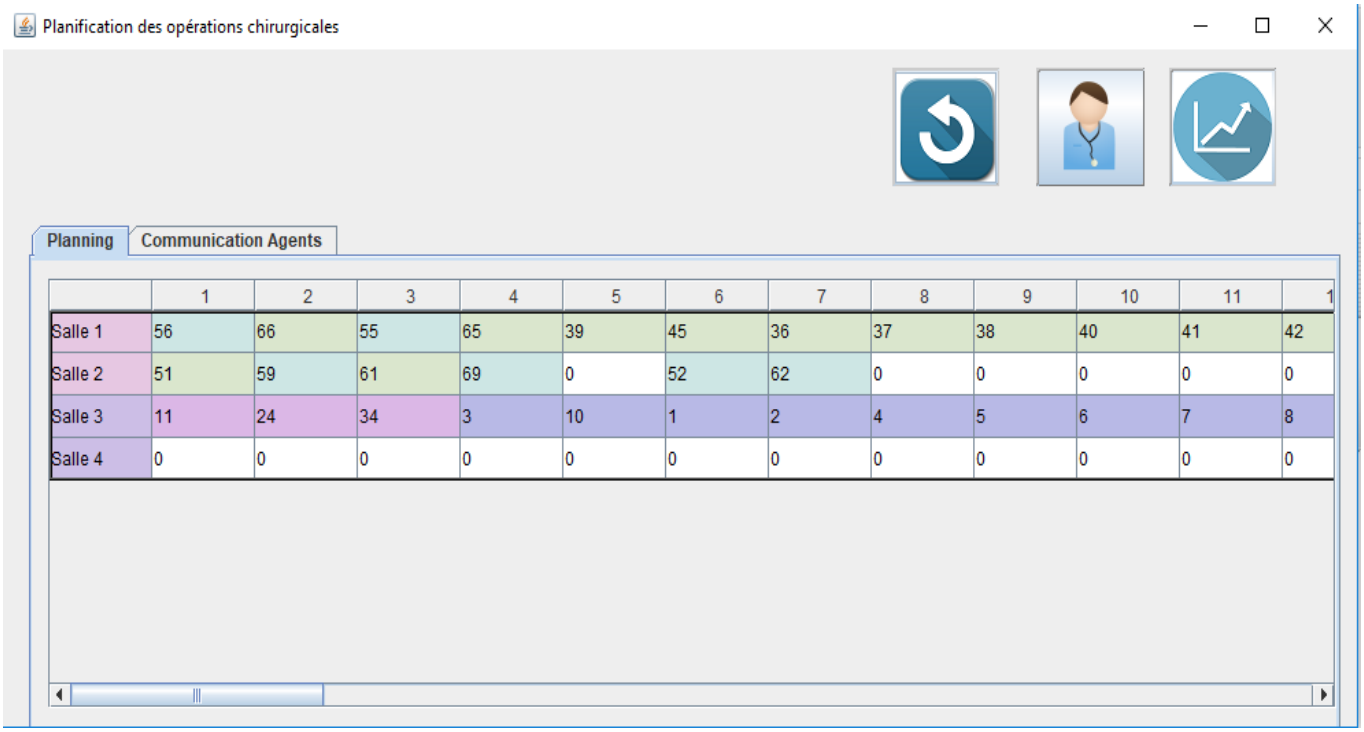


Figure 30: Interface représentant le planning du chirurgien 2 après changement

Après le rafraîchissement du planning par le bouton 

La case correspond à la période 1 et la salle 3 change de couleur, cela veut dire que le chirurgien a été remplacé.



	1	2	3	4	5	6	7	8	9	10	11	1
Salle 1	56	66	55	65	39	45	36	37	38	40	41	42
Salle 2	51	59	61	69	0	52	62	0	0	0	0	0
Salle 3	11	24	34	3	10	1	2	4	5	6	7	8
Salle 4	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31: Interface principale du Major après changement de planning

Conclusion

Dans ce chapitre, nous avons présenté au premier lieu, les différentes expérimentations effectuées en deux étapes :

La première étape contient des petites instances permettant la comparaison des solutions de MILP avec celles de TS. Et cela a montré l'efficacité de l'algorithme de pouvoir atteindre les solutions optimales dans un temps compétitif par rapport au MILP.

La deuxième étape contient des grandes instances qui sont proche à la réalité pour mesurer les performances de TS. Et donc l'algorithme donne des résultats satisfaisants.

Conclusion générale

Dans ce travail nous avons traité le problème de planification et d'ordonnancement des opérations chirurgicales dans le bloc opératoire, ces deux tâches qui sont considérés comme des phases importantes de la gestion de l'OT.

L'approche proposée consiste en deux phases :

- La phase proactive : permettant d'établir le planning de la semaine en considérant certaines contraintes et en minimisant la fonction objectif liée à la sous-utilisation des salles d'opérations et sur utilisation de celles-ci. Ce problème a été résolu par la méta heuristique TS et les résultats ont été comparé aux solutions MILP pour les petites instances, le TS proposé atteint la solution optimale pour chaque instance testée dans un temps d'exécution compétitif. Pour les problèmes de grande taille, notre algorithme a été expérimenté avec succès et a donné des résultats satisfaisants et encourageants.
- La phase réactive : a pour objectif de gérer les imprévus quotidiens, parmi lesquels il y a l'indisponibilité des chirurgiens, pour cela nous avons mis en œuvre un système multi-agents permettant la communication entre les agents afin de remplacer le chirurgien indisponible par un autre. Si ce n'est plus possible, de changer le planning.

Mais le bloc opératoire contient ainsi des salles d'hospitalisation et des salles de réveil, d'où la nécessité de tenir compte d'autres ressources telles que la disponibilité des anesthésistes, brancardiers, disponibilité des lits dans la salle de réveil et d'hospitalisation, etc.

Donc, comme perspectives, nous voudrions prendre en charge d'autres ressources matérielles : la salle de réveil et la salle d'hospitalisation. Et prendre en compte aussi la disponibilité des infirmiers et anesthésistes.

Pour la phase réactive, nous voudrions intégrer un autre aléas qui est la panne d'un matériel qui rend la salle non opérationnelle. Et donc la résolution se fera par le système multi agents en intégrant un scénario de panne matériel.

Bibliographie et Webographie

[1] N. Community. Available: https://fr.netbeans.org/edi/articles/concours/presentation-netbeans40-1.html

[2] IBM.Available: https://www.ibm.com/support/knowledgecenter/fr/SSSA5P_12.6.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_relnotes_intro.html

[3] A. Macario, T. Vitez, B. Dunn, and T. McDonald, "Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care," *Anesthesiology: The Journal of the American Society of Anesthesiologists,* vol. 83, no. 6, pp. 1138-1144, 1995.

[4] S. CHAABANE, "OPERATING THEATRE PREDICTIVE MANAGEMENT," INSA de Lyon, 2004.

[5] A. Hanset, D. Duvivier, O. Roux, and N. Meskens, "Programmation par contraintes pour l'ordonnancement d'un bloc opératoire," *Congrès International de Génie Industriel (CIGI 09),* 2009.

[6] J. M. Magerlein and J. B. Martin, "Surgical demand scheduling: a review," *Health services research,* vol. 13, no. 4, p. 418, 1978.

[7] M. Lamiri, "Operating Rooms Planning Under uncertainties," Ecole Nationale Supérieure des Mines de Saint-Etienne, 446 GI, 2007.

[8] F. Dexter, A. Macario, R. D. Traub, M. Hopwood, and D. A. Lubarsky, "An operating room scheduling strategy to maximize the use of operating room block time: computer simulation of patient scheduling and survey of patients' preferences for surgical waiting time," *Anesth Analg,* vol. 89, no. 1, pp. 7-20, Jul 1999.

[9] Guillaume GOUDENEGE, "Etat de l'art sur les modèles de planification des ressources humaines dans les établissements de santé," Génie Industriel, Ecole centrale Paris, 2009.

[10] H. FEI, "Vers un outil d'aide à la planification et à l'ordonnancement des blocs opératoires ", Sciences de Gestion FACULTÉS UNIVERSITAIRES CATHOLIQUES DE MONS, 2006.

[11] A. Jebali, "Vers un outil d'aide à la planification et à l'ordonnacement des ressources dans les services de soins," Institut National Polytechnique de Grenoble - INPG, 2004.

[12] M. A. Aziz Alaoui. Available: http://lmah.univ-lehavre.fr/index.php/axes-de-recherche/191-optimisation-et-recherche-operationnelle-2

[13] M. R. Serik, "Implémentation de méthodes de recherches locales sur les architectures multi et many-cœurs. Application au problème d'affectation quadratique à 3 dimension. THÈSE."

[14] M. Jaoua, "Algorithme de recherche tabou pour la planification optimale d'une campagne marketing sur les moteurs de recherche," École Polytechnique de Montréal, 2014.

[15] M. Widmer, "Les métaheuristiques: des outils performants pour les problèmes industriels," in *3ème Conférence Francophone de MOdélisation et SIMulation MOSIM*, 2001, vol. 1, pp. 25-27.

[16] H. Fei, C. Chu, N. Meskens, and A. Artiba, "Solving surgical cases assignment problem by a branch-and-price approach," *International Journal of Production Economics,* vol. 112, no. 1, pp. 96-108, 2008.

[17] A. Guinet and S. Chaabane, "Operating theatre planning," *International Journal of Production Economics,* vol. 85, no. 1, pp. 69-81, 2003.

[18] N. Saadani, A. Guinet, and S. Chaabane, "Ordonnancement des blocs operatoires," in *MOSIM: Conference francophone de MOdélisation et SIMulation*, 2006, vol. 6.

[19] H. Fei, N. Meskens, and C. Chu, "A planning and scheduling problem for an operating theatre using an open scheduling strategy," *Computers & Industrial Engineering,* vol. 58, no. 2, pp. 221-230, 2010.

[20] V. Augusto, X. Xie, and V. Perdomo, "Operating theatre scheduling with patient recovery in both operating rooms and recovery beds," *Computers & Industrial Engineering,* vol. 58, no. 2, pp. 231-238, 2010.

[21] P. Landa, R. Aringhieri, P. Soriano, E. Tànfani, and A. Testi, "A hybrid optimization algorithm for surgeries scheduling," *Operations Research for Health Care,* vol. 8, pp. 103-114, 2016.

[22] M. Gourgand, J. Pensi, and A. Tanguy, "PROGRAMMATION INTEGREE DES INTERVENTIONS CHIRURGICALES ET DES ACTIVITES DE MAINTENANCE PREVENTIVE," in *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*, 2014.

[23] R. Guido and D. Conforti, "A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem," *Computers & Operations Research,* vol. 87, pp. 270-282, 2017.

[24] K. Stuart and E. Kozan, "Online scheduling in the operating theatre," *Industrial Engineering and Management Society,* pp. 801-807, 2009.

[25] J. M. Molina-Pariente, E. W. Hans, J. M. Framinan, and T. Gomez-Cia, "New heuristics for planning operating rooms," *Computers & Industrial Engineering,* vol. 90, pp. 429-443, 2015.

[26] M. Souki, S. B. Youcef, and A. Rebai, "Planification des blocs opératoires," *ROADEF 2009,* p. 203, 2009.

[27] F. Jacques, "Les Systèmes Multi-agents, Vers une intelligence collective," *InterEditions, Paris,* vol. 322, 1995.

[28] N. R. Jennings and M. Wooldridge, "Applications of intelligent agents," in *Agent technology*: Springer, 1998, pp. 3-28.

[29] H. E. Nouri and O. Belkahla, *Résolution multi-agents du problème d'emploi du temps universitaire*. Éditions universitaires européennes, 2015.

[30] B. Chaib-Draa, I. Jarras, and B. Moulin, "Systèmes multi-agents: principes généraux et applications," *Principes et architectures des systèmes multi-agents,* 2001.

[31] A. M. Florea, "Agents et systèmes multi-agents," *Université Politechnica de Bucarest*.

[32] M. Sabar, "Une approche à base d'agents pour la planification et l'ordonnancement en temps réel de personnel dans un contexte de chaîne d'assemblage flexible," Université Laval, 2008.

[33] M. KLABI Hichem, "Vers un modèle de négociation réaliste dans les

systèmes multi-agents," 2012, August.

[34] D. K. Adina Florea, Stefan Pentiuc,. (2002). *Architectures des agents et langages*. Available: http://turing.cs.pub.ro/auf2/html/chapters/chapter2/chapter_2_2_2.html

[35] T. Garneau and S. Delisle, "Programmation orientée-agent: évaluation comparative d'outils et environnements," in *JFSMA*, 2002, pp. 111-124.

[36] Jacques Ferber, *Madkit pas à pas: Démarrage et prise en main du logiciel Madkit*. 2009.

[37] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "JADE—a java agent development framework," in *Multi-Agent Programming*: Springer, 2005, pp. 125-147.

[38] T. Moyaux, B. Chaib-draa, and S. D'Amours, "Satisfaction distribuée de constraintes et son applicationa la génération d'un emploi du temps d'employés," *5ème Congères International de Génie Industriel (CIGI03), Quebec,* 2003.

[39] A. Kouider, S. Ourari, B. Bouzouia, and M. Mihoubi, "APPROCHE MULTI-AGENTS POUR L'ORDONNANCEMENT DYNAMIQUE D'ATELIER DE PRODUCTION," in *9th International Conference on Modeling, Optimization & SIMulation*, 2012.

[40] T. Coudert, B. Grabot, and B. Archimède, "Logique floue et système multiagents pour un ordonnancement coopératif production/maintenance," in *3e Conférence Francophone de Modélisation et de SIMulation MOSIM*, 2001, vol. 1, pp. 485-490.

[41] M. Touat, S. Bouzidi-Hassini, and F. Benbouzid-Sitayeb, "UNE APPROCHE PROACTIVE/REACTIVE POUR L'ORDONNANCEMENT CONJOINT DE PRODUCTION ET DE MAINTENANCE SOUS CONTRAINTES DE RESSOURCES HUMAINES DANS UN ATELIER A UNE MACHINE," in *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*, 2014.

[42] M. H. Hentous and E. Docteur, "Approche multi-agents pour l'ordonnancement par coopération multi-ressources avec délais de production."

[43] Telecom Italia. *JAVA Agent DEvelopment Framework*

is an open source platform for peer-to-peer agent based applications. Available: http://jade.tilab.com/

[44] C. Giovanni, "JADE Tutorial–JADE Programming for Beginners," *TILab SpA,* 2003.

[45] D. K. Adina Florea, Stefan Pentiuc,. (2002). *Communication dans les Systèmes Multi-Agents*. Available: http://turing.cs.pub.ro/auf2/html/chapters/chapter4/chapter_4_5_1.html

Système multi agents pour la planification et l'ordonnancement des opérations chirurgicales avec prise en compte des aléas

Résumé

Dans ce travail nous nous focalisons sur le problème de planification et d'ordonnancement des opérations chirurgicales qui comptent parmi les tâches critiques et les plus importantes dans la gestion du bloc opératoire, ils consistent à affecter les patients aux salles d'opération et à définir le séquencement des interventions chirurgicales à effectuer dans les salles d'opération.

L'approche proposée consiste en deux phases : la phase proactive qui permet d'établir le planning de la semaine à l'aide de la méta heuristique TS en visant à maximiser l'occupation des ORs et minimiser les coûts liés à la sur utilisation de celles-ci sous certaines contraintes. Et la phase réactive permettant d'adapter le planning aux imprévus quotidiens en utilisant un système multi agents permettant la communication entre agents afin de déclarer l'indisponibilité d'un chirurgien, de le remplacer ou de changer le planning si nécessaire.

***Mots clés :** Planification et ordonnancement, Multi-agents, Salle d'opération, Recherche Tabou*

Multi agents system for planning and scheduling the surgical operations under uncertainties

Abstract

In this work, we focus on the planning and scheduling problem of the surgical operations which are among the critical and most impacting tasks which consist in assigning patients to ORs and in defining the sequence of surgical interventions to be performed in the ORs, respectively.

The proposed method consists in two phases: the proactive phase which consists in establishing the planning of the week that aims to maximize the ORs occupation and minimize the overtime related costs under certain constraints using TS metaheuristic. And the reactive phase consists in adapting the planning to uncertainties using multi-agents system that allows the communication between agents in order to declare the unavailability of the surgeon, to replace him or to change the planning if it's necessary.

***Key words :** Planning and scheduling, Multi-agents, Operating room, Taboo search.*

Master Systèmes Intelligents & Réseaux
Département d'Informatique
Faculté des Sciences et Techniques de Fès

A.U. 2017 - 2018