F.S.D..... N° D'ordre :

Ministère de l'Enseignement Supérieur et de le Recherche Scientifique

Université Saad Dahlab de Blida

Faculté des sciences

Département d'informatique

Mémoire Présenté par

Bouchenafa Anis
Lounes Ferhat

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique. MI
Filière : Informatique.
Option : Ingénierie des logiciels.

Titre

Conception et réalisation d'un site web dédié aux enchères combinatoires

Promoteur : Mr. Sidoumou

Soutenue le: 26-09-2011 **devant le jury composé de :**

-Mme OUAHRANI	**Présidente**
-MR ZAIR MUSTAFA	**Examinateur**
-Mlle ELGHERS	**Examinateur**

- promotion 2010/2011-

Dédicace

A CELUI QUI M'A INDIQUE LA BONNE VOIE EN ME RAPPELANT QUE LA VOLONTE FAIT TOUJOURS LES GRANDS HOMMES...Mon père

A CELLE QUI A ATTENDU AVEC PATIENCE LES FRUITS DE SA BONNE EDUCATION, Ma mère

Et pour toutes les peines endurées, toutes les privations et sacrifices consentis, pour faire de moi un Homme.

A mes tendres sœurs et frères.

A mes ami(e)s.

A mes collègues.

Je dédie ce travail

Bouchenafa Anis

DEDICACE

Je dédie ce mémoire à mes très chers parents qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance. J'espère qu'ils trouveront dans ce travail toute ma reconnaissance et tout mon amour.

Je dédis ce modeste travail à mes chères sœurs et Frères que j'adore.

Je tiens aussi à saluer toutes les personnes qui m'ont encouragé, de prés ou de loin

A Mes Ami(e)s, ainsi que tous mes camarades de la promotion.

Lounes Ferhat

بسم الله الرحمان الرحيم

الحمد و الشكر لك يا الله

Remerciements

Nous exprimons toute notre gratitude à notre enseignant et promoteur Mr **Sidoumou Med Rédha** pour tous le temps qui nous à consacré ainsi que pour tous ses conseils et précieuses recommandations.

On est de même reconnaissante à notre chef de département Mr. **Mohamed Massied** pour l'environnement de travail et le matérielle qui à été mis à notre disposition.

Et bien évidement on tient a remercié infiniment Melle **Boustia**, Melle **Feghoul Atef** pour leur aides et soutiens.

Mes remerciements vont aussi à **Harma Dalila** qui m a soutenue et encouragée tout au long de ce travail ..."

Sans oublier les membres du jury qui ont bien voulu nous faire l'honneur d'examiner notre travail.

Enfin nous remercions tous les enseignants de notre département qui ont participé à notre formation Master. Ainsi que pour leur aides et soutiens.

Résumé :

Les enchères combinatoires sont une généralisation des enchères traditionnelles, dans lesquelles les différents articles sont vendus, et un soumissionnaire est autorisé à soumissionner pour n'importe quelle combinaison d'objets à la fois. Cela est essentiel si un soumissionnaire considère l'utilité d'un ensemble d'éléments à être différent de la simple somme des utilités des éléments considérés séparément. Avec la vulgarisation de l'internet, de nouvelles opportunités s'ouvrent, permettant de toucher un plus grand nombre de clients potentiels. Néanmoins, le déploiement d'un site dédié aux enchères combinatoires n'est pas chose aisée du fait de la complexité algorithmique de la détermination la meilleure combinaison d'offres.

Nous développons dans ce travail un site web pour les enchères combinatoires s'appuyant sur un algorithme de résolution exact qui sera également implémenté.

Mots Clés : Enchères combinatoires, problème de détermination du gagnant (WDP), technologie J2EE.

Chapitre 1 :
Introduction générale

1. Introduction :

Les enchères sur Internet obéissent à la même logique de centralisation que dans le modèle historique : il s'agit de réunir un maximum de vendeurs et d'acheteurs potentiels en assurant la meilleure liquidité du marché. Les ventes aux enchères traditionnelles sont limitées dans leur efficacité par une contrainte physique : la vente a lieu à un endroit et à un horaire précis. Cette contrainte physique limite naturellement le nombre de ventes d'acheteurs potentiels, même s'il est souvent possible d'enchérir par téléphone ou par écrit en avance. Sur Internet en revanche, l'information est beaucoup plus facilement disponible, et les enchères peuvent parfaitement courir sur plusieurs jours et intéresser potentiellement des acheteurs partout dans le monde

Cette liquidité attire à leur tour les vendeurs. Un cercle vertueux est donc enclenché. Les acheteurs sont attirés par la variété des biens mis en vente, la perspective d'une bonne affaire, la commodité liée à l'Internet (faire des achats sans sortir de chez soi), et dans une bonne mesure par l'amusement de trouver facilement des biens rares, introuvables dans des canaux traditionnels. Les vendeurs trouvent un accès aisé à un marché très large, des coûts de distribution et de marketing très compétitifs, la possibilité de maximiser les prix et d'augmenter leurs ventes sans coût d'entrée ni de sortie autre que négligeable. Certaines entreprises traditionnelles utilisent ainsi les sites d'enchères en ligne comme un canal de distribution complémentaire, de nombreux petits commerces spécialisés mais aussi des grandes entreprises comme IBM qui y écoulent leurs fins de série.

Notre projet de fin d'étude s'inscrit dans ce domaine, il consiste à la conception et la réalisation d'un site web pour les enchères combinatoire.

2. **Problématique** :

Les enchères combinatoires permettent de vendre des collections d'objets à des acquéreurs dont les préférences peuvent être complexes. Comment concevoir un mécanisme d'allocation et de détermination des prix de faible complexité algorithmique et permettant de satisfaire au mieux les acquéreurs ?

Une vente aux enchères combinatoire est une vente aux enchères, dans lequel les différents articles sont vendus, et un soumissionnaire est autorisé à soumissionner pour n'importe quelle combinaison d'objets à la fois. Cela est essentiel si un soumissionnaire considère l'utilité d'un ensemble d'éléments à être différent de la simple somme des utilités des éléments considérés séparément.

Dans une enchère combinatoire à une seule unité le vendeur (commissaire priseur) est confronté à une série d'offres avec des prix différents, et son but est de sélectionner les plus rentables sous-ensembles d'entre eux de sorte que pour toutes les offres ne partagent aucun article.

Dans notre travail nous allons présentée les enchères combinatoires et leurs problème de détermination du gagnant.

Le concept des enchères combinatoires sera détaillé est expliqué dans le chapitre suivant.

3- **Objectif** :

Résoudre des problèmes difficiles d'optimisation combinatoire à l'optimalité est souvent un travail immense nécessitant des algorithmes très efficaces.

Notre objectif est l'implémentation d'un algorithme exact pour la résolution du problème d'enchères combinatoire.

Un algorithme par séparation et évaluation est de loin l'outil le plus largement utilisé pour résoudre à grande échelle les problèmes Complet d'optimisation combinatoire.

Cet algorithme est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète, toutefois C'est une méthode d'énumération implicite où toutes les solutions possibles du problème peuvent être énumérées. L'énumération explicite est normalement impossible à cause du nombre croissant exponentiellement de solutions potentielles. L'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions. L'utilisant des bornes pour la fonction à optimiser combinée avec la valeur de la meilleure solution trouvé actuellement permet à l'algorithme de rechercher dans des parties de l'espace de solution de manière implicite. Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées. Cet Algorithme sera présenté dans le chapitre suivant.

4- **Organisation du mémoire** :

Notre travail sera organisé en chapitres.

« **Chapitre I** » Introduction générale, où nous allons introduire notre projet, posé la problématique et définir nos objectifs.

« **Chapitre II** » Etat de l'art, où sera la présentation du concept d'enchère, la définition des enchères combinatoires et la présentation de l'algorithme à implémenté.

« **Chapitre III** » Conception, dans ce chapitre nous commencerons par la présentation du langage UML, nous passerons a l'analyse des besoins avec les diagrammes de cas d'utilisation, après ça sera suivit par le diagramme des classe et les diagrammes de séquences.

« **Chapitre IV** » Implémentation, c'est la mise en œuvre du site web et l'implémentation de l'algorithme.

« **Chapitre V** » Examens et tests, nous allons voir l'espace mémoire que va prendre l'algorithme ainsi que sa complexités, et terminé par des tests.

Chapitre 2 :
Etat de l'art

1. Définition des Enchères:

L'enchère est définie par le *Dictionnaire de l'académie française* comme une « offre d'un prix supérieur à la mise à prix, ou au prix qu'un autre a déjà offert, en parlant des choses qui se vendent ou s'afferment au plus offrant. »

Une mise aux enchères est une déclaration de mise en concurrence pour un objet ou un service précis pour un prix ou un contrat que l'on se propose de remplir et qui doit être *meilleure* (ou égale au minimum pour la première) que la précédente avec un incrément minimum.

Procédure : La procédure est effectuée par un tiers qui est par exemple un commissaire-priseur ou un site internet spécialisé. Toutes les enchères sont triées prioritairement par prix (l'offre la plus élevée arrive en premier), puis par quantité, puis par heure de placement, ce qui détermine l'adjudication.

S'il y a un prix de réserve qui n'est pas atteint alors la procédure est annulée. Très souvent un délai supplémentaire est accordé pour faire une surenchère.

Il y a des enchères spéciales comme par exemple lorsque les prix suivent un changement vers le moindre et l'adjudicataire (donc unique) est celui qui accepte la transaction aux conditions de ce moment. **[Wikipedia]**

2. Les types d'enchères:

Il existe désormais une très grande diversité de produits « enchère » disponibles. On peut aujourd'hui en recenser plus d'une vingtaine, différents. Interprété dans son sens large, le terme « enchère » désigne tout mécanisme structuré de concurrence visant à déterminer qui obtient l'article en jeu, et souvent à quel prix en amenant les acheteurs à révéler, directement ou par offre successives, leur prix de réservation pour cet article. On parle d'un couple règle d'allocation - règle de paiement. Si l'on inclut les mécanismes d'enchères doubles (où les deux côtés du marché sont en concurrence), tous les marchés boursiers deviennent du coup désignés par ce terme. Selon les stratégies des postulants et la nature de l'information qu'il s'agit de révéler, les différents formats d'enchère ont des performances variables. **[Wikipedia]**

Cependant, il n'existe théoriquement que 4 formats d'enchères différant sur le fond :

2.1. Enchères sous pli cacheté (au premier prix) :

Aussi appelées enchères scellés au premier prix. Chaque enchérisseur remet une enchère (offre) indépendamment des autres sous enveloppe ou électroniquement au commissaire-priseur qui examine toutes les offres. L'objet est attribué au plus offrant, qui paie son montant proposé. C'est un processus « statique » puisqu'il ne comporte qu'un tour. De plus, une des caractéristiques de ce type d'enchère est que l'enchérisseur ne reçoit aucun signal (offre) de la part des autres enchérisseurs. C'est le procédé classique utilisé lors des appels d'offre pour les Marchés publics entre autres ou encore pour les droits minéraux et de forage dans les terrains de l'État.

3. Problèmes d'optimisation combinatoire:

I. Introduction :

Les problèmes d'optimisation apparaissent dans d'innombrables domaines d'ingénierie, d'industrie ou encore économiques. L'optimisation, en termes générales, survient dans la minimisation du temps d'un processus, du coût où des risques, ainsi que dans la maximisation des profits, de l'efficacité ou de la qualité [TAL09]. Par exemple, il existe d'innombrables manières d'organiser l'exécution de tâches lors d'un processus de production avec des coûts en temps variables, il s'agira alors de trouver la configuration minimisant le temps de production. Les problèmes d'optimisation varient par leur complexité, c'est pourquoi il existe plusieurs approches de résolution.

Problèmes d'optimisation [TAL09]:

Un problème d'optimisation peut être défini par un couple (X, f), où X est l'ensemble des solutions réalisables, et $f: X \longrightarrow \mathbb{R}$, dite *fonction objectif,* est la fonction devant être optimisée. La fonction objectif assigne à toute solution $s \in X$ un réel indiquant sa valeur. La fonction objectif définit donc une relation d'ordre sur X.

La solution $s^* \in X$ telle que $\forall s \in X: f(s^*) \leq f(s)$ est appelée (cas de minimisation) ***optimum global*** (Pour les cas de maximisation il nous faut remplacer $\leq$ par $\geq$). Ainsi, le but principal de la résolution d'un problème d'optimisation est de trouver un optimal global (ou une ensemble d'optimums locaux).

Dans la pratique, différents modèles d'optimisation sont utilisés pour formuler, et ainsi résoudre, des problèmes de prise de décision. Les modèles les plus fructueux sont basés sur la programmation mathématique. Un modèle couramment utilisé dans la programmation mathématique est connu sous le nom de *programmation linéaire (**LP** : **L**inear **P**rogramming).* Ce modèle peut s'exprimer de la façon suivante :

$$f(x) = Min\ c.x$$

Soumis à :

$$A.x \geq b$$
$$x \geq 0$$

Où x est un vecteur de variables de décision continues caractérisant la solution, c et b (resp. A) sont des vecteurs (resp. matrice) de coefficients constants.
Dans un problème d'optimisation en programmation linéaire, tout de la fonction à optimiser f et des fonctions de contraintes $A.x \geq b$ sont linéaires.

Exemple (programmation linéaire pour la prise de décision): Une compagnie synthétise deux produits $Prod_1$ et $Prod_2$ à partir de deux types de matière première M_1 et M_2 disponibles en des quantités b_1, b_2. Chaque unité du produit i nécessite pour sa production A_{ij} unité de la matière j. L'objectif est de trouver la quantité x_1 (resp. x_2) de $Prod_1$ (resp. $Prod_2$) à produire afin de maximiser les profits sachant le profit généré par chacun des produits.

Les données du problème sont données dans le tableau suivant :

	Utilisation pour $Prod_1$	Utilisation pour $Prod_2$	Disponibilité
M_1	**6 (A_{11})**	**4 (A_{12})**	**24 (b_1)**
M_2	**1 (A_{21})**	**2 (A_{12})**	**6 (b_2)**
Profit par unité	**5 (c_1)**	**4 (c_2)**	

Figure S2.I.4.2.1 Tableau des données associées à la production (exemple programmation linéaire)

Le modèle de ce problème peut être formulé en un programme linéaire :

$$Max\ Profit = 5x_1 + 4x_2$$

Soumis aux contraintes :

$$6x_1 + 4x_2 \leq 24$$
$$x_1 + 2x_2 \leq 6$$
$$x_1, x_2 \geq 0$$

La figure II.2 illustre l'interprétation graphique du modèle. Chaque contrainte est représentée par une droite. La fonction objectif est une infinité de droites parallèles. L'objectif max est celui dont la droite croise le point extrême de l'espace des solutions. On déduit du graphe que la solution optimale est ($x_1 = 3, x_2 = 1{,}5$). Le profit correspondant est de 21.

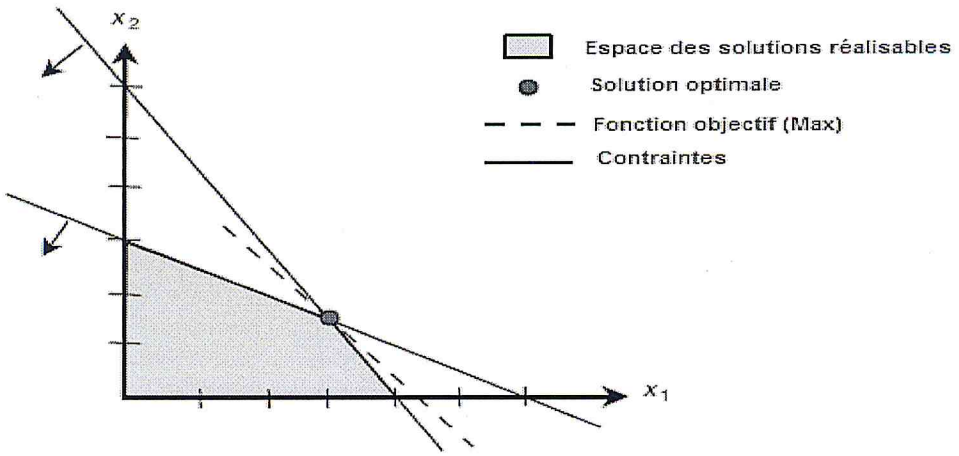

Figure S2.I.4.2.2 Représentation graphique du problème (PL) et sa résolution [TAL09]

- **Algorithme du simplexe :**

La méthode du simplexe est une procédure standard pour résoudre des programmes linéaires, c'est une méthode itérative qui améliore la solution à chaque étape.
En regardant la représentation graphique d'un programme linéaire à 2 variables, la méthode du simplexe recherches à travers les côtés du polygone.

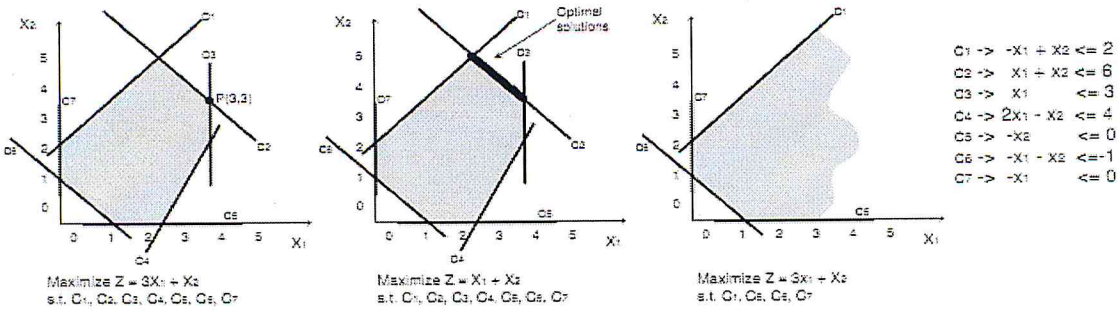

Figure S2.I.3 Représentation graphique d'une solution unique, solution multiple et une dégénérescence

La Figure S2.I.3 donne une représentation graphique d'une solution unique (à gauche), une solution multiple (au centre) et une dégénérescence (à droite).

Comme le nombre de sommet est fini, on peut toujours trouver la solution.
En bref, les étapes de la version tabulaire de cette méthode sont les suivants :

Etape 1. Transformer les inégalités en égalités en introduisant se qu'on appelle les variables d'écart. En conséquence, un système linéaire des équations est obtenu.

Etape 2. Réglez la fonction objective à zéro.

Étape 3. Ecrire le tableau simplexe initial.

Dans le tableau simplex, il y a une colonne pour chaque variable du problème, y compris les variables d'écart et la dernière colonne pour les valeurs de la solution. Ensuite, le processus est d'ajouter les coefficients des inégalités dans chaque rangée et les coefficients de la fonction objectif dans la dernière rangée.

Étape 4. Trouvez la variable que l'on appelle entrante et la faire sortir.

A. Pour choisir la variable d'entrer, l'algorithme regarde la dernière rangée, et choisit la variable avec le plus grand coefficient négatif (en valeur absolue). Cette colonne sera appelé la colonne du pivot.
B. Pour trouver la variable à faire sortir, le processus a besoin de diviser chaque élément de la dernière colonne par l'élément correspondant à la colonne du pivot étant donné que l'élément est supérieur à zéro. Si tous les éléments sont inférieurs à zéro, puis la solution est trouvée.

La ligne avec le résultat de la division avec la plus petite valeur positive indique le rang de la variable à faire sortir. Cette ligne est appelée pivot. Dans l'intersection de la ligne pivot et la colonne du pivot le pivot opérationnel est trouvé.

Étape 5. Pour trouver le coefficient de la nouvelle table de l'algorithme on obtient les coefficients des nouveaux X et cela en divisant les coefficients de la ligne pivot par le pivot opérationnel.

Puis, avec la réduction gaussienne l'algorithme transforme à zéro le reste des éléments de la colonne. De cette façon, les méthodes simplex obtiennent les nouveaux coefficients des autres lignes, y compris la fonction objective.

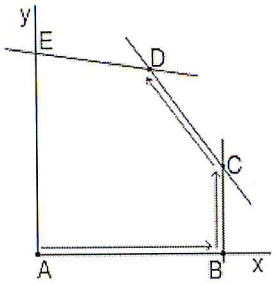

Figure S2.I.4 Recherche simplex

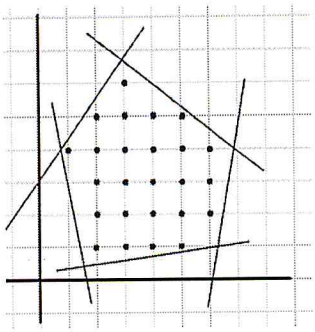

Figure S2.I.5 Solution réalisables dans un programme en nombres entier

Le processus répète les étapes 4 et 5 et s'arrête lorsque tous les coefficients de la fonction objective sont positifs.
Dans ce cas, la solution optimale est trouvée. Les valeurs de la solution sont les coefficients de la dernière colonne.
La figure 2.4 montre la représentation graphique du processus. La méthode simplex recherches à travers le sommet du polygone décrit par les contraintes. Notez que la recherche avec la méthode du simplexe commence avec une solution réalisable.
C'est la raison pour laquelle cette méthode est aussi appelée la méthode en deux phases. Dans la première phase une solution réalisable de base est fournie où il est démontré qu'aucune solution faisable existe. Puis, à partir de cette solution de base, le problème est résolu dans la deuxième phase.

Variantes des problèmes linéaires :

On distingue trois variantes pour les problèmes linéaires, problèmes linéaires continue, discret ou mixte. La théorie de l'optimisation continue est, en termes d'algorithmes d'optimisation, plus aboutie que celle de l'optimisation discrète avec l'existence de méthodes efficaces tel le simplexe. De multiples applications de la vie réelle doivent être modélisées par des variables discrètes, rendant les modèles continus inappropriés. En effet, dans de nombreux problèmes d'optimisation pratiques, les ressources sont indivisibles (machines, travailleurs...). Dans la programmation linéaire en variables entières (**ILP** : **I**nteger **L**inear **P**rogramming) les variables de décision sont discrètes. Lorsque les variables de décision sont des deux types (continu et discret), on parle de programmation linéaire en variables mixtes (**MLP** : **M**ixt **L**inear **P**rogramming). La programmation mixte généralise donc l'ILP et la LP.

La classe des problèmes d'optimisation combinatoire englobe celle des ILP. Les problèmes de cette classe se caractérisent par des variables de décision discrètes ainsi que par un ensemble de solutions réalisables fini. Cependant, la fonction objectif et les contraintes peuvent prendre n'importe quelle forme.

La programmation linéaire est l'un des modèles les plus satisfaisants pour la résolution des problèmes d'optimisation linéaires continues. En effet, pour ces problèmes d'optimisation des algorithmes exacts efficaces, telles que la méthode du simplex, existent. Nous en donnons dans ce qui suit une brève présentation.

Optimisation combinatoire :

III.1 Problème d'optimisation combinatoire :

Un problème d'optimisation combinatoire (POC) est définit par un ensemble d'instances. A chaque instance correspond un ensemble **discret** S de solutions (ou de configurations), un ensemble de solutions réalisable ou admissibles $X \subseteq S$ et une fonction objectif $f: X \mapsto \mathbb{R}$ [HAOxx]. Résoudre une instance d'un POC revient à rechercher parmi les solutions réalisables celle qui optimise le coût de la fonction objectif.

III.2 Exemple (problème du sac à dos) [SOL07]:

Le problème du Sac à Dos est un problème générique de l'informatique théorique au même titre que le voyageur de commerce (TSP). Le problème du Sac à Dos aussi noté KP (Knapsack Problem en anglais), représente une situation dans laquelle une personne dispose de n objets (d'utilité $u_i \in \mathbb{N}\backslash\{0\}$ et d'un poids $p_i \in \mathbb{N}\backslash\{0\}$ pour $i \in \{1, \dots, n\}$) et d'un sac ayant une capacité maximale $c \in \mathbb{N}$. Le problème est de trouver un ensemble d'objets dont la somme des utilités est maximale. Il faut donc trouver un sous-ensemble d'objets Y tel que $\sum_{y \in Y} p_y \leq c$ et qui maximise $\sum_{y \in Y} u_y$.

Le problème du sac à dos peut se résumer donc ainsi :

- *Problème (données):*
  Deux ensembles de n entiers $\{u_i / u_i > 0, i = 1, \dots, n\}$ et $\{p_i / p_i > 0, i = 1, \dots, n\}$ et un entier $c > 0$.
- *Espace de Solutions :* $S = \mathcal{P}(\{1, \dots, n\})$ *(ensemble des parties de* $\{1, \dots, n\}$*).*
- *Espace de Solutions réalisables :* $X \subseteq S / \forall Y \in X : \sum_{y \in Y} p_y \leq c$.
- *Objectif :* Trouver un sous-ensemble Y de $\{1, \dots, n\}$ $(Y \in X)$ tel que $\sum_{y \in Y} p_y \leq c$ qui maximise : $f = \sum_{y \in Y} u_y$.

Illustration :

Pour illustrer le problème nous allons prendre la situation suivante. Un campeur, dont le sac a pour capacité (poids maximum) $c = 20$, doit choisir parmi ces objets :

objet	poids	utilité
tente	11	20
gourde	7	10
duvet	5	25
vêtement	5	11
réchaud	4	5
nourriture	3	50
pharmacie	3	15
anti-moustique	2	12
lampe	2	6
cape	2	5
gamelle	2	4
couteau	1	30

Figure S2.I.6 Tableau Objet-(poids, utilité) (exemple KP)

Une méthode intuitive pour essayer de résoudre le KP serait de prendre les objets ayant la plus grande utilité. Le campeur prendrait alors pour cette instance du problème la

nourriture, le couteau, le duvet et la tente ce qui représente un poids de 20 pour une utilité de 125.

Une autre méthode intuitive consisterait à prendre les objets les moins lourds, le campeur prendrait alors le couteau, l'anti-moustique, la lampe, la cape, la gamelle, la nourriture, la pharmacie et le réchaud ce qui représente un poids de 19 pour une utilité de 127.

Une dernière méthode un peu moins naïve consisterait à prendre les objets dont le rapport utilité/poids est le plus important. Dans ce cas le campeur prendrait le couteau, la nourriture, l'anti-moustique, le duvet, la pharmacie, la lampe et la cape ce qui fait un poids de 18 pour une utilité de 143.

On a donc 3 résultats différents ce qui montre que le problème n'est pas si facile que ce qu'il semble. De plus aucune de ces méthodes ne donne la solution optimale qui a un poids total de 20 et une utilité totale de 147 (calculé par programmation dynamique).

III.3 Complexité des POC :

Schématiquement, on dit qu'un algorithme est efficace (ou qu'il est «bon»), si le nombre des opérations nécessaires pour résoudre le problème est borné par une fonction polynomiale d'un paramètre caractérisant la taille du problème [HAD02]. Un problème auquel un tel algorithme de résolution existe est dit *polynomial* [HAOxx]. La majorité des problèmes d'optimisation combinatoire ne sont pas polynomiaux, ils appartiennent à la classe des problèmes ***NP-difficiles***. Garantir l'optimalité des solutions trouvées pour ce genre de problèmes implique des méthodes exactes dont le temps de calcul croît exponentiellement par rapport à la taille des instances. Les méthodes approchées ne garantissent pas l'optimalité de la solution retournée, mais tendent à trouver un compromis en sacrifiant quelque peu de la « qualité » de la solution en faveur d'un temps de calcul raisonnable.

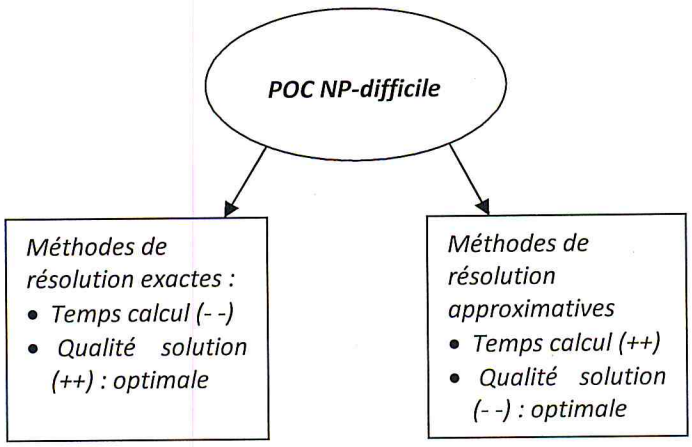

Figure S2.I.7 Caractéristiques des méthodes de résolution

III.4 Méthodes de résolution :

Sommairement, on identifie deux approches différentes dans la résolution de problèmes d'optimisation combinatoire : les méthodes exactes et les méthodes approximatives. Les méthodes exactes garantissent de trouver une solution optimale pour une instance d'un problème, néanmoins, pour des problèmes NP-Difficiles de grande taille le temps de calcul peut atteindre des mois voir des années. Les méthodes approximatives quant à elles, trouvent de « bonnes » solutions en un temps réduit mais ont le lourd inconvénient de ne pas garantir l'optimalité des solutions. Dans la *figure I.2* nous donnons une classification des méthodes d'optimisation plus complète basée sur la taxonomie de Talbi [TAL09] :

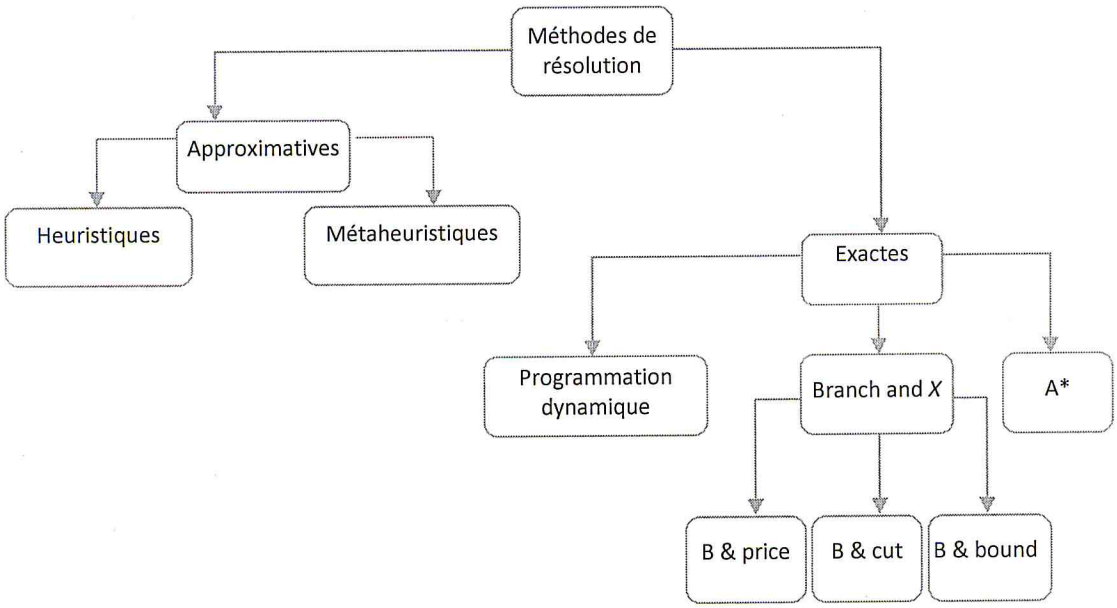

Figure S2.I.8 Taxonomie des méthodes de résolution

II.4.1 Les méthodes approximatives:

Une *heuristique* est une méthode approximative conçue pour résoudre un problème ou une instance spécifique. Pour reprendre l'exemple du problème du sac à dos (point II.2), la méthode qui consiste à prendre les éléments selon l'ordre décroissant des utilités, est une méthode heuristique. Une heuristique est donc une façon de procéder, guidée par une logique déduite du problème à traiter, et ne pouvant être appliquée à d'autres problèmes.

A l'inverse, les métaheuristiques se proposent, sans une connaissance préalable, de s'adapter à n'importe quel problème d'optimisation. Ce sont des méthodes génériques et d'un niveau d'abstraction élevé. Les métaheuristiques reprennent donc des concepts généraux d'optimisation. Ce sont des méthodes généralement itératives, qui tentent d'améliorer à chaque itération des solutions passées en entrée avec des procédés spécifique à chaque métaheuristique. On distingue principalement deux types de métaheuristiques ; les métaheuristiques à solution unique et métaheuristiques à population de solutions.

- Les métaheuristiques à solution unique ne traitent à chaque itération qu'une seule solution. Ils se basent sur la notion de «*voisinage*» d'une solution. Le voisinage [DEV07] d'une solution s est un sous-ensemble de X, noté $V(s)$, contenant les solutions dites ***voisines*** directement atteignables à partir d'une transformation

donnée de s. Une même solution peut avoir plusieurs voisinages selon les modifications que l'on applique sur elle. Nous illustrons cette notion par un exemple simple : soit une espace de recherche S dont les éléments sont des vecteurs de bits de longueur fixe égale à 3. Soit $V_1(s)$ et $V_2(s)$ deux voisinages de la solution $s \in S$ générés par deux méthodes (transformations) différentes. $V_1(s)$ contient les solutions voisines définies comme étant les solutions différant de s par un seul bit, $V_2(s)$ est l'ensemble des voisins de s obtenus en permutant deux bits.

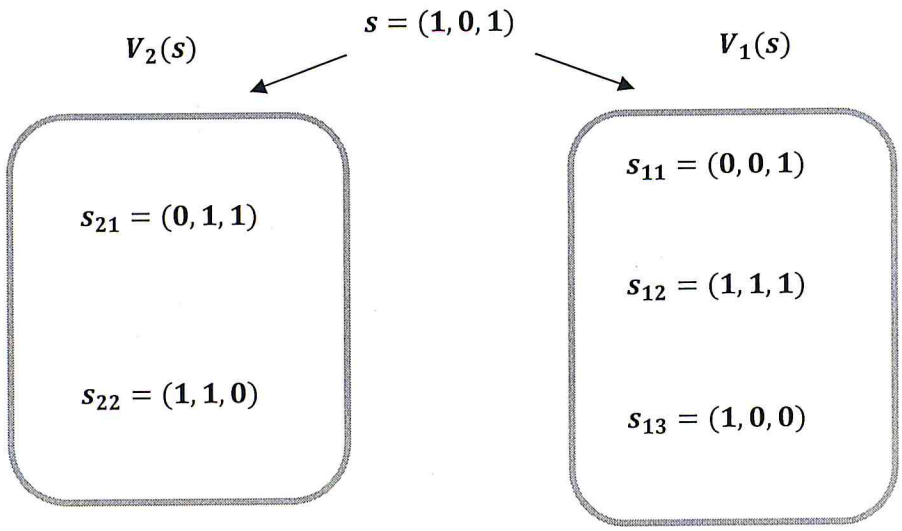

Figure S2.I.9 Deux voisinages pour une solution.

Les métaheuristiques à solution unique sont donc des méthodes itératives qui explorent à chaque étape le voisinage d'une solution (solution courante) de différentes manières (selon la méthode et ses paramètres). Nous donnons ci-dessous un exemple de métaheuristique à solution unique : *la Recherche locale (LS : Local Search)*.

Les méthodes de recherche locale, appelées également méthodes de descente, sont des métaheuristiques très utilisées pour résoudre la classe des problèmes d'optimisation combinatoire. Elles doivent leur succès à leur rapidité d'exécution et à leur simplicité, une rapidité due d'une part à sa mémoire réduite à la seule solution courante. Les méthodes de descente tirent leur nom du fait qu'à chaque pas elles progressent vers une solution voisine de qualité meilleure. La descente s'arrête lorsque tous les candidats (voisinage) sont moins bons que la solution courante ; c'est à dire lorsqu'un optimum (local à priori) est atteint [Pap82].

Algorithme I.1 Algorithme de descente générique	
1) Choix d'une solution initiale C_0 2) Solution courante $C \leftarrow C_0$; 3) Génération des candidats (voisins de C) 4) Choix du meilleur candidat S 5) Si $f(S) < f(C)$ alors	**Mémoire** • Solution courante

$C \leftarrow S$
aller en 3
Fin.

- Contrairement aux méthodes à solution unique les métaheuristiques à population traitent à chaque itération un ensemble de solutions qu'on appelle ***population.*** La majorité des méthodes à population sont bio-inspirées, ce qui signifie qu'elles reprennent le fonctionnement de la nature (reproduction, évolution, recherche de nourriture...etc.). Prenant l'exemple des ***Algorithmes génétiques*** (AG) qui reprennent le principe de la sélection naturelle Darwinienne.

Les origines des AG reviennent à John Holland qui dégage de son étude des processus d'adaptation naturelle une théorie de l'adaptation au milieu des années 1970 [Hol75]. Il s'appuie sur cette théorie pour construire des systèmes artificiels capables d'adaptation : c'est la naissance de l'algorithme génétique [HAD02]. Les AG usent des principes mais aussi de la terminologie du processus d'adaptation biologique. Ainsi, les solutions de la population sont appelés *individus (chromosomes ou encore phénotypes)* et les itérations sont des générations. Des opérateurs de *sélection*, *croisement* et *mutation* sont appliqués aux chromosomes dans le but d'en créer de nouveaux.

Le schéma général d'un algorithme génétique est de partir d'une population de solutions (généralement aléatoire). S'en suit une évaluation des individus de la population dont un sous-ensemble est sélectionné de façon à favoriser les meilleurs individus. Un ensemble d'enfants est ensuite construit à partir de la sous-population sélectionnée en utilisant des mécanismes de croissement et de mutation sur les individus. Ceci est répété jusqu'à atteindre un critère d'arrêt (nombre d'itérations par exemple).

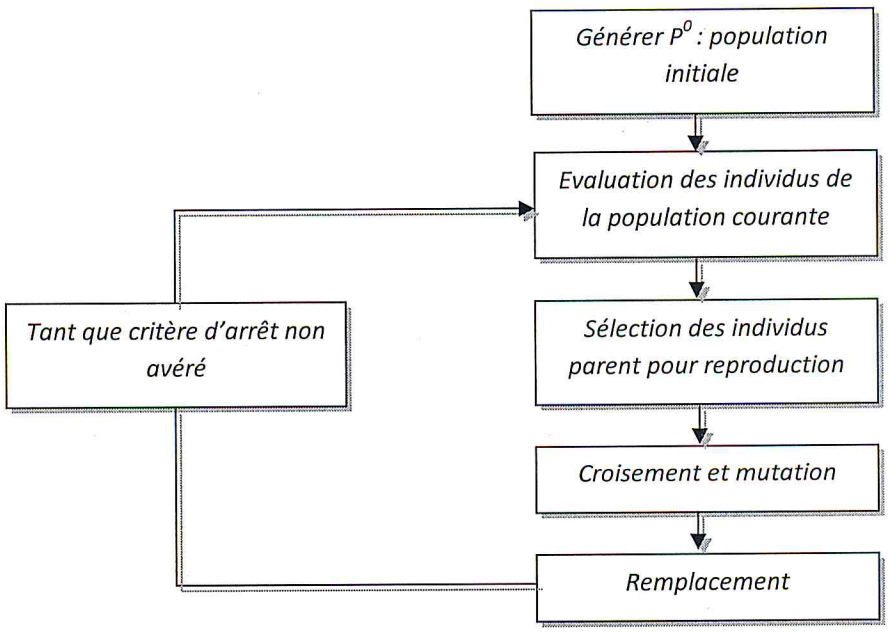

Figure S2.I.10 Diagramme de fonctionnement de l'AG standard

Dans l'algorithme génétique canonique [Tal95] les solutions sont des vecteurs de bits, cependant pour que les GA soit efficaces, le codage des solutions doit être adapté et spécifique au problème traité.

D'une manière générale, la sélection est un mécanisme stochastique. Une fonction d'évaluation, dénommée fonction *fitness*, donne le niveau d'adaptation d'un individu ; ce niveau est encore appelé la *fitness*. La sélection détermine la probabilité de reproduction d'un individu proportionnelle à sa *fitness*. Une fois les probabilités établies, les candidats à la reproduction sont tirés aléatoirement pour constituer la population des individus sélectionnés.

Le *croisement* ou « cross over » consiste en la recombinaison des caractéristiques génétiques de deux parents pour la production de deux enfants. Les chromosomes des deux parents sont découpés pour être croisés, ce qui permet de mélanger leurs caractéristiques.

La *mutation*, généralement appliqué après le croisement, affecte légèrement le phénotype des individus. Elle peut modifier chaque individu selon une probabilité qui est un paramètre du GA, souvent de l'ordre de 1%. Pour l'AGC, la mutation, quand elle s'applique, change la valeur d'un bit (*flip*), choisi au hasard dans le phénotype de l'individu [HADxx].

Le remplacement termine le cycle du GA produisant une nouvelle génération de parents. On distingue des mécanismes de remplacement générationnels comme pour L' AGC : la nouvelle génération occulte complètement les parents, Ou d'autres mécanismes de remplacement basés sur le maintien dans la population de certains parents. Pour les GA *k*-élitistes, la technique de remplacement préserve les *k* individus les plus performants. Pour la méthode stable (*steady-state*), seuls un ou deux individus sont remplacés à chaque génération.

II.4.2 Les méthodes exactes :

Les méthodes approximatives sont des méthodes intéressantes pour la résolution des problèmes d'optimisation combinatoire car elles permettent de trouver un compromis entre perte de qualité des solutions et amélioration du temps de calcul. Toutefois, certains domaines critiques tels les domaines financiers nous obligent à résoudre les problèmes de manière optimale, ce qui est du domaine des méthodes exactes.

Comme indiqué sur la figure I.7, on peut trouver dans la classe des algorithmes exacts les méthodes suivantes : la programmation dynamique, les Branch and X (Bound, Cut ou Price) ainsi que l'algorithme A^*. Ces méthodes peuvent être vues comme des techniques de recherche arborescentes. La recherche est effectuée sur l'ensemble de l'espace de recherche, le problème est résolu en étant subdivisé en des sous-problèmes simples.

La *programmation dynamique* est basée sur la division récursive du problème en des sous-problèmes. Cette stratégie se fonde sur le principe de Bellman qui suggère que les « sous-

politiques » d'une politique optimale sont elles-mêmes optimales [TAL09]. Cette méthode d'optimisation par étages est le résultat d'une séquence de décisions partielles. La procédure permet d'éviter une énumération totale de l'espace de recherche en supprimant des séquences de décisions partielles ne pouvant conduire à l'optimum.

L'algorithme du *Branch and Bound* (*séparation et évaluation*) et le A^* sont des méthodes basées sur une énumération implicite de toutes les solutions du problème d'optimisation. L'espace de recherche est exploré en construisant d'une manière dynamique un arbre dont la racine est le problème à résoudre, les nœuds internes les sous-problèmes et les feuilles les solutions potentielles. L'énumération est implicite car en réalité des branches entières de l'arbre de recherche sont identifiées comme ne pouvant conduire à l'optimum, et ainsi supprimées, grâce à une fonction d'estimation. Le *Branch and Bound* étant l'une des méthodes exactes les plus populaires pour la résolution des problèmes d'optimisation, nous la développons quelque peu dans ce qui suit.

Branch and Bound [Wik11]:

Soit un problème d'optimisation combinatoire dont l'espace de recherche est S et de fonction objectif f. L'objectif est de trouver une solution réalisable optimale $x \in S$. D'un point de vue purement existentiel, le problème est trivial : une telle solution x existe bien car l'ensemble S est fini. En revanche, l'approche effective du problème se confronte à deux difficultés. La première est qu'il n'existe pas forcément un algorithme simple pour énumérer les éléments de S. La seconde est que le nombre de solutions réalisables est très grand, ce qui signifie que le temps d'énumération de toutes les solutions est prohibitif (la complexité algorithmique est en général exponentielle).

Dans les méthodes par séparation et évaluation (Branch and Bound), la séparation permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que l'évaluation évite l'énumération systématique de toutes les solutions.

La phase de séparation consiste à diviser le problème en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables de telle sorte que tous ces ensembles forment un recouvrement (partition) de l'ensemble S. Ainsi, en résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation peut être appliqué de manière récursive à chacun des sous-ensembles de solutions obtenus, et ceci tant qu'il y a des ensembles contenant plusieurs solutions. Les ensembles de solutions (et leurs sous-problèmes associés) ainsi construits ont une hiérarchie naturelle en arbre, souvent appelée arbre de recherche ou arbre de décision.

L'évaluation d'un nœud de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions réalisables associées au nœud en question ou, au contraire, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème (typiquement, qu'il n'y a pas de solution optimale). Lorsqu'un tel nœud est identifié dans l'arbre de recherche, il est donc inutile d'effectuer la séparation de son espace de solutions. À un nœud donné, l'optimum du sous-problème peut être déterminé lorsque le sous-problème devient « suffisamment simple ». Par exemple,

lorsque l'ensemble des solutions réalisables devient un singleton, le problème est effectivement simple : l'optimum est l'unique élément de l'ensemble. Pour déterminer qu'un ensemble de solutions réalisables ne contient pas de solution optimale, la méthode la plus générale consiste à déterminer une borne inférieure pour le coût des solutions contenues dans l'ensemble. Si on arrive à trouver une borne supérieure de coût inférieure au coût de la meilleure solution trouvée jusqu'à présent, on a alors l'assurance que le sous-ensemble ne contient pas l'optimum.

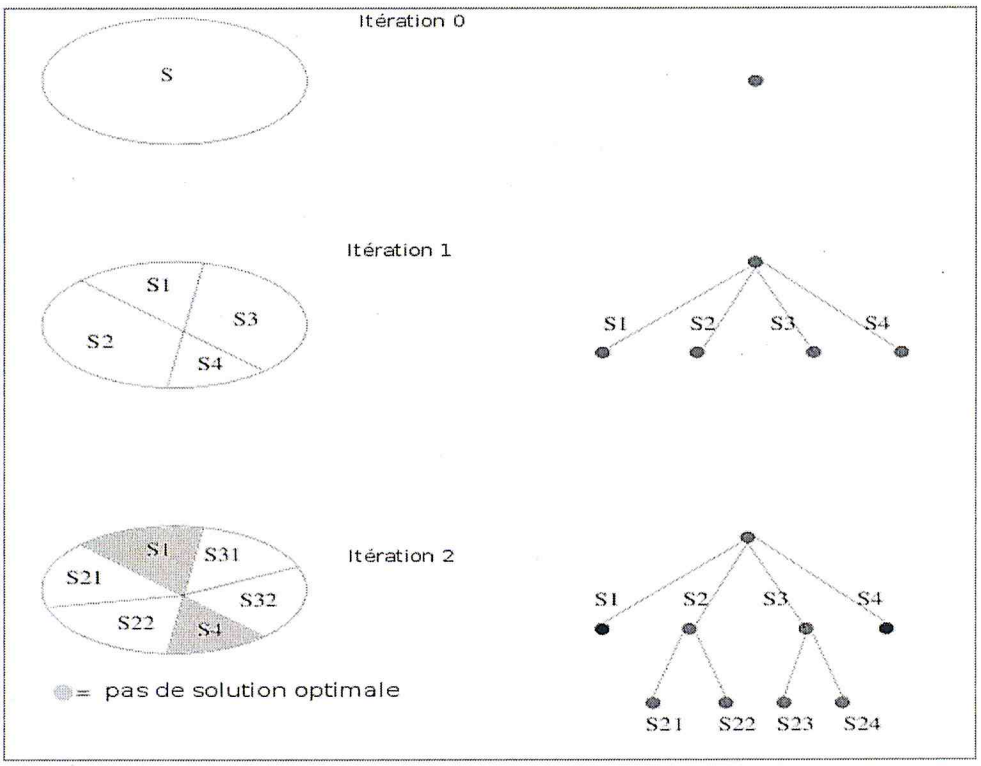

Figure S2.I.11 Schéma d'arborescence.

Conclusion:

Dans ce chapitre nous avons abordé les problèmes d'optimisation de manière générale et les problèmes d'optimisation combinatoire en particulier. L'une des méthodes de résolution les plus efficaces pour les problèmes linéaires (fonction objectif et contraintes linéaires) continu est la méthode du simplexe, cependant, de nombreux problèmes de la vie réelles ne sont pas continus et nécessitent d'autres méthodes. On identifie grossièrement deux familles de résolutions : méthodes approximatives et exactes. Il semble que l'une des méthodes exactes les plus utilisées soit l'algorithme de résolution par séparation et évaluation (Branch and bound).

Dans le prochain chapitre nous aborderons le problème d'optimisation combinatoire qui fait l'objet de notre sujet : le problème des enchères combinatoires.

4. les Enchères Combinatoires :

4.1. Introduction:

Une enchère est dite "*combinatoire*" lorsque les soumissionnaires peuvent placer une enchère sur une ensemble (ou combinaison) d'objets, appelé paquet, au lieu d'être restreint à un seul article. Dans une enchère combinatoire à une seule unité le vendeur (commissaire priseur) est confronté à une série d'offres avec des prix différents, et son but est de sélectionner les plus rentables sous-ensembles d'entre eux de sorte que pour toutes les offres ne partagent aucun article.
Les enchères combinatoires (**CA: C**ombinatorial **A**uctions) définissent un problème d'optimisation combinatoire traité par les chercheurs du domaine ainsi que de celui de la programmation mathématique. Elles présentent une foule de nouveaux défis par rapport aux enchères traditionnelles.

4.2. Enchères combinatoires :

a) Définitions et notations :

➢ **Définition 1 : (CA)**

Une enchère combinatoire (CA) est un couple d'ensembles finis (I, B) où:
$I = \{I_1, \dots, I_m\}$ est l'ensemble des articles mis aux enchères, et $B = \{B_1, \dots, B_n\}$ désigne l'ensemble de toutes les offres (« Bids ») dans l'enchère.

Chaque offre $B_i \in B$ est un couple (S_i, P_i) tel que $S_i \subseteq I$ est un ensemble d'articles de I et $P_i \geq 0$ est un nombre réel qui représente le prix offert par le soumissionnaire i pour S_i.

Notations:

- $S : B \to I$, est une fonction telle que : $S(B_i) = S(S_i, P_i) = S_i$.
  $P : B \to \mathbb{R}$, est une fonction telle que : $P(B_i) = P(S_i, P_i) = P_i$.
  En claire, la fonction S retourne l'ensemble des articles concernés par une offre, la fonction P retourne son prix (prix offert).

- Deux offres B_i, B_j sont dites *disjointes* si et seulement si $S(B_i) \cap S(B_J) = \emptyset$ i.e. les ensembles d'articles concernés par les deux offres ne contiennent aucun élément (article) en commun. On dit aussi que les offres sont *compatibles*.

➢ **Définition 2 : (Issue d'une CA)**

Une *issue* à une enchère combinatoire $\mathcal{A} = (I, B)$ est un ensemble $\boldsymbol{U} \subseteq B$ dans lequel toutes les offres sont deux à deux disjointes (deux à deux compatibles). En effet, un même article ne peut être vendu plus d'une fois.

Notons qu'il existe plusieurs issues à une enchère combinatoire. Notons par $\mathfrak{U}$ l'ensemble de toutes les issues.

➢ **Définition 3 : (Solution d'une CA et gain)**

Soit $\mathcal{A} = (I, B)$ une enchère combinatoire et $U \in \mathfrak{U}$ une issue de $\mathcal{A}$. Le *gain* P_u engendré par l'issue U est donné par : $P_u = \sum_{B_i \in U} P(B_i)$ (somme des prix de chaque offres constituant l'issue). Ainsi on définit une solution valide S pour $\mathcal{A}$ comme un couple (U, P_u).

b) <u>Problème de détermination du gagnant (WDP)</u>:

Les enchères combinatoires présentent une foule de nouveaux défis par rapport aux enchères traditionnelles. Certains de ces défis sont informatiques, et certains économiques. En ce qui concerne l'informatique le défi consiste à déterminer efficacement les meilleures offres une fois qu'elles ont été soumises au commissaire-priseur. Ce dernier devant choisir d'entre toutes les offres celles compatibles qui lui apporte le plus grand gain. Ceci est le *problème de la détermination de gagnant.*

Le problème de la détermination du gagnant (**WDP** : **W**inner **D**etermination **P**roblem) dans les enchères combinatoires est un problème d'optimisation combinatoire qui peut être énoncé comme suit :

> Soit $\mathcal{A} = (I, B)$ une enchère combinatoire. Le WDP de $\mathcal{A}$ est de trouver une solution $S = (W, P_W)$ telle que :
>
> - $W \subseteq B$.
> - $W \in \mathfrak{U}$ (W est une issue de $\mathcal{A}$).
> - $P_W = Max_{U \in \mathfrak{U}}\{P_u\}$.

Dans la pratique, le problème d'optimisation combinatoire dans la détermination des enchères gagnantes est d'étiqueter les offres avec des variables de décision x_i, $x_i = 1$si l'offre B_i est gagnante (offre choisie), 0 autrement (offre perdante) de façon à maximiser le revenu du commissaire-priseur, sous la contrainte que chaque article peut être attribué au plus à un seul

soumissionnaire puisqu'un article ne peut être vendu à deux personne en même temps. Plus formellement on obtient :

$$f = Max \sum_{i=1}^{n} x_i P_i \ (n : \text{nombre d'offres}, P_i : \text{prix de l'offre } B_i)$$

Sous les contraintes :
$\forall B_i, B_j \in B, B_i \ et \ B_j \ non \ compatibles: x_i + x_j \leq 1.$
$x_i, x_j \in \{0,1\}.$

c) Complexité du WDP et méthodes de résolution :

Un grand nombre de problèmes d'optimisation combinatoire dans le monde réel partage les propriétés suivantes: Ils sont des problèmes d'optimisation, sont faciles à énoncer, ont une taille fini, mais généralement ont un très grand nombre de solutions réalisables, en plus de ça il n'existe pas d'algorithmes efficaces pour leur résolution.

Tout comme le problème du sac à dos, Le problème de la détermination du gagnant WDP dans une vente aux enchères combinatoire ne déroge pas à cette règle. C'est un problème NP-complet, ce qui a été prouvé par **(Rothkopf et al. 1998, Karp, 1972)**. On retrouve dans la littérature de nombreuses tentatives afin de faire face à ce problème critique de la complexité des calculs. On distingue trois principales voix :

- *Conception d'algorithmes approximatifs* [FLY99]: cette approche ne permet malheureusement pas d'obtenir des approximations suffisantes en des temps de calcul polynomiaux [RPH98, San02].
- *Conception d'algorithmes exacts en temps polynomial pour des classes restreintes d'enchères combinatoires* [RPH98, San03]: l'inconvénient de cette approche est que l'on restreint les choix des soumissionnaires, et qu'ils ne peuvent ainsi pas exprimer complètement leurs préférences.
- *Conception d'algorithmes exacts, généralement rapides, mais qui requièrent un temps exponentiel dans le pire cas* : ces algorithmes se basent le plus souvent sur des approches de recherche arborescentes ainsi que sur des techniques de décomposition.

Il y a récemment eu plusieurs recherches dans l'élaboration d'algorithmes efficaces pour résoudre le problème de la détermination du gagnant « WDP » dans les enchères combinatoires. Mais en raison de la complexité algorithmique du problème, tout algorithme optimal sera lent sur certaines instances du problème. Citons quelque uns d'entre eux :

CASS [LEY03,FLY99] (**C**ombinatorial **A**uction **S**tructured **S**earch), **CABOB** [San05] (**C**ombinatorial **A**uction **B**ranch **O**n **B**ids), **CABRO** [MUxx] (**C**ombinatorial **A**uction **BR**anch and bound **O**ptimizer). Ces logiciels ont été conçus spécialement pour résoudre le problème des enchères combinatoires.

C1) Combinatorial Auction Structured Search (CASS) :

CASS est une solution basée sur l'algorithme Branch and Bound qui exploite l'information contextuelle d'une enchère. Pour ce faire, les offres (bids) sont divisées en des groupes appelés «bacs» (bins), de la manière suivante :
Pour chaque article de I un bac est créé. Etant donné un ordre sur les articles de l'enchère, les offres sont distribuées sur les différents bacs de manière à ce que toute offre soit mise dans le bac correspondant à son plus petit article. Ainsi, une offre ne se trouvera que dans un seul bac. Cette division permet à l'algorithme non seulement d'exclure le choix d'offres incompatibles (offres du même bac) mais aussi de négliger des bacs entiers.
CASS effectue une recherche en profondeur en utilisant une heuristique h pour le retour en arrière dans l'arbre de recherche. Il retient en mémoire la solution partielle $\mathcal{L}^{best}$ (ensemble d'offres compatibles) de plus grand gain (ou revenu) trouvée jusqu'alors. Chaque nœud interne du graphe de recherche correspond à une solution partielle (issue partielle) $\mathcal{L}$, pour laquelle l'algorithme calcule une borne supérieure $h(\mathcal{L})$ qui représente le revenu potentiel pouvant être obtenu avec les articles restant de l'enchère. L'algorithme opère un retour arrière quand $P_{\mathcal{L}} + h(\mathcal{L}) \leq P_{\mathcal{L}^{best}}$. De plus, si une issue U est complétée, CASS la retient en mémoire si $P_u > P_{\mathcal{L}^{best}}$ puis effectue un retour arrière. Nous renvoyons à [LEY03, FLY99] pour une description plus détaillée.

C2) Combinatorial Auction Branch On Bids (CABOB):

CABOB [San05] est un autre algorithme dédié au WDP. Il fait suite à une amélioration d'un autre algorithme du même auteur, ***BOB*** [San03] *(**Branch On Bids**)*. CABOB est un algorithme arborescent basé sur le Branch and Bound avec un parcours en profondeur. Durant la recherche, l'algorithme maintient une structure de graphe appelé « graphe d'offres » (bid graph) qui détient l'information sur les offres conflictuelles (offres incompatibles) : les sommets du graphe correspondent aux offres ne contenant aucun article déjà alloué (i.e. les offres non déjà incompatibles avec les articles alloués) ; deux sommets sont reliés si les offres correspondantes sont en concurrence pour un article. Pendant la recherche l'algorithme utilise des bornes inférieures et supérieures pour les gains que peuvent apporter les articles non alloués. Là aussi nous renvoyons à [San05, San06] pour de plus amples informations.

C3) Combinatorial Auction BRanch and bound Optimizer (CABRO):

CABRO est essentiellement un algorithme de recherche en profondeur Branch and Bound, contrairement aux algorithmes cités précédemment CABRO utilise plusieurs techniques de prétraitement afin de réduire la taille du problème. On donnera dans le chapitre suivant de plus amples détails sur l'algorithme celui-ci étant la principale source d'inspiration de notre implémentation. **[DAT08]**

5. Conclusion :

Dans ce chapitre nous avons défini les enchères de manière générale pour après cités quelque types d'enchère, suite à ça nous avons posé le problème d'optimisation d'une part ainsi que de l'optimisation combinatoire d'autre part. Après nous somme passé aux méthodes de résolutions approximatives et exactes. Pour finir nous avons parlé des enchères combinatoires ou on a expliqué le problème de détermination du gagnant, sa complexité et sa résolution.

2. Les diagrammes des cas d'utilisations :

Les diagrammes des cas d'utilisation identifient les fonctionnalités fournies par le système (cas d'utilisation), les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers. Les cas d'utilisation sont utilisés dans la phase d'analyse pour définir les besoins de "haut niveau" du système. Les objectifs principaux des diagrammes des cas d'utilisation sont:

- Fournir une vue de haut-niveau de ce que fait le système.
- Identifier les utilisateurs ("acteurs") du système.
- Déterminer des secteurs nécessitant des interfaces homme-machine. (IHM)

2.1. Les acteurs :

Un acteur est un utilisateur du système, et est représenté par une figure filaire. Le rôle de l'utilisateur est écrit sous l'icône. Les acteurs ne sont pas limités aux humains. Si le système communique avec une autre application, et effectue des entrées/sorties avec elle, alors cette application peut également être considérée comme un acteur.

Acteur	**Description**
Administrateur	Géré le site et les utilisateurs
Vendeur	Propose les articles à vendre
Acheteur	Fait des achats

Tableau 2.1: Tableau des Acteurs des Cas d'utilisation.

2.2.Les cas d'utilisation :

Un **cas d'utilisation** définit une manière d'utiliser le système et permet d'en décrire les exigences fonctionnelles. D'après Bittner et Spence, « Un cas d'utilisation, défini simplement, permet de décrire une séquence d'événements qui, pris tous ensemble, définissent un système faisant quelque chose d'utile »[1]. Chaque cas d'utilisation contient un ou plusieurs scénarios qui définissent comment le système devrait interagir avec les utilisateurs (appelés acteurs) pour atteindre un but ou une fonction spécifique d'un travail.

Cas d'utilisation	**Description**
Consulté les items et les enchères	Pour enchérir, faire une proposition sur les items qui l'intéresse.
Posté les items	Remplir un formulaire pour enregistrer les items
Consulté les enchères de ses propre items	Prendre une décision en consultant l'algorithme
Gérer les utilisateurs	Ajouter, modifier ou supprimer un utilisateur ainsi que notifier un utilisateur
Consulté espace perso	Consulter ou modifier le profil

Tableau 2.2 : Description des cas d'utilisation.

2.3.Diagramme de cas d'utilisation :

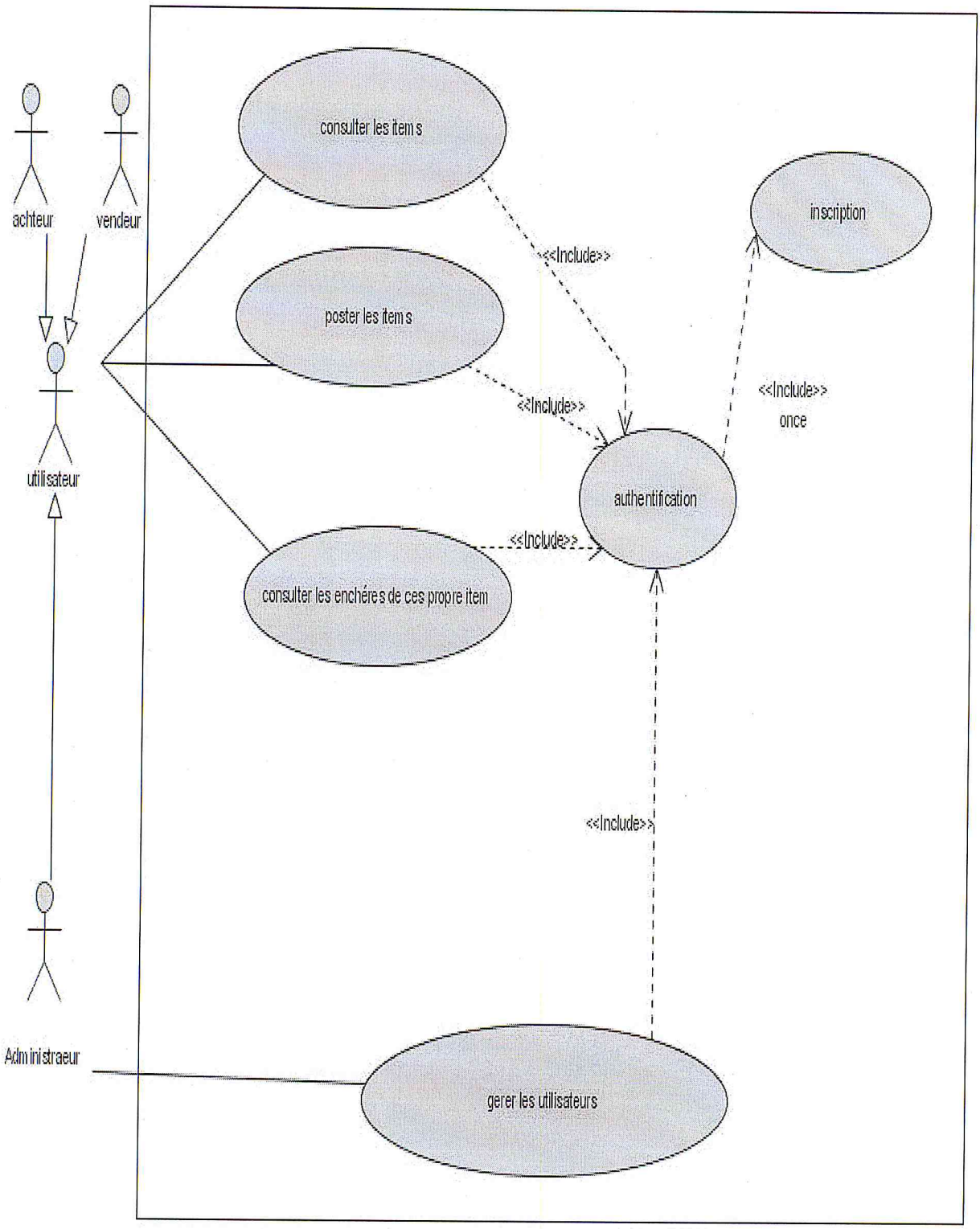

Diagramme 2.1 « Diagramme de cas d'utilisation globale »

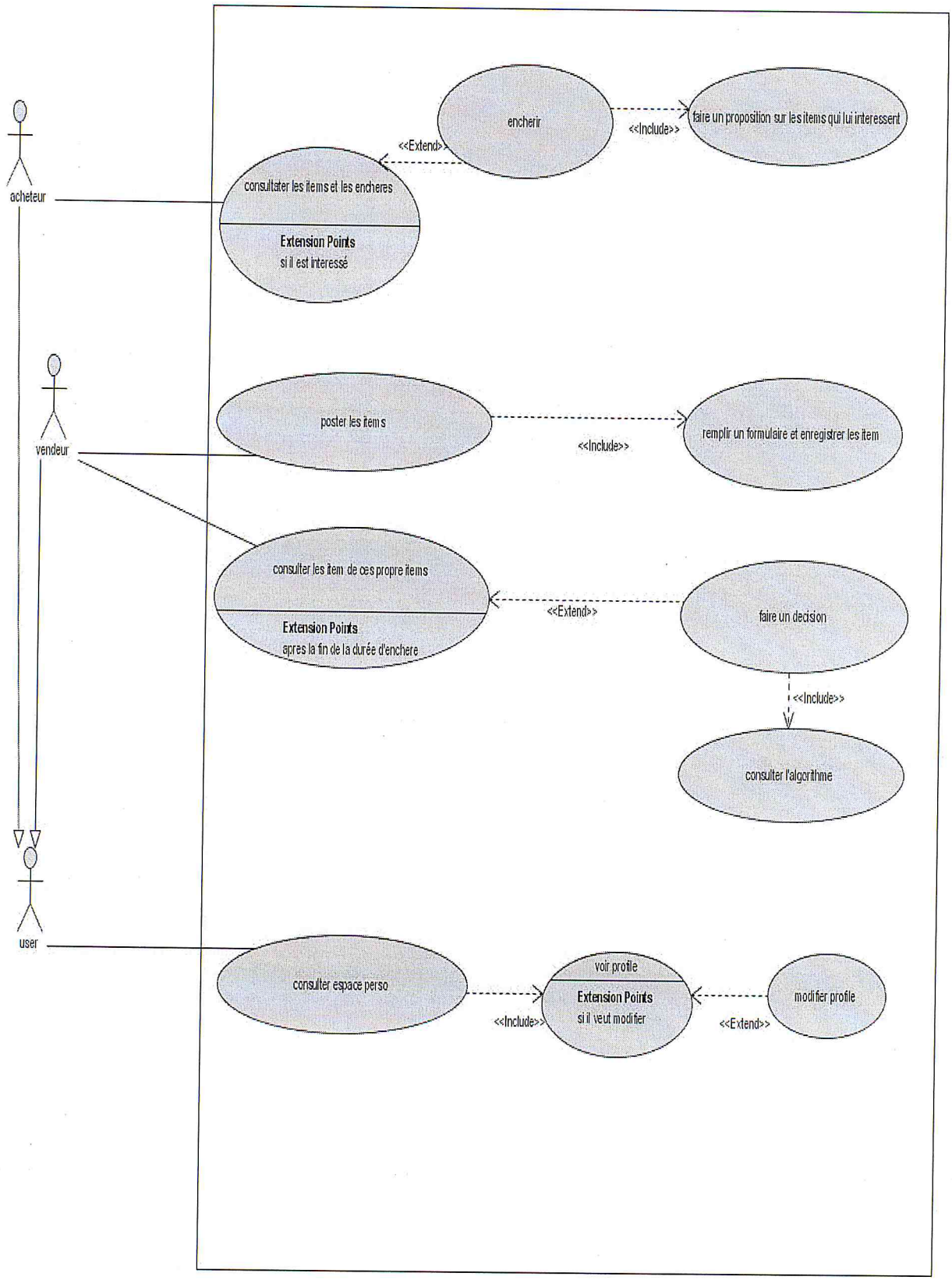

Diagramme 2.2 « Diagramme de cas d'utilisation pour acheteur et vendeur »

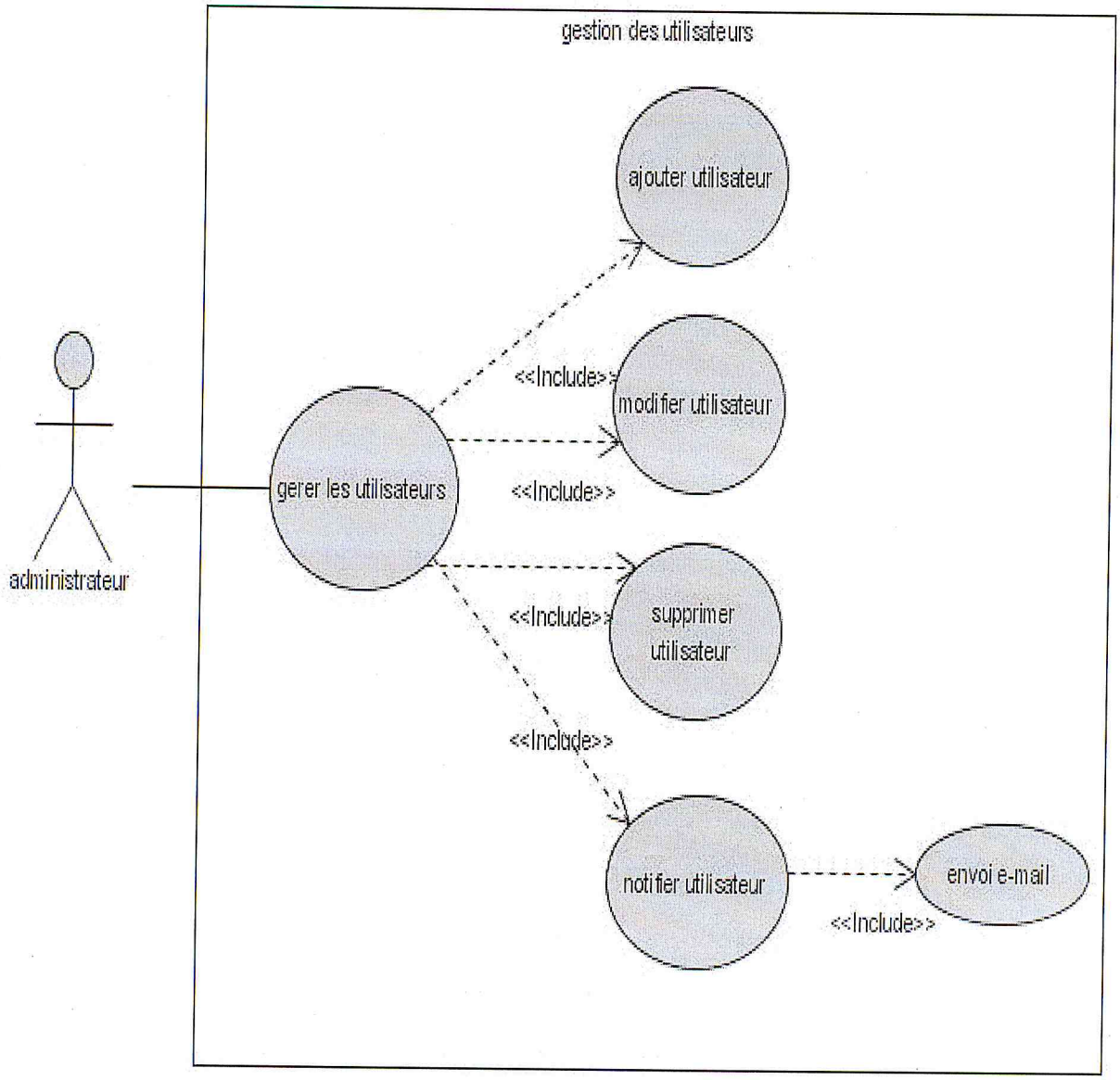

Diagramme 2.3 « Diagramme de cas d'utilisation gestion des utilisateurs »

3. Diagramme de classe :

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe. Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

3.1. Dictionnaire de données :

Classe	Attributs	Désignation	Taille	Type
Encherire	proposition isWon	-La proposition sur les items - si il a gagné l'enchère alors 1 Si non c'est 0		float tinyint()
Categorie	*n_cat* cat	-L'identifiant de la classe Categorie -Nom ou désignation de Categorie	11 100	int varchar
Enchere	*n_enchere* date_debut date_fin heure_deb ut heure_fin etat	-L'identifiant de la classe Enchere -Date de début d'enchère -Date de fin d'enchère -Heure de début d'enchère -Heure de fin d'enchère -Etat d'enchère : encoure Si non terminé.	11 30	int date date time time String

Item	*n_item* designatin description	-L'identifiant de la classe Enchere -désignation de l'item -description sur l'item	11 50	int varchar
Compte	username password	-Nom d'utilisateur pour l'authentification. -Mot de passe d'utilisateur	50 50	varchar varchar
User	n_user nom prenom date_naissance sexe adresse n_ccp n_compte_banc email role	-L'identifiant de la classe Enchere -Le nom d'utilisateur -Le prénom d'utilisateur -La date de naissance d'utilisateur -Le sexe -L'adresse d'utilisateur -Le numéro de compte postal d'utilisateur -Le numéro de compte bancaire d'utilisateur -L'email d'utilisateur -Le rôle d'utilisateur	11 50 50 7 50 8 15 50 5	int varchar varchar date varchar varchar int int varchar varchar
Acheteur				
Vendeur				
Admin				

Tableau 3.1 : Dictionnaire des données.

Remarque :

-les classes Acheteur, Vendeur et Admin hérite de la classe User
Ont les mêmes attributs comme la classe mère la différance est dans les méthodes.

-pour les attributs qui ont des valeurs nulles dans la colonne taille, leur taille est gérée par le SGBD.

3.2.Diagramme de classe :

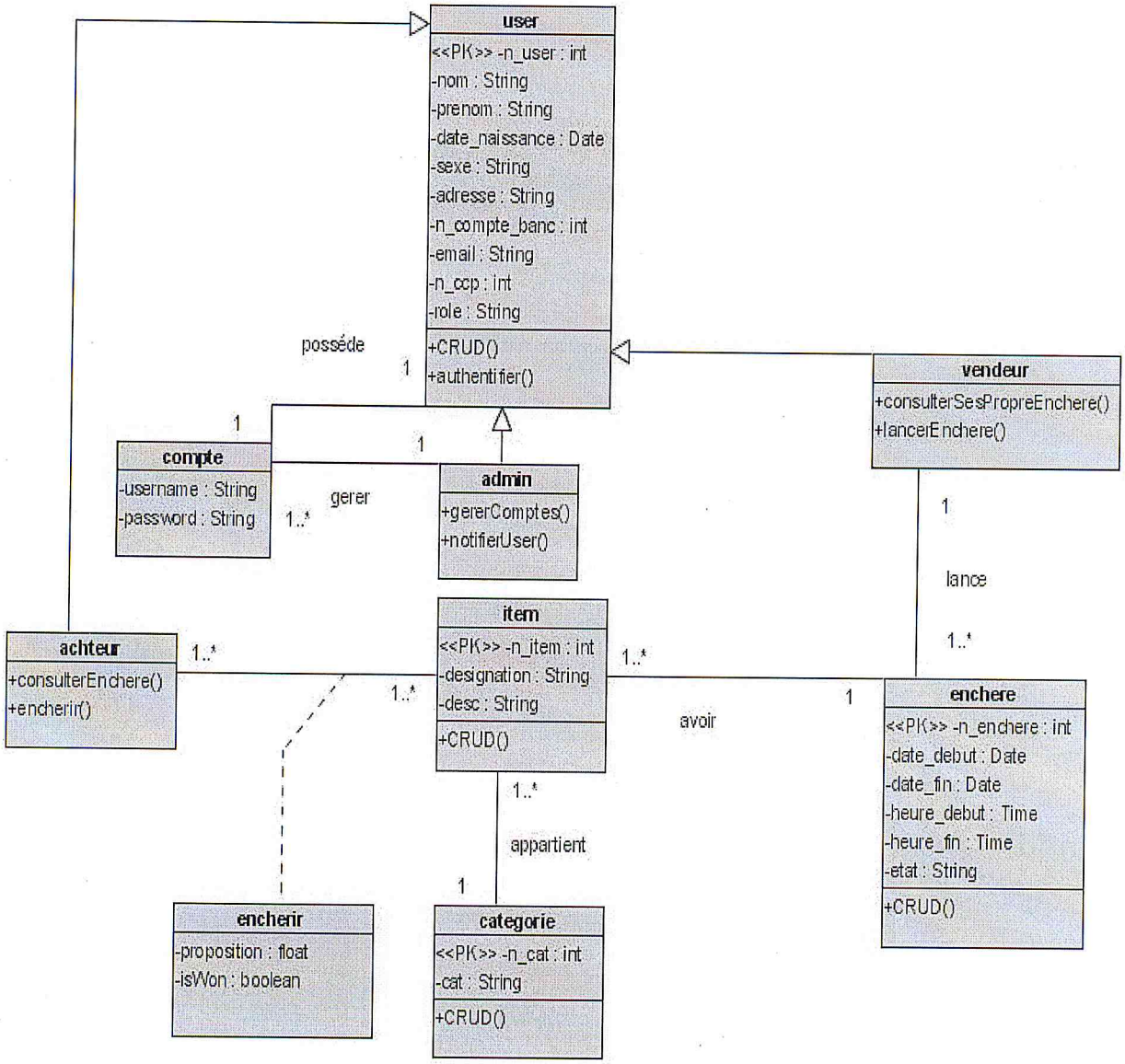

Diagramme 3.2 « Diagramme de classe»

4. Diagramme de séquences :

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.

On montre ces interactions dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

Les périodes d'activité des classes sont symbolisées par des rectangles.

Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets.

On va représenter notre conception par les diagrammes de séquences suivant :

- **Scénario 1 : Inscription**

Pour la séquence inscription, ce si est le déroulement des événements :

- L 'utilisateur effectue une demande d'inscription.
- Le système fait appelle à la page d'inscription.
- La page inscription fourni le formulaire à remplir.
- L'utilisateur saisi ses informations.
- La page inscription fait appel au SGBD pour la sauvegarde des informations.
- Si la procédure d'enregistrement se fait avec succès.
- La page d'inscription remet un message à l'utilisateur disant que le traitement s'est effectué avec succès.
- Si non.
- La page inscription renvoi le formulaire.

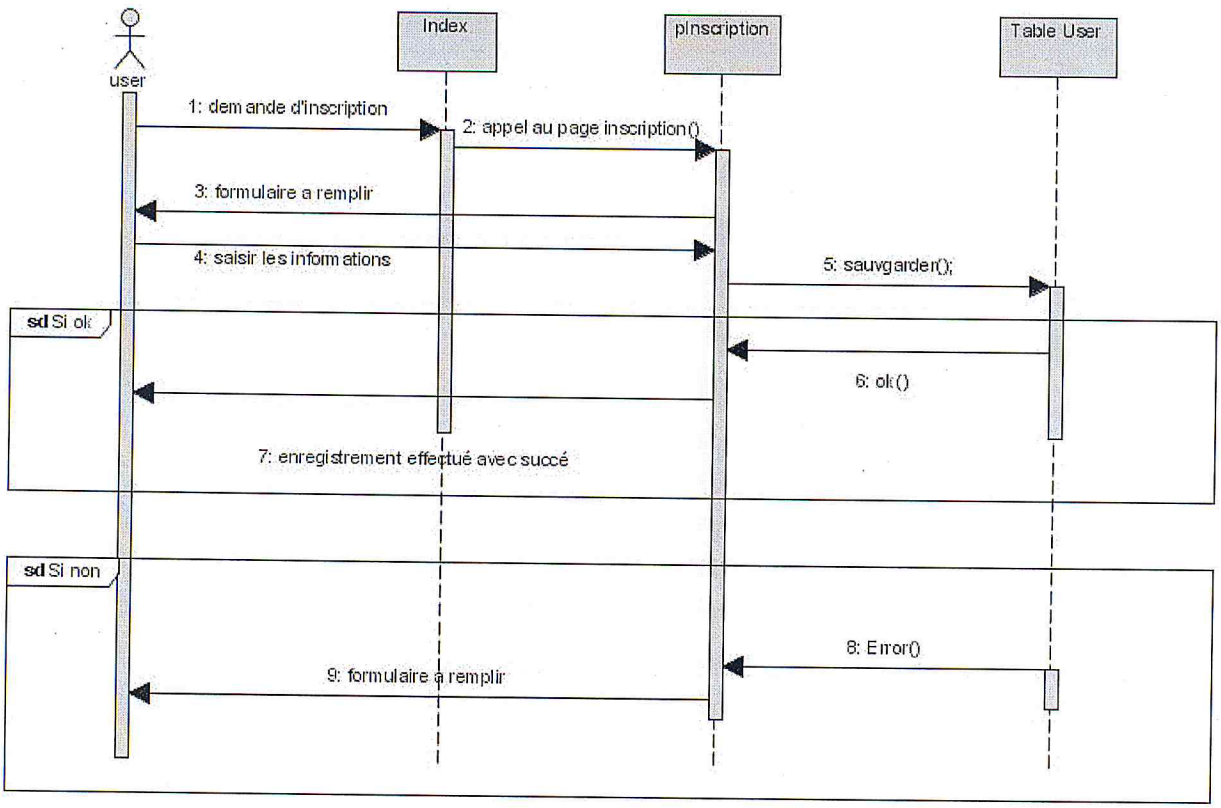

Diagramme 4.1 : Diagramme de séquence Inscription.

- **Scénario 2 : Authentification**

  - L'utilisateur demande à s'authentifier.
  - La page authentification envoi un formulaire.
  - L'utilisateur saisi son nom d'utilisateur et son mot de passe.
  - La page authentification fait appel à la fonction vérifier () qui a son tour fait appel au SGBD pour vérifier l'existence du compte.
  - Si le compte existe la page authentification fait une redirection vers la page index.
  - Si non ou en cas d'erreur, la page authentification renvoi le formulaire précédent.

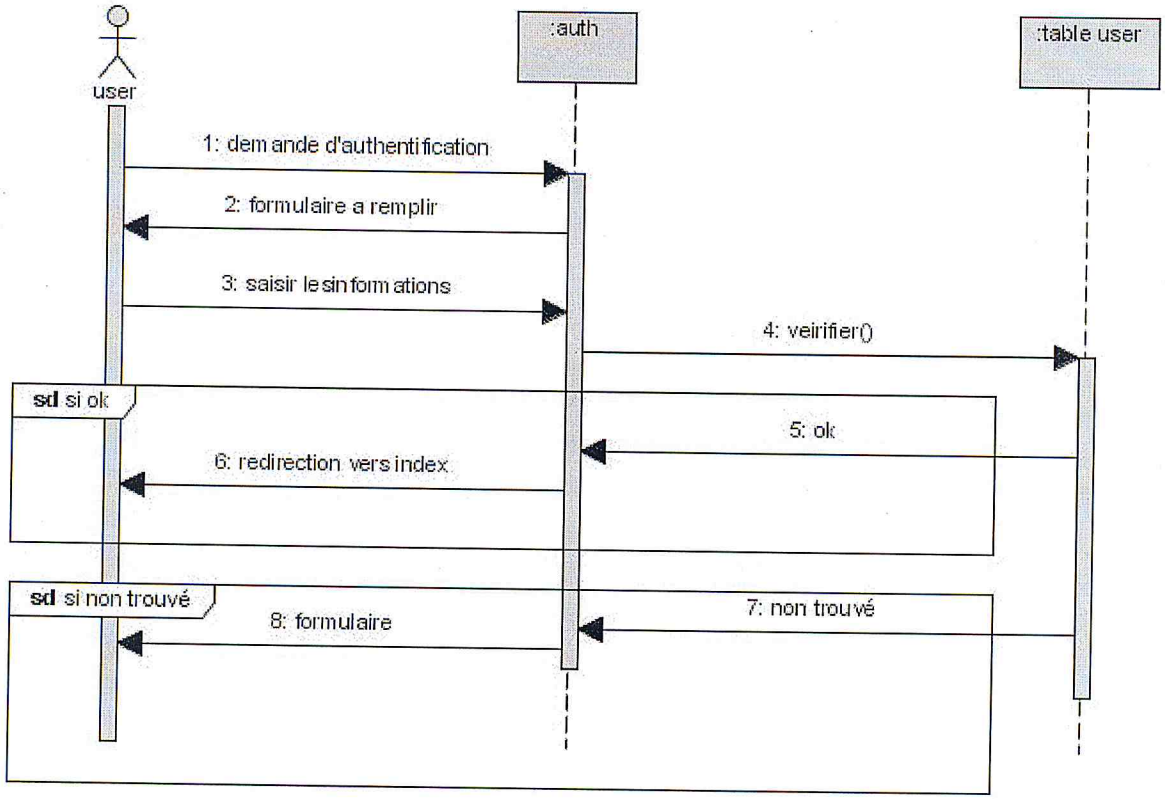

Diagramme 4.2 : Diagramme de séquence Authentification.

- **Scénario 3 : Consulter profil**

  - L'utilisateur demande a consulté son profil.
  - La page profil fait appel a la méthode select(n_user) avec le numéro de l'utilisateur comme paramètre, cette méthode fait appel au SGBD pour exécuter la requête.
  - Le SGBD envoi le résultat.
  - Un détail du profile envoyé au utilisateur par la page profil.
    Si l'utilisateur veut faire une mise à jour :
  - Il fait une demande.
  - La page des mises à jour consulte le SGBD.
  - Le SGBD envoi le résultat de la requête.

- Un formulaire remplie avec les informations de l'utilisateur est envoyé.
- L'utilisateur fait sa mise à jour.
- La page mise à jour appel le SGBD pour l'enregistrement des mises à jour & il retourne un ok.

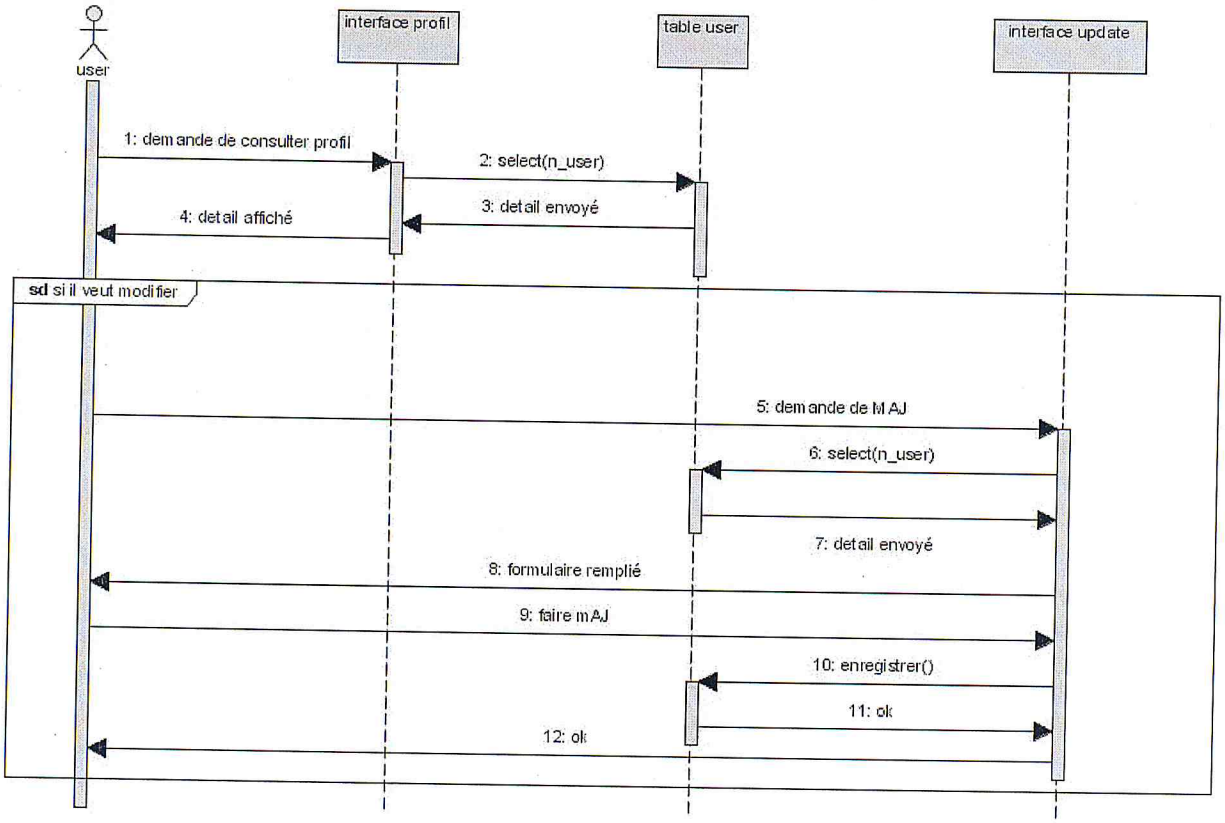

Diagramme 4.3 : Diagramme de séquence Consulter profil.

- **Scénario 4 : Poster items**

  - L'utilisateur fait une demande d'ajout d'items et d'enchères
  - La page additem lui envoi un formulaire.
  - L'utilisateur fourni les informations sur les items.
  - La page additem appel le SGBD pour enregistrer les information dans la table enchère.
  - Un autre appel enregistre les informations dans la table items.
  - La page renvoi la liste d'items ajouté.
  - L'utilisateur lance l'enchère.
  - Le SGBD sauvegarde des informations supplémentaires comme le temps de départ, temps d'arrêt, date départ et date d'arrêt.

- Le message de réponse.
- Un détail retourné qui contient les informations sur les items et sur les enchères.

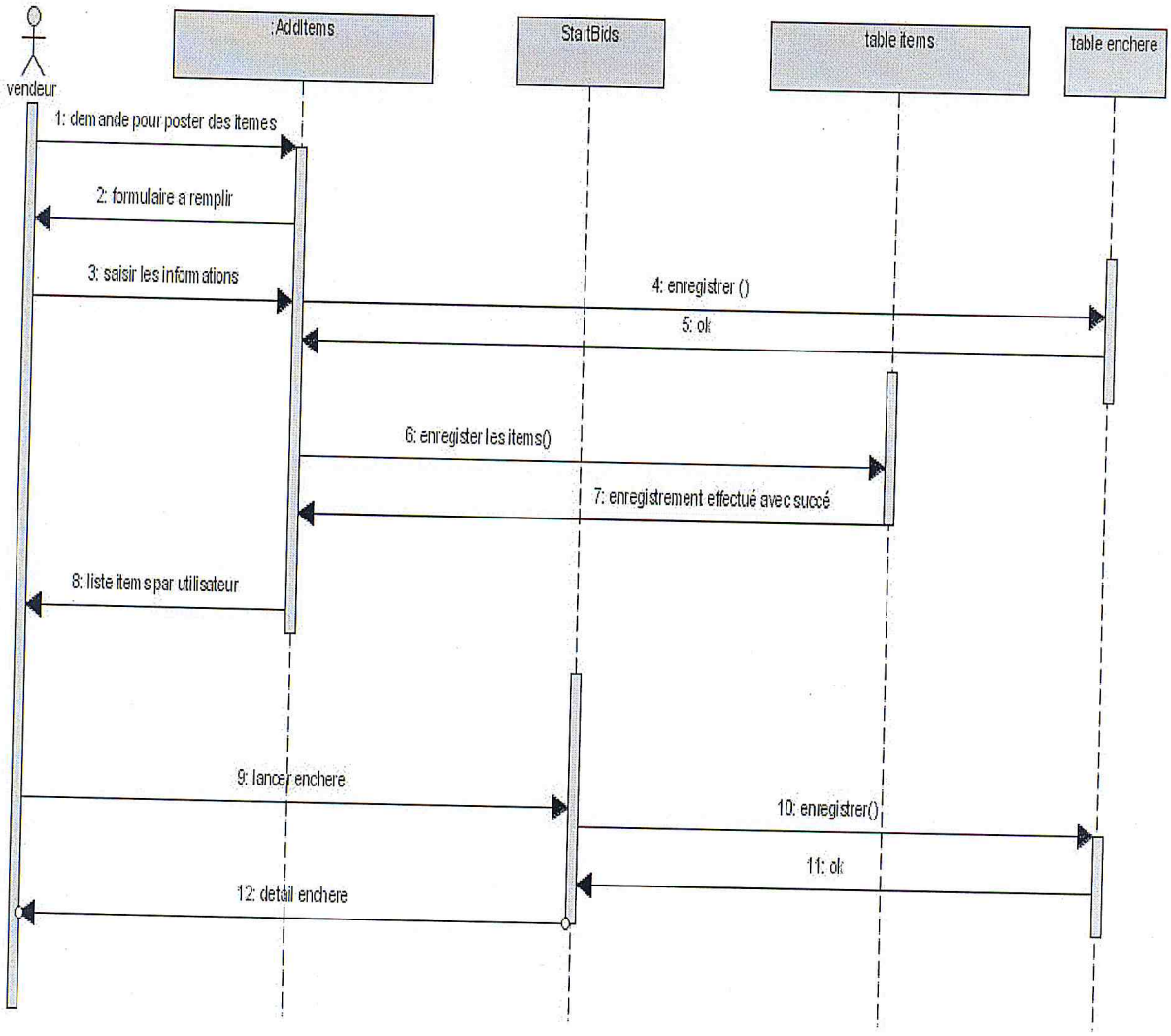

Diagramme 4.4: Diagramme de séquence Poster items.

- **Scénario 5 : Enchérir**

  - L'utilisateur demande à voir la liste des enchères en cours.
  - La page liste enchère déclenche la méthode lister () pour ramener la liste des enchères.
  - La liste est envoyée.
  - L'utilisateur sélectionne une enchère.
  - Le détail des enchères est listé.
  - Lancement de l'enchère
    S'il est intéressé il va faire une proposition.

- Le système sauvegarde la proposition et lui renvoi une liste qui contient tous ses propositions dans le système.

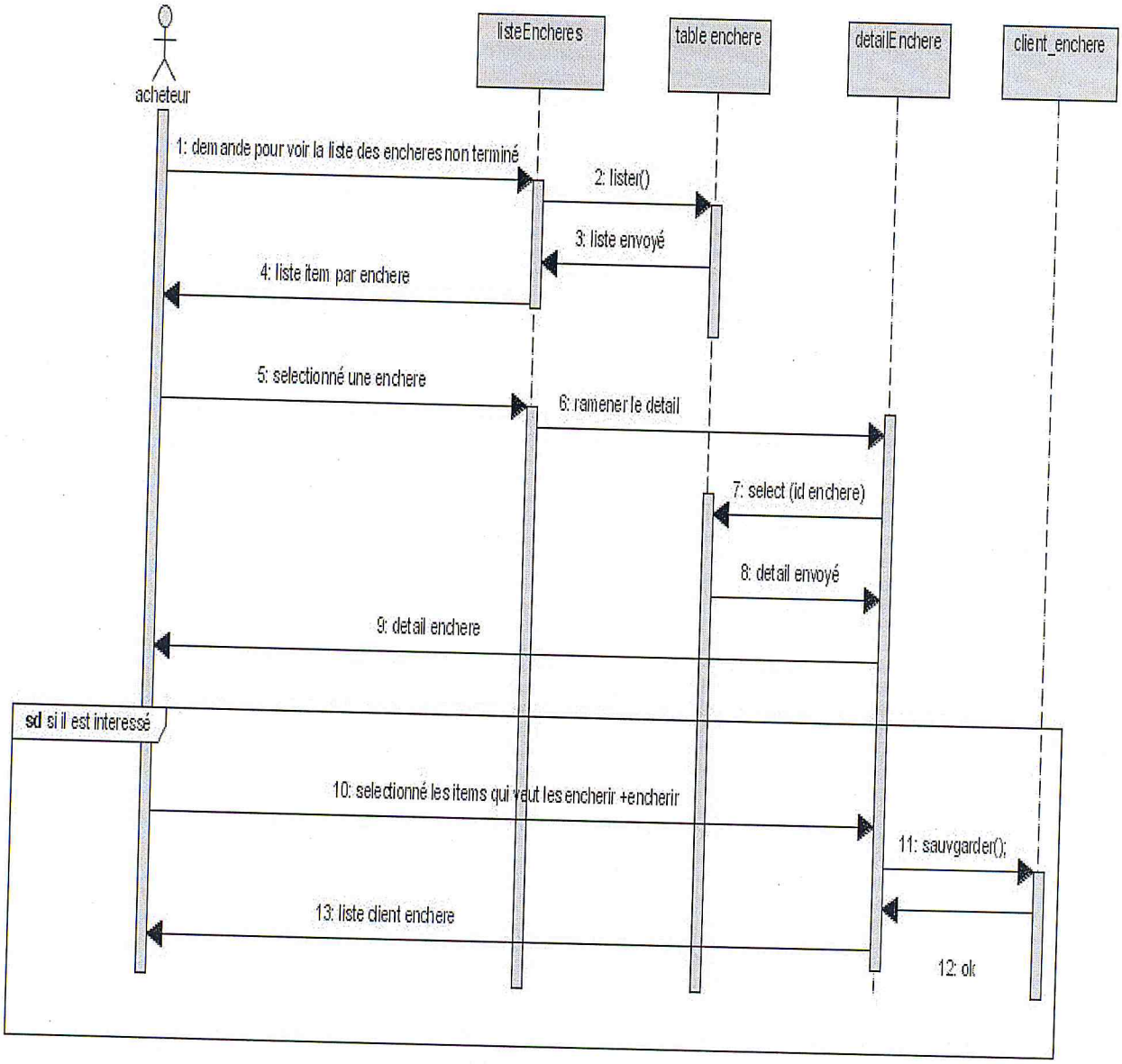

Diagramme 4.5 : Diagramme de séquence Enchérir.

- **<u>Scénario 6</u> : Consulté la liste d'items de ses propres enchères.**

  - L'utilisateur demande de voir sa liste des enchères.
  - La liste est retournée.
  - Il sélectionne une enchère.
  - Le détail d'enchère est retourné.
    Si le temps des enchères est terminé.
  - L'utilisateur demande au système le résultat, le système consulte l'algorithme.

- Le système effectue une mise à jour sur la colonne isWon.
- Le système renvoi le résultat au client.

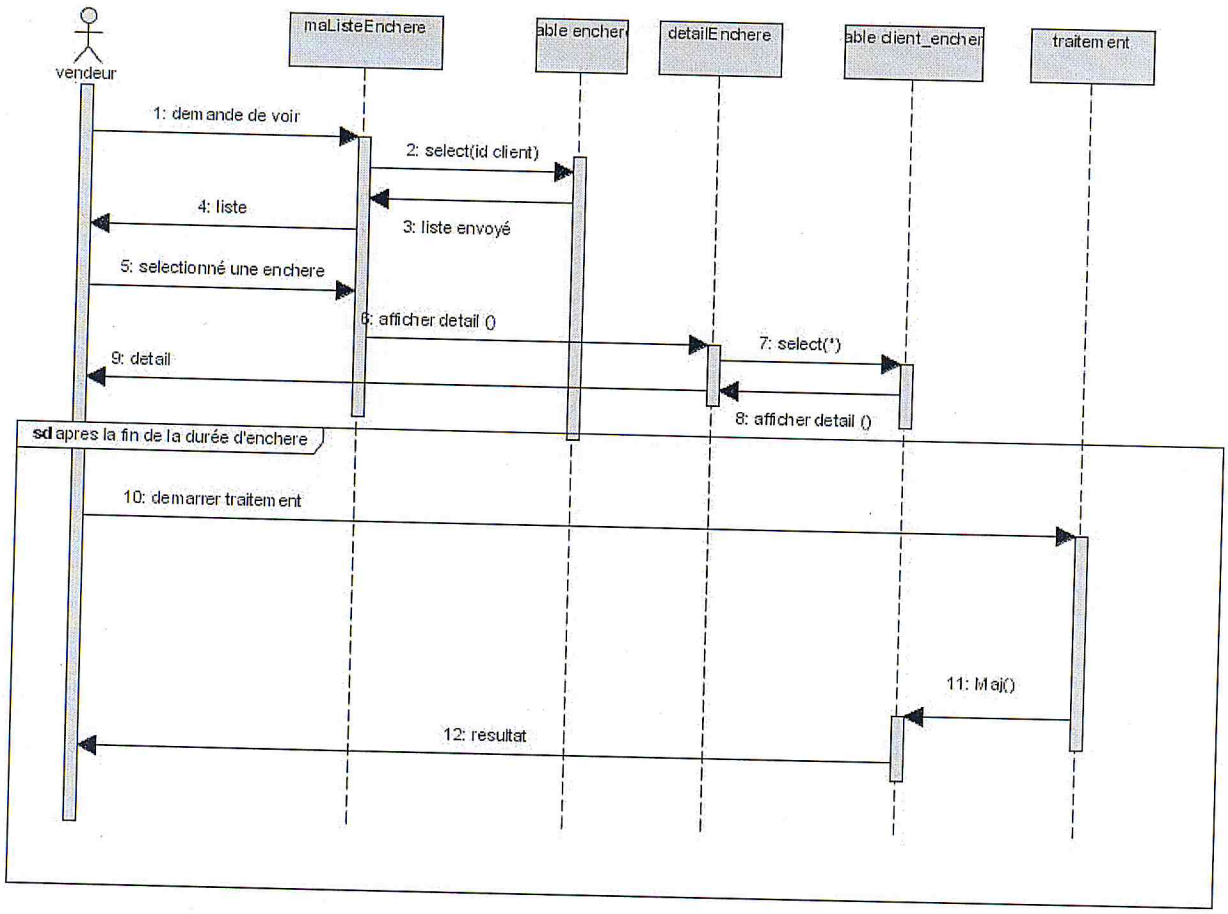

Diagramme 4.6 : Diagramme de séquence Consultation de la liste d'items de ses propres enchères.

- **Scénario 7 : Ajouter utilisateur**

  - L'administrateur demande d'ajouté un utilisateur.
  - Le système lui envoi un formulaire.
  - L'administrateur fourni les informations nécessaires.
  - Le système sauvegarde.
    Si c'est fait.
  - Le système lui affichera un message de succès.

Si non.

- Le système lui renvoi le formulaire une autre fois.

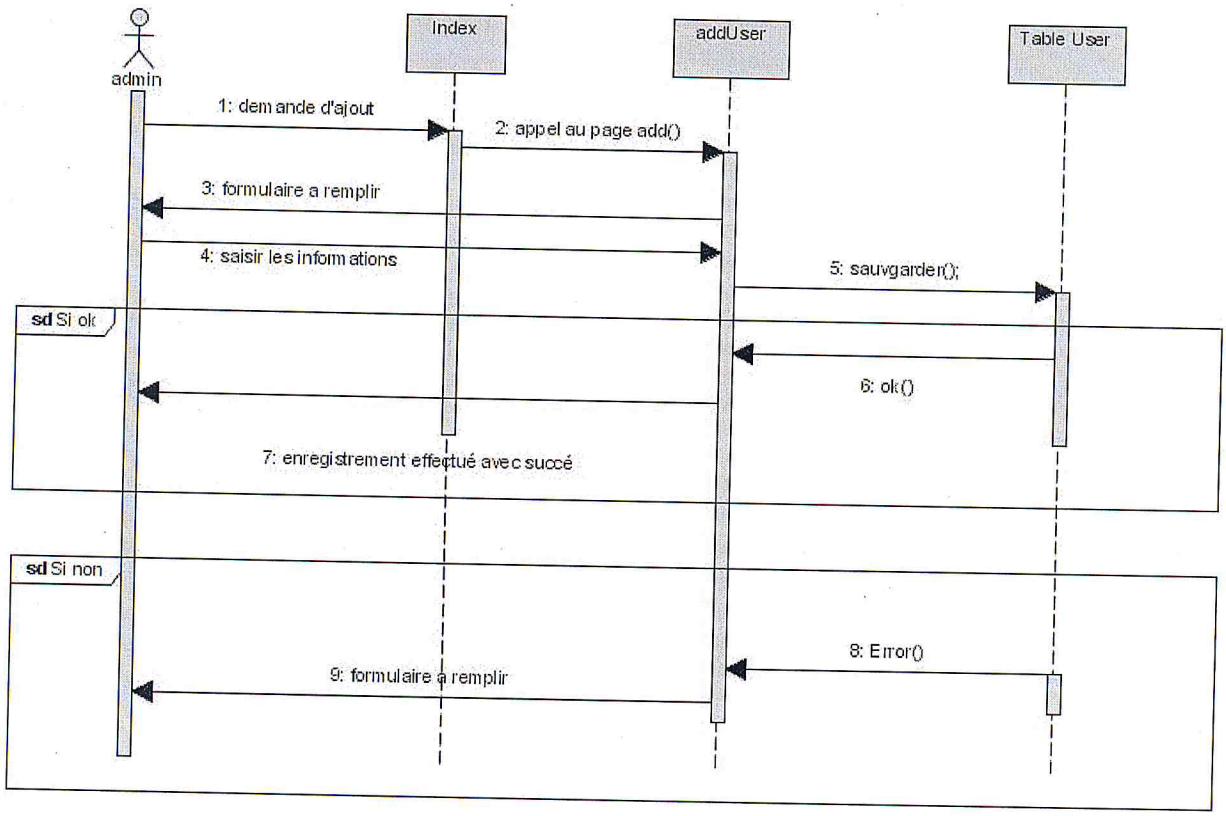

Diagramme 4.7 : Diagramme de séquence Ajouter utilisateur.

- **Scénario 8 : Modifier utilisateur**

  - L'administrateur demande de modifier un utilisateur.
  - Le système lui envoi la liste des utilisateurs.
  - L'administrateur sélectionne un utilisateur.
  - Le système lui envoi son formulaire rempli.
  - L'administrateur effectue la mise à jour.
  - Le système sauvegarde les modifications.
    Si c'est fait.
  - Le système retourne un message de succès.
    Si non.
  - Le formulaire remplie de l'utilisateur sélectionné est envoyé une autre fois.

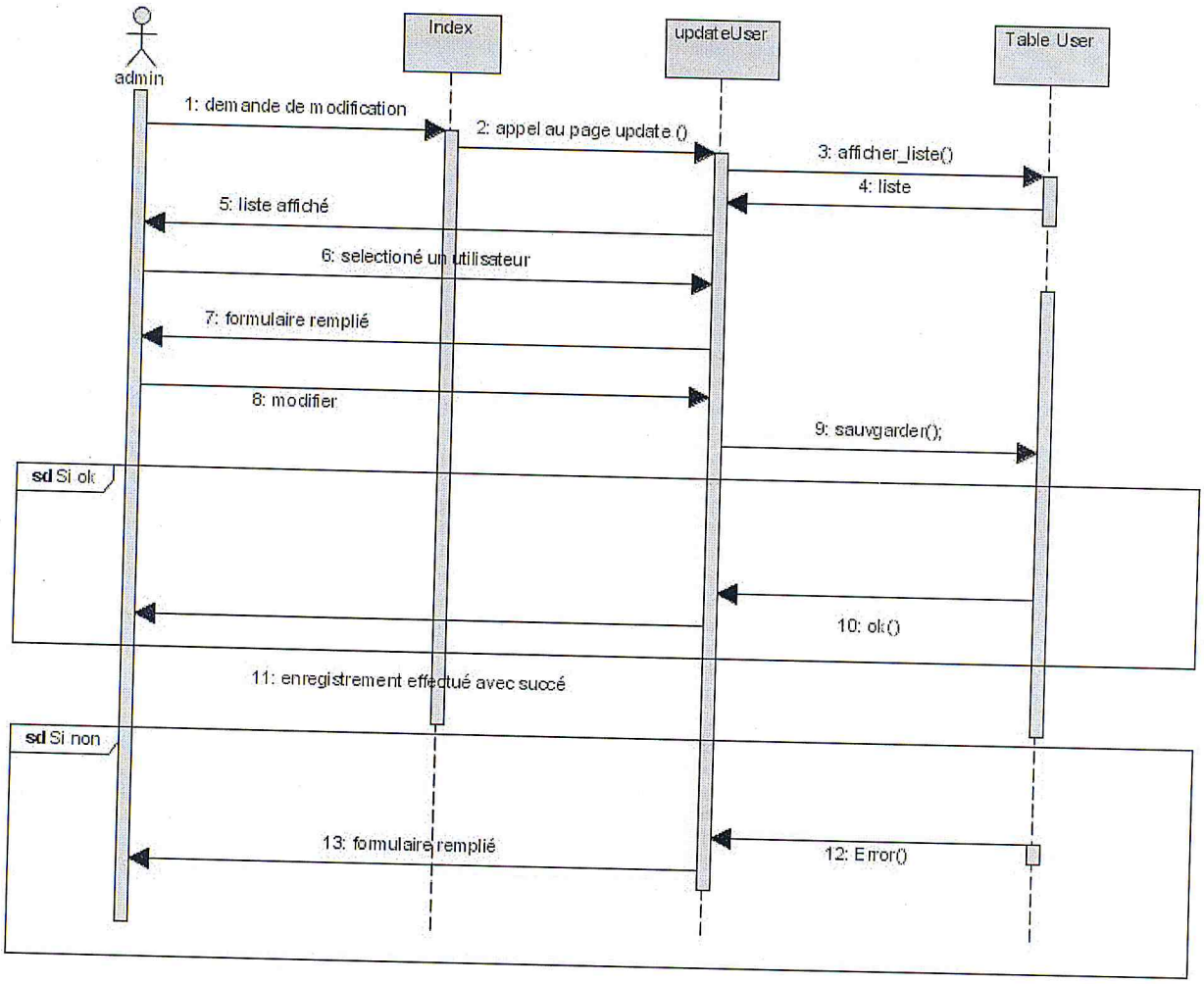

Diagramme 4.8 : Diagramme de séquence Modifier utilisateur.

- **Scénario 9 : Supprimer utilisateur**

  - L'administrateur demande à supprimer un utilisateur.
  - Le système lui envoi la liste des utilisateurs.
  - L'utilisateur sélectionne un utilisateur.
  - Le système sauvegarde les modifications.
  - Le système renvoi un message de succès.
    Si non.
  - La liste est envoyée une deuxième fois.

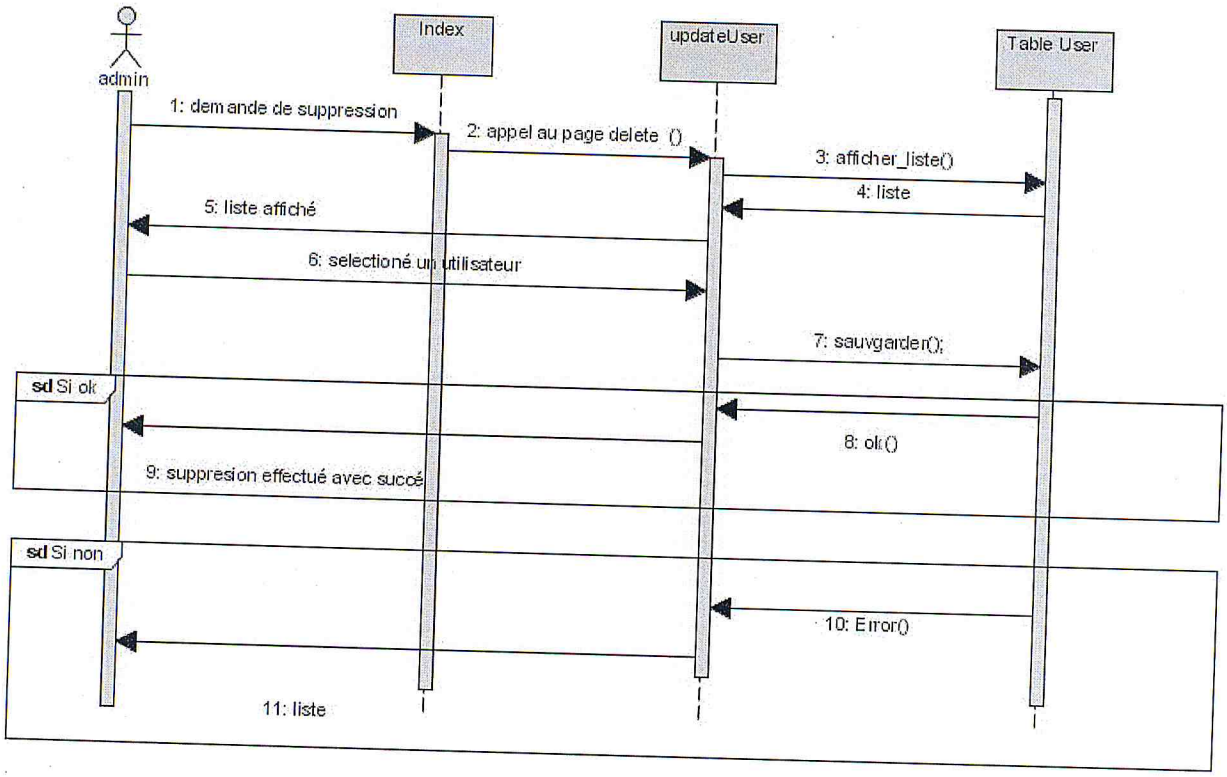

Diagramme 4.9 : Diagramme de séquence Supprimer utilisateur.

5. PASSAGE AU MODELE RELATIONNEL :

Voici un ensemble de règle simple pour passé du modèle relationnel au modèle objet et qui doit donc tenir que de deux concept la relation et les attributs.

R1 : Chaque classe devient une relation :

R2 : Que faire avec les méthodes ?

Voici trois possibilités de traité les méthodes sans faire appel aux traitements procéduraux

- Mémoriser les attributs calculés.
- Utiliser des vues.
- Utiliser les méthodes de mise à jour.

R3 : traduire les associations :

L'objectif est de mémorisé les liens entres les objets des deux classe de l'association.

R4 : L'association attribuée : elle est traité de la même façon que l'association a la quelle elle est rattachées

R5 : Agrégation et composition : l'agrégation et la composition se traitent comme les associations.

Notre base de données prend le schéma suivant :

-Encherir {n_enchere(FK), n_user(FK), n_item, proposition, isWon }.

- categorie { n_cat, cat}.

- enchère { n_enchere, n_user(FK), description, date_debut, date_fin, heure_debut, heure_fin, etat }.

- item { n_item, n_enchere (FK), designation, cat, desc }.

-user {n_user, nom, prenom, date_naissance, sexe, adresse, n_ccp, n_compte_banc, email, username, password, question, answer, role}.

6. Conclusion :

Dans ce chapitre nous avons commencé par une brève présentation de notre langage de modélisation, après nous somme passé à la conception de notre système par des diagrammes de cas d'utilisations, Diagramme de classe et diagrammes de séquences, et nous avons terminé par le passage au modèle relationnel.

Chapitre 4 :
Implémentation

1-Introduction :

Comme notre projet est consisté à résoudre le problème de détermination du gagnant (WDP) pour une enchère combinatoire sur un site web, nous avons découpé la phase implémentation en deux parties

- ✓ La partie traitement et résolution : c'est la partie importante de notre projet.
- ✓ La partie web : pour la gestion des enchères (creation, faire proposition,….).

Qui vont être reliées à la fin, pour atteindre l'objectif de notre projet.

2-La partie traitement et résolution :

I. Introduction :

Résoudre des problèmes ***NP-difficiles*** d'optimisation combinatoire à l'optimalité est souvent un travail immense nécessitant des algorithmes efficaces. Un algorithme par séparation et évaluation est de loin l'outil le plus largement utilisé pour résoudre de manière exacte les problèmes NP-Complet d'optimisation combinatoire. Le choix des heuristiques est un critère décisif dans un algorithme Branch and Bound ; celle-ci conditionnent fortement le temps de calcul, car selon leurs qualité elles permettent une réduction plus au moins importante de l'espace de recherche. Le temps de calcul peut aussi être amélioré par l'analyse du problème traité. En exploitant ses caractéristiques on aboutit à une réduction de sa taille et par conséquent la taille de son espace de recherche.

Pour le problème des enchères combinatoires toutes les solutions proposées et exposées dans le chapitre précédent (CASS, CABOB, CABRO) utilisent la même heuristique, par contre pour la réduction de la taille du problème, CABRO se distingue par une phase de prétraitement plus développées permettant d'exploiter eu maximum les caractéristique de l'instance.

Rappelons que notre objectif est d'implémenter un algorithme pour la résolution du problème des enchères combinatoires. Pour ce faire, et après prospection, nous avons choisi de nous inspirer des travaux précédents, à savoir : CASS pour sa structure de données que nous avons jugée intéressante pour l'extraction des relations d'incompatibilité entre les offres, ainsi que de CABRO pour ses phases de prétraitements.

Nous avons opté lors du développement de notre application pour le langage JAVA. Langage orienté objet pourvu d'une riche bibliothèque qui facilite grandement la programmation concurrente grâce à un haut niveau d'abstraction. De plus il permet une portabilité des applications sur différents systèmes d'exploitation.

L'environnement de développement intégré (EDI) utilisé est l'environnement Open Source Netbeans7.

➢ Démarche de l'algorithme :

Avant d'exposer l'algorithme nous donnons quelques notations utilisées.

Dans une enchère combinatoire le commissaire-priseur reçoit un ensemble d'enchères $B = \{b_1, \dots, b_n\}$. Chaque enchère est composée d'un prix $p(b_i)$ et d'un sous-ensemble d'articles $g(b_i)$ de taille $n(b_i)$ $(n(b_i) = |g(b_i)|$. l'ensemble de tout les articles est donné par $I = \{it_1, \dots, it_2\}$. $b(it_i)$ dénote l'ensemble des enchères qui concerne l'article it_i et $C(b_i)$ l'ensemble des enchères compatibles avec b_i. De plus, $C(b_i, b_j)$ et $\neg C(b_i, b_j)$ indiquent que les enchères b_i et b_j sont compatibles ou incompatibles.

Pour certaines instances, il n'est pas nécessaire d'exécuter toutes les phases de l'algorithme, par exemple quand la solution optimale est trouvée avant la phase de recherche. L'algorithme est capable de s'arrêter prématurément soit quand toutes les enchères ont été supprimées ou lorsqu'à un certain point de l'exécution la borne inférieure globale *GLB* rencontre la borne supérieure globale *GUP*.

Cette phase utilise des algorithmes rapides (de complexité polynomiale) afin de réduire la taille du problème en supprimant les offres qui ne peuvent appartenir à la solution optimale ou ceux qui y apparaitront surement. Cette phase tient en 7 stratégies indépendantes (7 étapes) utilisant différents critères pour supprimer des offres.

- *Etape 1 : Offres avec aucune compatibilité*. Dans cette étape toutes les offres qui n'ont aucune autre offre compatible sont supprimées, hormis l'offre de plus grand prix b_h. Ces offres ne feront surement pas partie de la solution optimale car le bénéfice d'une solution contenant une de ces offres ne dépasserait pas son propre prix qui est inférieur aux prix de b_h.
  Exemple :

  $I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères
  $B_1 =< \{I_1, I_2, I_4\}, 1500.00 >$
  $B_2 =< \{I_2, I_3\}, 2000.00 >$
  $B_3 =< \{I_4\}, 1000.00 >$
  Meilleur offre $b_h = B_2$

  L'offre B_1 va être supprimée car sa liste d'offre compatible est vide: $C(B_1) = \emptyset$ et son prix est inferieur à la meilleure offre :
  $P(B_1) = 1500.00 < P(B_h) = 2000.00$

- *Etape 2 : Offres de la solution.* Pour certaine instances il peut y avoir des offres dont les articles sont uniques (seule offre à contenir ces articles), qui n'ont donc aucune offre incompatible. Dans ce cas, l'offre est assurément une partie de la solution optimale. Cette étape identifie toutes les offres avec cette particularité et les rajoute à la solution optimale *BOS* (Bids of the Optimal Solution), puis les supprime de l'ensemble des offres.

Exemple :

$I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères
$B_1 =< \{I_1, I_2, I_3\}, P_1 >$
$B_2 =< \{I_2, I_3\}, P_2 >$
$B_3 =< \{I_4\}, P_3 >$
$B_4 =< \{I_1, I_2\}, P_4 >$
L'offre B_3 va être écartée du problème ; puisqu'elle est compatible avec toutes les autres offres, elle fera surement partie de la solution optimale.

- *Etape 3* : *Offres dominées.* Ce prétraitement est le même que celui réalisé par CASS [LEY03, FLY99] et CABOB [San05] : élimination des offres dominées. Une offre est dite dominée par une autre offre si l'ensemble de ses articles englobe l'ensemble d'articles d'une autre offre et que son prix est moindre. Plus formellement, pour toute paire d'offres (b_1, b_2) telle que $g(b_1) \subseteq g(b_2)$ et $P(b_1) \geq P(b_j)$, b_j peut être supprimée car elle ne sera jamais préférée à b_i.
  Exemple :

  $I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères
  $B_1 =< \{I_1, I_2, I_4\}, 1500.00 >$
  $B_2 =< \{I_1, I_4\}, 1500.00 >$
  $B_4 =< \{I_4\}, 1000.00 >$
  L'offre B_1 est dominée par B_2 et sera supprimée du problème car :
  $S(B_2) = \{I_1, I_4\} \subset S(B_1) = \{I_1, I_2, I_4\}$ et $P(B_2) = 1500 \geq P(B_1) = 1500$.

- *Etape 4 : Offres 2-dominées.* Cette étape est une extension de l'étape précédente, qui vérifie si une offre est dominée par une paire d'offres. Dans certain cas, une offre n'est pas dominée par une seule autre offre mais par l'union de deux offres (union des ensembles d'articles et addition des prix). Cette étape peut aisément être généralisée à une vérification d'*offres n-dominées*. Cependant, pour beaucoup d'instances, la probabilité qu'a une offre d'être dominée par n offres est négligeable pour de grandes valeur de n, c'est pourquoi cette généralisation n'est pas utilisée pour $n > 2$.
  Exemple :

  $I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères
  $B_1 =< \{I_1, I_2, I_4\}, 1500.00 >$
  $B_2 =< \{I_1, I_2\}, 1000.00 >$
  $B_3 =< \{I_4\}, 600.00 >$
  L'offre B_1 est dominée par l'offre $B_2 \cup B_3$ et sera supprimée du problème car :
  $S(B_2) \cup S(B_3) \subseteq S(B_1)$ Et $P(B_2) + P(B_3) = 1600 \geq P(B_1) = 1500$.

- *Etape 5 : Offres pseudo-dominées.* Cette étape est une généralisation plus complexe de la domination. Comme pour la domination (étape 3) on traite des paires d'offres (b_1, b_2) mais dont l'inclusion des ensembles d'offres n'est pas complète : tous les articles de b_1 sont inclus dans l'ensemble d'articles de b_2 excepté un seul article it_k. Dans cette situation l'offre b_1 peut être supprimée

seulement si, additionné à son prix, le prix de la meilleure (de plus grand prix) offre lui étant compatible et contenant it_k ne dépasse pas le prix de l'offre b_j.
Exemple :

Soit $I = \{I_1, I_2, I_3, I_4\}$ l'ensemble d'articles mis aux enchères.
$B_1 =< \{I_1, I_2, I_4\}, 1500.00 >$
$B_2 =< \{I_1, I_2\}, 1000.00 >$
$B_3 =< \{I_4\}, 400.00 >$
$B_4 =< \{I_3, I_4\}, 700.00 >$

$S(B_1) = S(B_2) \cup \{I_4\}$: L'offre B_1 est dominée par l'union des deux offres B_2 et B_4 car B_4 est la meilleur offre compatible avec B_2 contenant l'article I_4et $P(B_2) + P(B_4) \geq P(B_1)$. B_1 Sera par conséquent supprimée.

- *Etape 6 : calcule des bornes inférieures et supérieures.* Dans cette étape des procédés rapides de calcul des bornes inférieures et supérieures pour chaque offre sont effectués dans le but de supprimer les offres avec une borne supérieure inférieure à la *borne inférieure globale* (**GLB - G**lobal **L**ower **B**ound la borne inférieure globale est la meilleure borne inférieure associée à une solution valide) : les offres supprimées ne pourraient pas améliorer la solution trouvée jusqu'alors. La borne supérieure u d'une offre b_x est calculée selon l'équation (1) où $C'(b_x, it_k)$ est l'ensemble des offres compatibles avec b_x contenant l'article it_k. Schématiquement, la borne supérieure d'une offre b_x est calculée en additionnant a son prix les meilleurs prix des offres contenant les articles n'appartenant pas à $g(b_i)$. La borne inférieure d'une offre b_x (notée$l(b_x)$) est ensuite calculée en construisant une solution contenant initialement ladite offre, en lui rajoutant itérativement les solutions compatibles. Les offres compatibles sont ordonnées dans l'ordre descendant de leur borne supérieure (calculée précédemment). Toutes les solutions trouvées par cet algorithme sont des solutions valides qui mettent à jour la borne inférieure globale.

$$u(b_x) = p(b_x) + \sum_{\forall it_k \notin g(b_x)} max_{\forall j \in c'(b_x, it_k)}\, p(b_j)/n(b_j) \qquad (1)$$

Exemple :
Calcul des bornes supérieures :

$I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères.
$B_1 =< \{I_1, I_4\}, 1500.00 >$
$B_2 =< \{I_2, I_3\}, 1000.00 >$
$B_3 =< \{I_2\}, 400.00 >$
$B_4 =< \{I_3\}, 700.00 >$

$$u(B_1) = P(B_1) + \max\left\{\frac{P(B_2)}{2}, \frac{P(B_3)}{1}\right\} + \max\left\{\frac{P(B_2)}{2}, \frac{P(B_4)}{1}\right\}$$
$$= 1500 + 500 + 700 = 2700.$$

$$u(B_2) = P(B_2) + \max\left\{\frac{P(B_1)}{2}\right\} + \max\left\{\frac{P(B_1)}{2}\right\}$$
$$= 1000 + 750 + 750 = 2500.$$
$$u(B_3) = P(B_3) + \max\left\{\frac{P(B_1)}{2}\right\} + \max\left\{\frac{P(B_4)}{1}\right\} + \max\left\{\frac{P(B_1)}{2}\right\}$$
$$= 400 + 750 + 700 + 750 = 2600.$$
$$u(B_4) = P(B_4) + \max\left\{\frac{P(B_1)}{2}\right\} + \max\left\{\frac{P(B_3)}{1}\right\} + \max\left\{\frac{P(B_1)}{2}\right\}$$
$$= 700 + 750 + 400 + 750 = 2600.$$

Calcul de la borne inférieure:
$GLB = \max\{l(B_i)/\ \forall B_i \in B\}$
$l(B_1) = P(B_1) + P(B_3) + P(B_4) = 1500 + 400 + 700 = 2600$
$l(B_2) = P(B_2) + P(B_1) = 1000 + 1500 = 2500$
$l(B_3) = P(B_3) + P(B_1) + P(B_4) = 400 + 1500 + 700 = 2600$
$l(B_4) = P(B_4) + P(B_1) + P(B_3) = 700 + 1500 + 400 = 2600$
$GLB = \max\{2600,2500,2600,3200\} = 2600$
Toutes les offres ayant leur borne supérieure inférieure à la borne inférieure globale GLB doivent être supprimées car aucune solution utilisant ces offres ne pourra être meilleure que celle déjà trouvée. Dans notre exemple B_2 sera supprimée.

- *Etape 7 : Offres dominées selon la compatibilité*. Cette étape est une autre généralisation de la notion d'offres dominées. Une offre b_i est dominée selon la compatibilité par une autre offre b_J si l'ensemble des offres compatibles avec b_i est inclus dans l'ensemble des offres compatibles avec b_j et que son prix est inférieure. Formellement, pour toute paire d'offres (b_i, b_j) telle que $C(b_i) \subseteq C(b_j)$ et $p(b_i) \geq p(b_j)$, b_j peut être supprimée car elle ne sera jamais préférable à b_i.
  Exemple :

  $I = \{I_1, I_2, I_3, I_4\}$ L'ensemble d'articles mis aux enchères.
  $B_1 =< \{I_2, I_4\}, 1000.00 >$
  $B_2 =< \{I_1, I_4\}, 1200.00 >$
  $B_3 =< \{I_3\}, 750.00 >$
  $B_4 =< \{I_1, I_2\}, 1000.00 >$
  $B_5 =< \{I_4\}, 750.00 >$

  L'offre B_1 est dominée par selon la compatibilité par l'offre B_2 car $C(B_1) \subseteq C(B_2)$ et $P(B_1) = 1000 \leq P(B_2) = 1200$. Elle sera donc supprimée du problème.

Au sortir de cette phase, puisque le problème a changé il est préférable de ré-exécuter toutes les étapes précédente pour s'assurer d'avoir éliminé toutes offres

ne pouvant pas appartenir à la solution optimale ou celle qui y apparaitront surement.

Donc la phase 1 doit être est répété jusqu'à ce qu'on ne puisse plus écarter de nouvelles offres.

- Seconde phase : Calcule précis des bornes supérieures et bornes inférieures

Dans cette phase, l'algorithme calcule des bornes supérieures et inférieures plus précises pour chaque offre. La phase débute par un ordonnancement des offres selon les bornes supérieures calculées dans l'étape 6 de la phase précédent par un ordre ascendant. Dans le but de calculer la borne supérieure d'une offre b_i, un problème linéaire PL relaxé est formulé. La formulation relaxée définit le problème en permettant d'accepter partiellement les offres (variables de décision réelles dans [0, 1]), pouvant ainsi être résolu par la méthode du simplexe. La version relaxée ne contient ni l'offre courante b_i ni les offres incompatibles. En additionnant le prix de l'offre b_i à la valeur de la solution du PL relaxé, une borne supérieure plus précise que celle trouvée dans l'étape 6 de la première phase est obtenue, cette dernière servira d'heuristique dans la phase suivante.
En ce qui concerne le calcul de la borne inférieure d'une offre b_i, il se fait en utilisant la solution du PL relaxé « représentant les variables de décision ».

Une nouvelle solution du problème CA est créée contenant dans un premier temps l'offre b_i et les offres ayant leurs variables de décisions supérieures à 0,5. *Cette hypothèse ne produit pas de solutions invalide « contenant des offres incompatibles », car les offres compatibles sont limités à la somme d'au plus 1, donc deux offres incompatibles ne peuvent avoir tous deux des valeurs supérieures à 0,5.*

Après cela, on tentera d'ajouter les offres restantes à la solution (celles avec des valeurs inférieures ou égales à 0,5) dans l'ordre décroissant. Bien sûr, si la solution du problème linéaire avait été une solution entière ce processus ne serait pas nécessaire. Ainsi La borne inferieure de l'offre b_i est la valeur de la solution notée V.

Comme pour l'étape 6 de la première phase à chaque fois que la borne supérieure d'une offre est inférieure à la borne inférieure globale « GLB » l'offre est supprimée vu qu'on ne pourrait pas construire une solution la contenant meilleure que celle déjà trouvée.

Exemple [San05]: calcul des bornes inférieure et supérieure pour une offre B_0 de prix $P(B_0) = 2$, dont les offres compatibles sont données dans le tableau III.1 :

Offres	Articles	Prix
B_1	I_1, I_2	2
B_2	I_2, I_3	2
B_3	$I_1\ I_3, I_4$	2
B_4	I_3, I_4, I_5	2
B_5	I_5, I_6	4.5
B_6	I_6	3

Tableau S2III.1 Tableau représentant une instance d'enchère combinatoire [TSA05]

L'enchère combinatoire décrite dans le tableau III.1 correspond au problème linéaire LP suivant :

$$f = Max\ 2x_1 + 2x_2 + 2x_3 + 2x_4 + 4.5x_5 + 3x_6$$

Sous les contraintes :

$$x_1 + x_3 \leq 1$$
$$x_1 + x_2 \leq 1$$
$$x_2 + x_3 + x_4 \leq 1$$
$$x_3 + x_4 \leq 1$$
$$x_4 + x_5 \leq 1$$
$$x_5 + x_6 \leq 1$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \in [0,1](réels)$$

La résolution du PL avec l'algorithme du simplexe donne le résultat suivant :

Offres	**Variables de décision x_i**
B_1	0.5
B_2	0.5
B_3	0.5
B_4	0
B_5	1
B_6	0
	$f = 7.5$

Tableau S2.III.2 Solution du LP [TSA05]

Ainsi, on calcule la borne supérieure de l'offre B_0, $u(B_0) = P(B_0) + f = 2 + 7.5 = 9.5$.

La borne inférieure de l'offre B_0, $l(B_0) = P(B_0) + P(B_5) + P(B_3) = 2 + 4.5 + 2 = 8.5$

- Troisième phase : Recherche

La troisième phase effectue une recherche avec séparation et évaluation en profondeur d'abord avec l'ensemble d'offres restantes des phases précédentes, L. La meilleure solution trouvée jusqu'à présent est stockée dans la variable globale GLB (Global Lower Bound).

La procédure de recherche commence par appeler branch_and_bound (L, cSolution) Initialement cSolution $= BOS$, rappelant que BOS (Bids of the Optimal Solution) est l'ensemble des offres appartenant surement à la solution optimale construit auparavant dans l'étape 2 de la première phase. Dans ce qui suit nous donnons le pseudo-code de la procédure de recherche :

```
Procedure branch_and_bound(L, cSolution)
Début
    trier(L)    //en ordre décroissant
    Pour toute offre bid de L Faire
    Si (v(cSolution) = GUP) alors Sortir (); //GUP = Global Upper Bound
    Si (v(cSolution) < u(bid) et bid ∈ cSolution et u(bid) > GLB) alors
            cSolution2 = cSolution ∪ bid ;
            bin = cSolution ∪ bid ;
            Si V(cSolution2) > V(GLB) alors GLB ← cSolution2 ;
            compatibleBidsList = C(bid) ;
            Pour toute offre c de compatibleBidsList Faire
                Si (c compatible avec cSolution2 ou c ∈ L) alors
                    compatibleBidsList = compatibleBidsList − c;
                Sinon
                    bin ← bin ∪ c ;
                FSI
            Fin Pour
            LP ← Formuler le problème linéaire à partir du bin ;
            lp_solution ← resoudre LP avec Simplex ;
            Si (LP résolu en N itérations) alors
                VD ← Variable de décision (lp_solution);
                cSolution3 ← nouvelle solution;
                Trier bin en ordre descendant par rapport aux Variables de décisions ;
                Pour toute offre b de bin faire
                    Si b est compatible avec cSolution3 alors cSolution3 ← Csolution3 ∪ b ;
                Fin Pour
                Si (V(cSolution3) > cSolution2(GLB) alors cSolution2 = cSolution3;
                Si V(cSolution2) > V(GLB) alors GLB ← cSolution2 ;
                Si (bid ∈ cSolution2 et V(cSolution2) > u(bid) U(bid) = V(cSolution2) ;
                    Si (lp_solution n'est pas un solution entière et V(lp_solution) > V(GLB))
                    Si (bid ∈ cSolution2id) et compatibleBidsList Non vide) alors
                    branch_and_bound (compatibleBidsList, cSolution2) ;
                FSI
            Sinon
                Si (bid ∈ cSolution2) et compatibleBidsList Non vide alors
                    branch_and_bound (compatibleBidsList, cSolution2) ;
                FSI
            FSI
        FSI
    Fin Pour
Fin
```

L'algorithme construit des solutions au fur et à mesure qu'il parcourt l'espace de recherche et garde la meilleure solution dans la variable globale GLB dont la valeur représentera la borne inferieur.
L'algorithme commence par trier en ordre descendant la liste des offres L en fonction de la borne supérieure « heuristique » calculée auparavant, ensuite on prend dans l'ordre une offre bid de la liste L et on crée une nouvelle liste d'offres bin contenant les offres compatibles avec la solution $cSolution$.

Après cela, l'algorithme formule et résout le problème de programmation linéaire (LP) lié à la solution actuelle afin e construire une nouvelle solution $cSolution3$ et met à jour la meilleure solution trouvée GLB si $cSolution3$ est meilleure.
Si le résultat du problème LP est entier, alors l'algorithme termine la recherche dans la branche actuelle, car la solution optimale de la branche a été trouvée : « Bounding ». Sinon si

la valeur de la fonction objectif du LP est supérieure à la borne inférieure «c'est une branche prometteuse » l'algorithme continue la recherche en profondeur dans cette branche: « Branching », avec comme nouveaux paramètres $< L, cSolution >$ avec L : La liste des offres compatible avec l'offre courante pas encore incluses dans la solution actuelle et $cSolution \leftarrow cSolution2$, la solution construite dans la branche courante.

Exemple :

Soit $I = \{I_1, I_2, I_3, I_4\}$ l'ensemble d'articles mis aux enchères.

$b_1 =< \{I_1, I_2, I_3\}, P_1 >$ $b_2 =< \{I_4\}, P_2 >$
$b_3 =< \{I_1, I_3, I_4\}, P_3 >$ $b_4 =< \{I_2, I_3, I_4\}, P_4 >$
La représentation de données correspondante est :

$b_1 =< 1110, P_1 >$ $b_2 =< \{1\}, P_2 >$
$b_3 =< \{1010\}, P_3 >$ $b_4 =< \{111\}, P_4 >$

On veut vérifier la compatibilité entre deux offres b_i et b_j « *Rappelons que deux offres ne sont compatibles que si elles n'ont aucun article en commun* ».
Soit $|S(b_i)| = n, |S(b_j)| =$ m le nombre d'articles couvert par les offres b_i et b_j.
Avec l'opérateur binaire AND pour savoir si les deux offres b_i et $b_j$sont compatibles il nous suffit de vérifier que $S(b_i)\ And\ S(b_j) = 0$, ce qui demande ≅ Max(n,m)/32 opérations avec un processeur 32 bits.
Avec la méthode classique pour prouver que deux offres sont incompatibles on doit vérifier que $\forall\ Item \in S(b_i)\ , Item \notin S(b_j)$ cette vérification nous demande n*m opérations.

Les bacs «Bins» :

Un bac « Bin » (structure utilisée dans CASS) représente une partition de l'ensemble des offres. Considérons que les articles sont ordonnés, Il y'a un bac pour chaque article, chaque offre appartient au bac correspondant à son article de plus bas ordre.

Bids : ensemble d'offre Type : Liste d'offres.
B : cumul Type : offre.
Le champ b cumul est de type Bid « offre » ou son prix représente la somme des prix des offres composant le bin et sont ensemble d'article représente l'union des articles inclus dans les offres composant le bac « Bin ».

La puissance de cette structure réside dans le fait qu'elle permet de réduire le temps de calcul lors de la vérification de la compatibilité, par exemple afin de trouver la liste d'offres compatible avec une offre bid_i, au lieu de tester la compatibilité avec toutes les offres on fait le test de compatibilité uniquement sur les $bins$ correspondants aux articles non inclus dans l'offre bid_i .
Exemple :

$I = \{I_1, I_2, I_3, I_4\}$ l'ensemble d'articles mis aux enchères.

$b_1 =< \{I_1, I_2, I_3\}, P_1 >$ $b_2 =< \{I_4\}, P_2 >$
$b_3 =< \{I_1, I_3, I_4\}, P_3 >$ $b_4 =< \{I_2, I_3, I_4\}, P_4 >$
$b_5 =< \{I_1, I_2, I_4\}, P_5 >$ $b_6 =< \{I_3, I_4\}, P_6 >$
$b_7 =< \{I_3\}, P_7 >$ $b_8 =< \{I_1\}, P_8 >$
$b_9 =< \{I_3, I_4\}, P_9 >$ $b_{10} =< \{I_2\}, P_{10} >$
$b_{11} =< \{I_1, I_4\}, P_{11} >$ $b_{12} =< \{I_2, I_4\}, P_{12} >$

La représentation des offres correspondant à cette enchère sous forme de Bins est la suivante :

Bin_1	Bin_2	Bin_3	Bin_4
b_1	b_4	b_6	b_2
b_3	b_{10}	b_7	
b_5	b_{12}	b_9	
b_8			
b_{11}			

Figure S2.Erreur ! Il n'y a pas de texte répondant à ce style dans ce document..6 Les Bins

Pour calculer la liste des offres compatibles avec l'offre b_1 : $C(b_1) = \{b_2\}$ on vérifie uniquement la compatibilité avec les offres contenues dans bin_4 car $I_4 \notin S(b_1)$ au lieu de la vérifier avec 11 offres de $B - \{b_1\}$.

Dans la première étape de la phase de prétraitement on doit vérifier qu'une offre $b_i$est compatible avec toutes les autres offres (offre appartenant à la solution optimale), avec la méthode classique on doit vérifier que : $c$est compatible avec toutes les offres appartenant à $-\{b_i\}$. Si $|B| =$ N est la cardinalité de l'ensemble d'offres, le nombre de vérifications à faire est de N-1.
Avec l'utilisation du champ b « cumul » il nous suffit de vérifier que l'offre $b_i$est seule dans son bac et qu'elle est compatible avec tous les champs b des autres offres, soit m vérifications avec m étant le nombre de bins = $|I|$.

Les Solutions «Solution» :

Comme dit précédemment une solution est un ensemble d'offres compatibles deux à deux entre elles, cette classe est une extension de la classe Bin ou la seule différence réside dans le fait qu'on doit vérifier qu'une offre $b_i$est compatible avec celles appartenant au bac avant de pouvoir l'ajouter, le champ b « cumul » va servir encore une fois pour faire cette vérification.

Problème linéaire « LP» :

Représente le problème linéaire sous forme de tableaux utilisés dans l'algorithme du simplexe.
targetCoefficients : coefficients de la fonction objectif Type : tableau de nombres réels.
constraints : coefficient des contraintes Type : tableau 2 dimensions de nombres réels.
equations : type des inégalités : tableau de nombres entiers.
rhs : les coefficients des inégalités Type : tableau de nombres réels.
minimize : définit l'objectif Type : booléen.

Solveur du problème d'enchères combinatoires « CA_Solver » :

C'est la classe principale.
I : l'ensemble d'articles mis à l'enchère Type : tableau d'items.
Bins : l'ensemble d'offre concernant l'enchère Type : tableau de Bins.
bh : meilleur offre Type : Bid.
BOS : Bids of the Solution sert à stocker les offres appartenant surement à la solution optimale calculer dans la phase 1 étape 2 Type : Solution.
GLB : Global Lower Bound stock la meilleure solution trouvée sa valeur représente la borne inférieure globale Type : Solution.
GUP: Global Upper Bound borne supérieure globale sert à arrêter l'algorithme au cas où la borne inférieure l'atteint Type: réel.

3-La partie web :

1. Architecture logique :

L'architecture logique du système proposée est une architecture 3/tiers, L'architecture 3/tier (de l'anglais tier signifiant étage ou niveau) est un modèle logique d'architecture applicative qui vise à séparer très nettement trois couches logicielles au sein d'une même application ou système, à modéliser et présenter cette application comme un empilement de trois couches, étages, niveaux ou strates dont le rôle est clairement défini :

La présentation des données : Elle correspond à la partie de l'application visible et interactive avec les utilisateurs. On parle d'Interface Homme Machine.

Le traitement métier des données Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation, Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

Et enfin l'accès aux données persistantes (persistance en anglais) : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

Dans cette approche, les couches communiquent entre elles au travers d'un « modèle d'échange », et chacune d'entre elles propose un ensemble de services rendus. Les services d'une couche sont mis à disposition de la couche supérieure. On s'interdit par conséquent qu'une couche invoque les services d'une couche plus basse que la couche immédiatement inférieure ou plus haute que la couche immédiatement supérieure (chaque niveau ne communique qu'avec ses voisins immédiats).

Le rôle de chacunes des couches et leur interface de communication étant bien définis, les fonctionnalités de chacune d'entre elles peuvent évoluer sans induire de changement dans les autres couches. Cependant, une nouvelle fonctionnalité de l'application peut avoir des répercussions dans plusieurs d'entre elles. Il est donc essentiel de définir un modèle d'échange assez souple, pour permettre une maintenance aisée de l'application.

Ce modèle d'architecture 3-tier a pour objectif de répondre aux préoccupations suivantes :

Allègement du poste de travail client (notamment vis-à-vis des architectures classiques client-serveur de données);

Prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc.) ;

Introduction de clients dits « légers » (plus liée aux technologies Intranet/HTML qu'au 3-tier proprement dit) ;

Amélioration de la sécurité des données, en supprimant le lien entre le client et les données. Le serveur a pour tâche, en plus des traitements purement métiers, de vérifier l'intégrité et la validité des données avant de les envoyer dans la couche de données.

Et enfin, meilleure répartition de la charge entre différents serveurs d'application.

Exemple d'architecture 3/tiers :

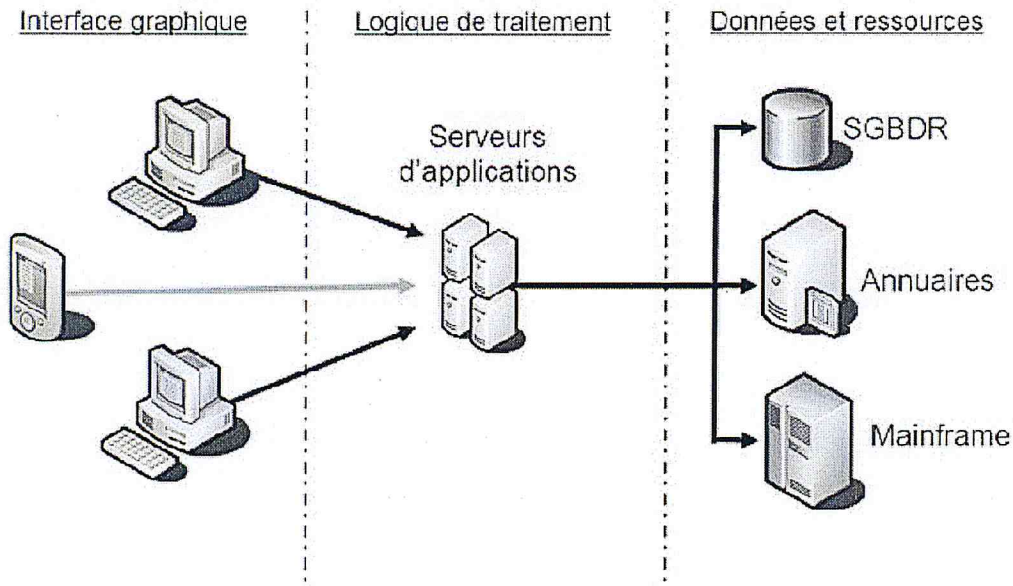

Figure S1.II.1 Architecture 3/tiers [LANxx].

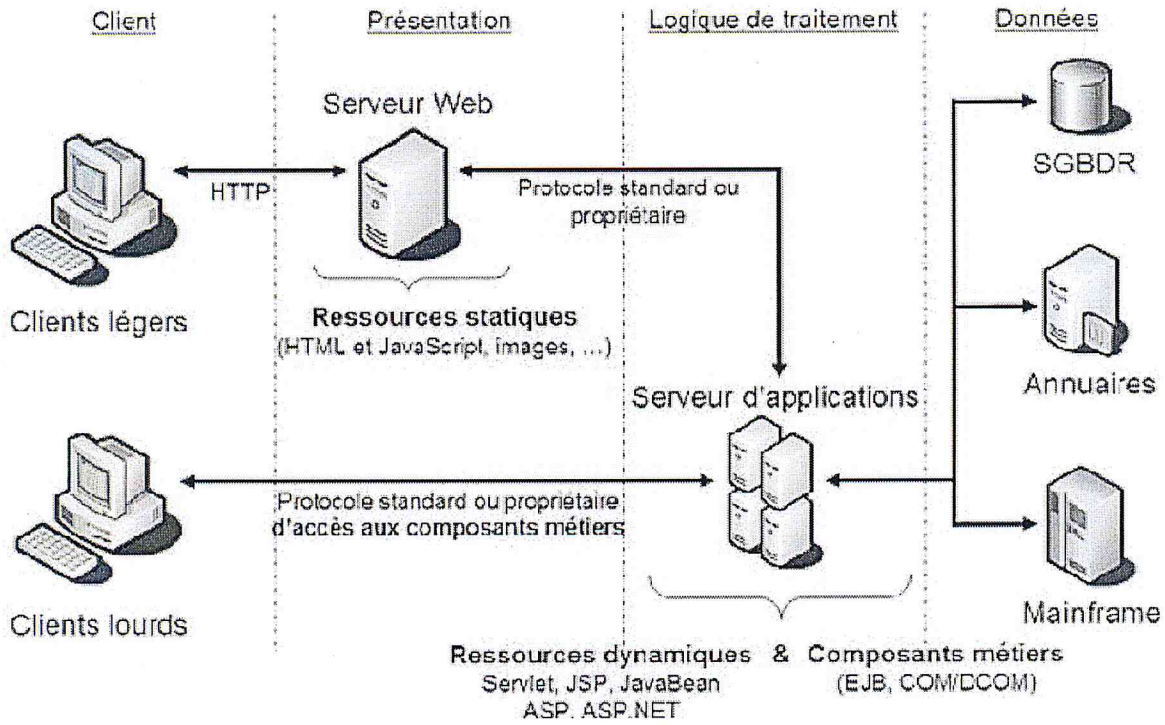

Figure S2.II.2 Architecture n/tiers [LANxx].

Dans les applications web, la répartition se fait en : client, serveur d'application, serveurs web et serveur de bases de données.

- **Le client [WIKI]:** Un client (l'ordinateur demandeur de ressources) équipé d'une interface utilisateur (généralement un navigateur web) chargé de la présentation.. Il est encore appelé client léger. Il a pour rôle d'afficher correctement du code XHTML et d'interpréter du code script.

  JAVA Script [LANxx]: Langage de programmation de scripts principalement utilisé pour les pages WEB interactives. Il s'agit d'un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas créés au sein de classes mais sont tous équipés de constructeurs permettant de générer leurs propriétés, et en particulier une propriété de prototypage destinée à générer des objets héritiers personnalisés.

  Utilisation :
  Du code JavaScript peut être intégré directement au sein des pages Web, pour y être exécuté sur le poste client. C'est alors le navigateur Web qui prend en charge l'exécution de ces programmes appelés scripts. Généralement, JavaScript sert à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML via l'interface Document Object Model, fournie par le navigateur (on parle alors parfois de HTML dynamique ou DHTML). Il est aussi utilisé pour réaliser des services dynamiques, parfois futiles, strictement cosmétiques ou à des fins ergonomiques. JavaScript n'est pas limité à la manipulation de documents HTML et peut aussi servir à manipuler des documents SVG, XUL et autres dialectes XML.

- **Le serveur d'application [WIKI]:** Serveur assurant la médiation entre un logiciel client et un serveur de base de données. Le serveur d'application prend place comme intermédiaire entre le logiciel client, qui gère uniquement l'interface utilisateur, et le logiciel serveur de base de données. Il prend en charge toute la logique de l'application (logique métier). Dans notre application nous avons utilisé JSP (Java Server Pages).

  JSP [LANxx] : Une page JSP est constituée d'un squelette de code HTML pour la partie statique, et de code Java pour permettre l'intégration de données obtenues en résultat des traitements. La structure d'une JSP est donc hétérogène, Le mélange d'HTML et de code Java dans les pages JSP permet de séparer la présentation (en HTML) des aspects procéduraux contenus dans le code. On a ainsi une grande souplesse dans le développement de sites web.la page JSP devra être traitée par le serveur d'application avant d'être envoyée dans la réponse client. Ce traitement est réalisé par le serveur d'application au premier appel de la page et à chaque fois que cette page est modifiée par un programmeur. Un serveur d'application transforme une JSP en classe Java, puis la compile en servlet, c'est ensuite l'exécution de cette servlet particulière qui provoquera la génération de la réponse. Chaque autre requête vers cette JSP est en fait directement prise en charge par la servlet générée. C'est cette étape particulière du cycle de vie d'une JSP qui fait qu'un serveur d'application JEE a besoin d'un compilateur Java, et que par conséquent, une grande majorité d'entre eux nécessite une installation de java fournie par JDK plutôt que par un JRE.

- **Le serveur web [WIKI]:** est un logiciel paramétré pour pouvoir traiter tout types de pages et notamment celles qui contiennent des instructions de programmation. Il reconnaît ces pages grâce à l'URL qu'il reçoit, effectue les traitements demandés et transmet le résultat au format html au client. Un serveur web est donc un simple logiciel capable d'interpréter les requêtes http arrivant sur le port associé au protocole HTTP (par défaut le port 80), L'exécution d'une page JSP est dépend a un conteneur, ce conteneur est appelé Appache TomCat.

  Appache TomCat [LANxx]: est un conteneur libre de servlets et JSP Java EE. Issu du projet Jakarta, Tomcat est un projet principal de la fondation Apache. Tomcat implémente les spécifications des servlets et des JSP du Java Community Process . Il est paramétrable par des fichiers XML et de propriétés, et inclut des outils pour la configuration et la gestion. Il comporte également un serveur HTTP.

- **Serveur données [WIKI]:** est un logiciel qui va héberger la base de données (SGBD, système de gestion de bases de données). Pour notre part, nous avons choisi MySql (en version 5.1.36)

  MySQL [Thixx]: est un système de gestion de base de données relationnelles(SGBDR) rapide, robuste et facile d'utilisation.il est adapté à la gestion de données dans environnement réseau, notamment en architecture client/serveur. Il est fourni avec de nombreux outils et est compatible avec de nombreux langages de programmation.il est le plus célèbre SGBDR du monde Open Source. Ce serveur de base de données est interrogeable via SQL (Strured Query Language), le standardisé

le plus populaire pour interroger les bases de données. Le SQL permet de manipuler les données très facilement.

Architecture physique [WIKI]:

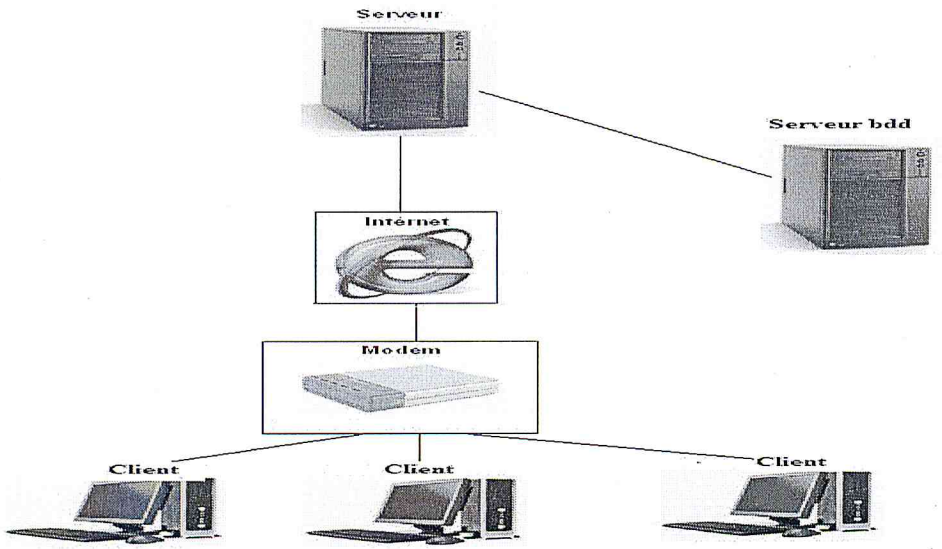

Figure S1.II.3 Schéma de l'architecture physique [WIKI].

- **Le client :** est ordinateur ou un Pocket Pc disposants d'un simple navigateur web pour accéder au site.
- **La Machine serveur :** hébergera le site, le serveur web (Apache Tomcat 6), mais aussi toutes les ressources postées sur le site (documents, videos, etc.).
- **La machine serveur de base de données :** Le serveur de base de données est la machine d'implémentation de la base de données. A cet effet, elle sera dotée du SGBD MySql 5.1.36

Sécurité du système :

Theodoros Evdoridis **[Tho08]** a identifié 3 types de menaces qui peuvent planer sur la sécurité d'un portail web :

Les Menaces au niveau du réseau : se présentent sous la forme de méthodes mises au point par des personnes malveillantes, qui utilisent un couple matériel/logiciel spécifique pour prendre possession de mots de passes et de sessions utilisateurs. le Sniffing ou le Spoofing attack sont 2 méthodes très utilisées

Les menaces au niveau du serveur : si le serveur n'est pas bien protégé contre le virus, l'intégrité du portail est menacée, surtout au niveau de la base de données.

Les menaces dues à l'architecture du portail : le fait d'utiliser une architecture 3 tiers, les logiciels malveillants ont 3 points d'accès au système (client, serveur web, serveur bdd) se qui augmente la vulnérabilité du portail.

Nous pouvons citer la technique la plus utilisée qui est l'Injections SQL ,qui est un type d'exploitation d'une faille d'une application web, injectant une requête SQL non prévue par le système et pouvant compromettre sa sécurité.

Les injections se font à travers les champs de saisie offerts à l'utilisateur, dans le cas où les chaines de caractères entrées par l'utilisateur ne sont pas traitées.

Cross-site scripting , est un autre type d'exploitation de faille des sites web dont le principe consiste, cette fois, à injecter du script (souvent de type javascript), dans les pages vulnérables, en utilisant la plupart du temps des paramètres d'URL ou des messages postés. Si ces données (paramètres d'URL, etc.) arrivent, telles quelles, dans la page web transmise au navigateur, le script malveillant va s'exécuter et va accomplir des actions indésirables.

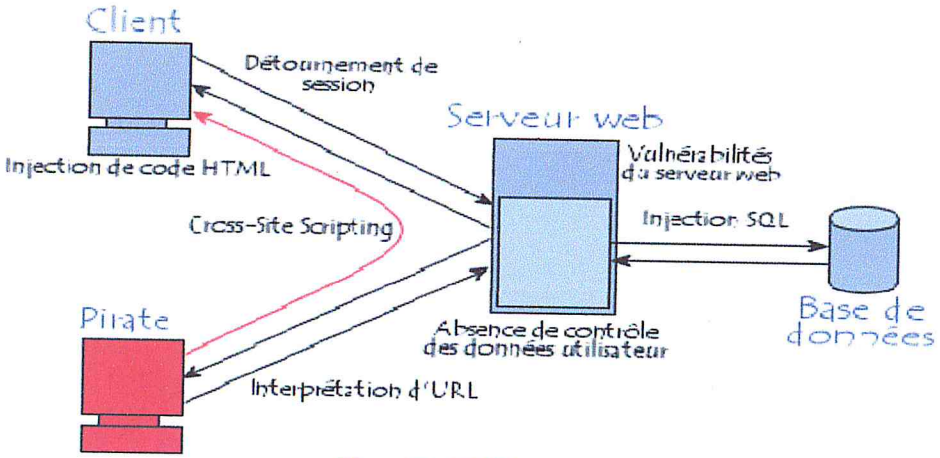

Figure S1.II.4 Risques encourus par un site web.

- **Authentification des utilisateurs :**

Pour pouvoir garantir les privilèges d'accès instaurés aux ressources et aux fonctionnalités, les utilisateurs doivent être authentifiés sur le site. Le procédé d'authentification que nous avons choisi est le procédé le plus utilisé pour l'application web, l'authentification se fait par pseudo/mot de passe. Dés que l'administrateur valide l'inscription, le couple (pseudo/mot de passe) est enregistré dans la base de données (le mot de passe est crypté), et un identifiant est attribué à l'utilisateur.

2. Déploiement :

Interface Principal :

Figure S1.II.5 Page index

- Ceci est la fenêtre principale de notre site web, elle permet d'accéder dynamiquement a toutes les autres pages.

Interface d'inscription :

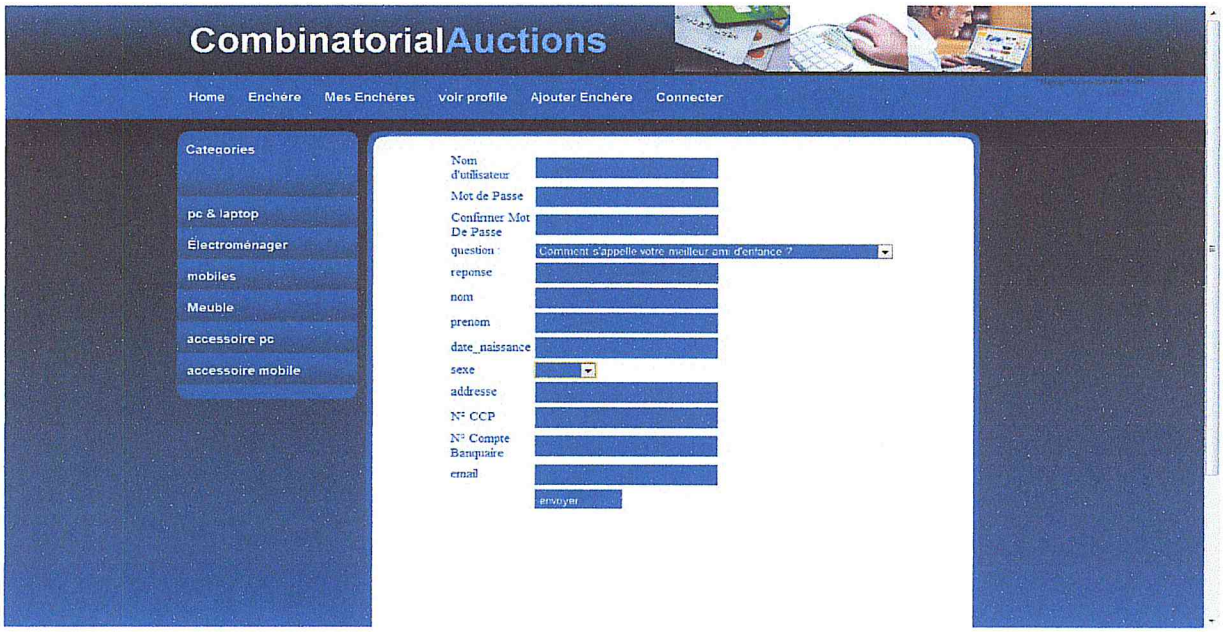

Figure S1.II.6 Page d'inscription.

- *Cette figure nous montre l'interface d'inscription, ou l'utilisateur au moment de s'inscrire il rempli les champs d'information personnelles le concernant pour s'enregistré a notre base* de données.

Interface d'authentification :

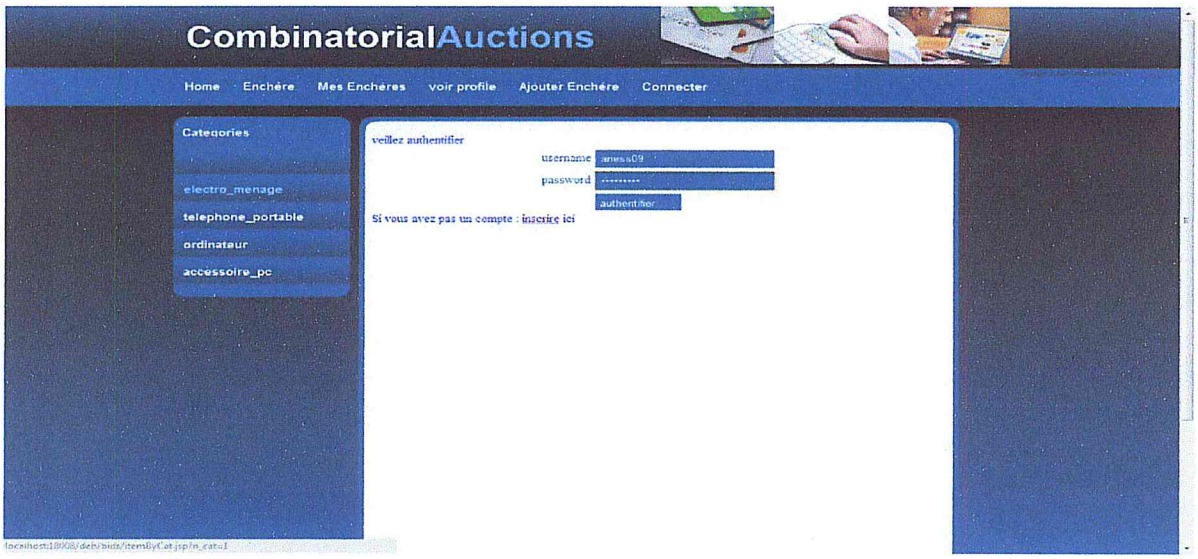

Figure S1.II.7 Page d'authentification.

- Cette interface permet au utilisateur de se connecté au site avec son nom d'utilisateur et son mot de passe. Afin d'accédé a son profil.

Interface d'ajout des items :

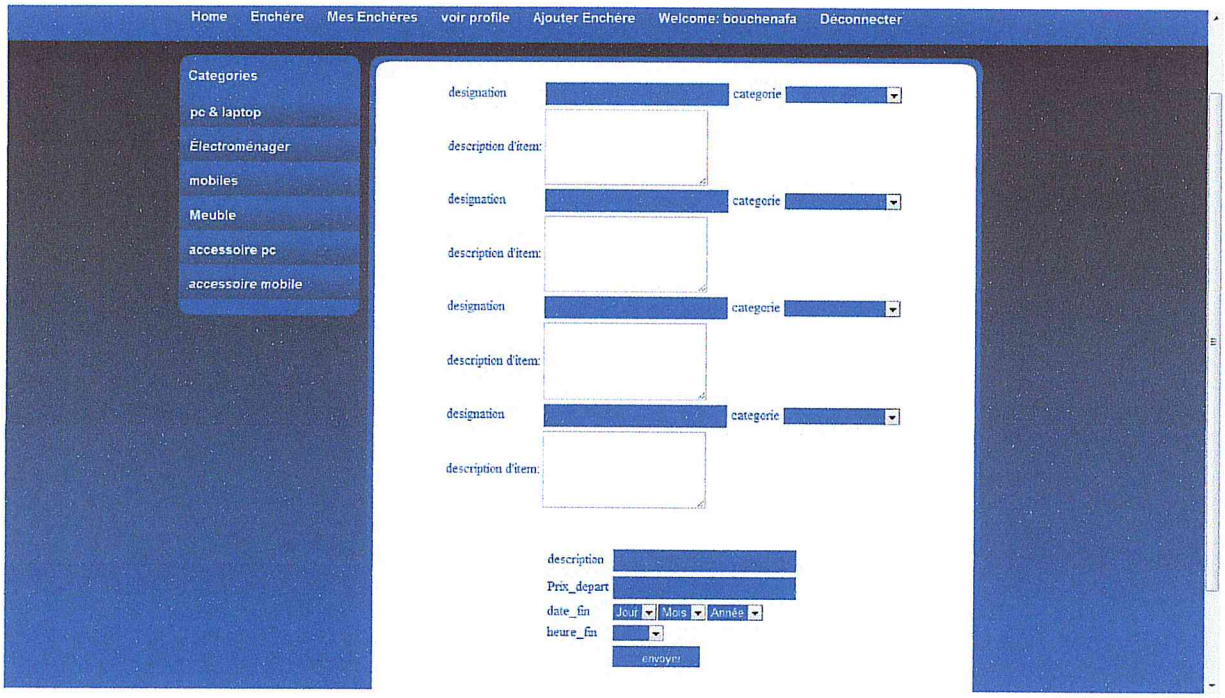

Figure S1.II.8 Page d'ajout d'items.

❖ Dans cette page l'utilisateur doit spécifier le nombre d'items à ajouter avec les informations nécessaires :

- Désignation d'item.
- Catégorie d'item.
- Une description sur l'item (état, couleur, poids,……etc.).
- Description sur l'enchère parce que ce champ est nécessaire pour lister les enchères.
- Date de fin d'enchère et heure de fin.

L'interface Enchérir :

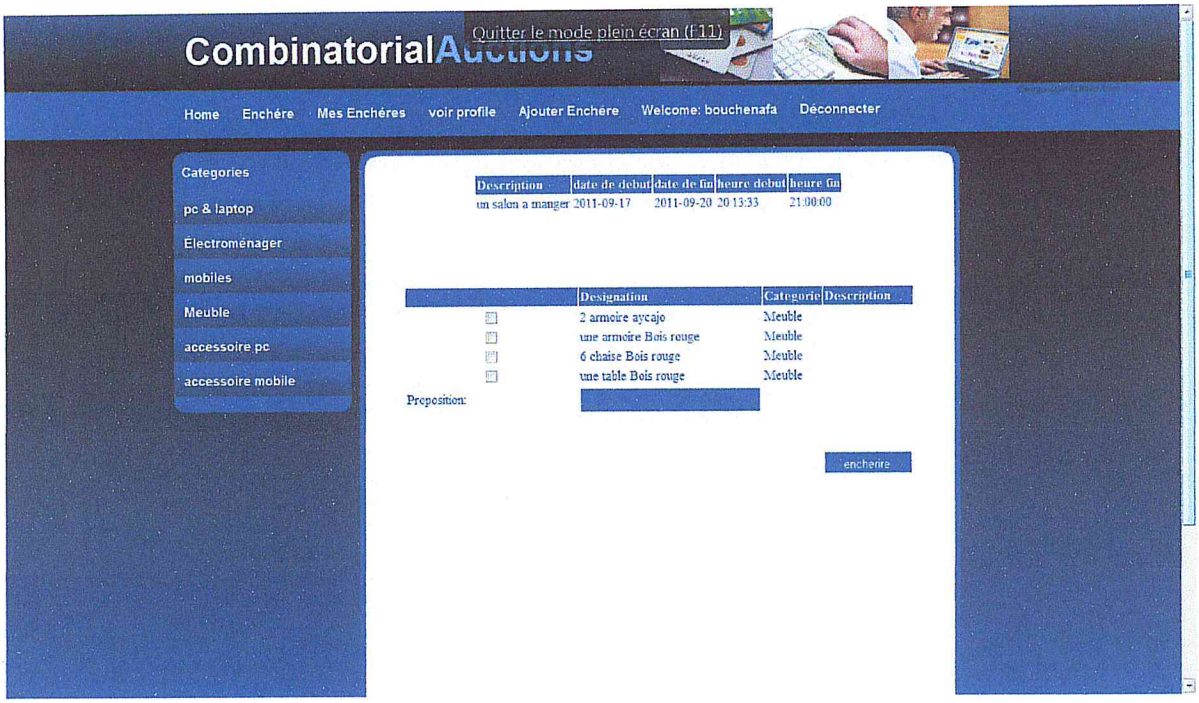

Figure S1.II.9 Page Enchérir.

- Dans cette page l'acheteur doit cocher les items qui les intéressent, avant de donner une proposition sur l'ensemble choisi.

Interface proposition :

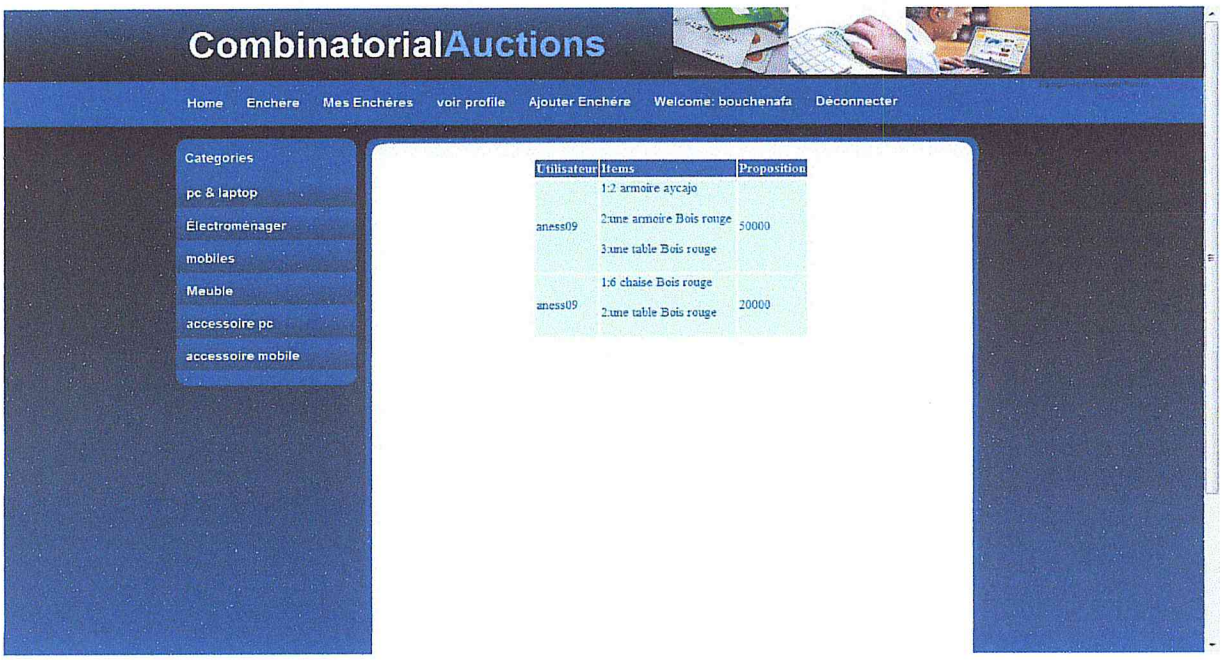

Figure S1.II.10 Page proposition

- C'est la page proposition ou le vendeur et l'acheteur peuvent voir les propositions données dans une enchère.

Interface liste d'enchère :

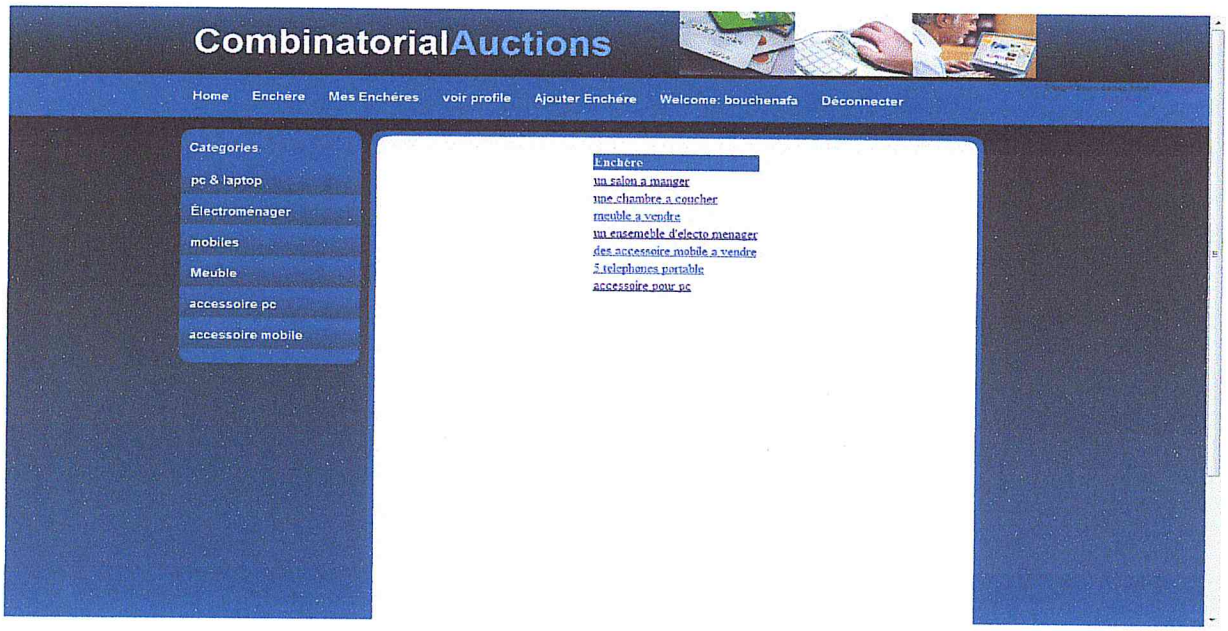

Figure S1.II.11 Page liste d'enchères.

- Dans la page liste, l'acheteur peut voir la liste de toutes les enchères en cour (non clôturées).

Interface liste de mes enchères :

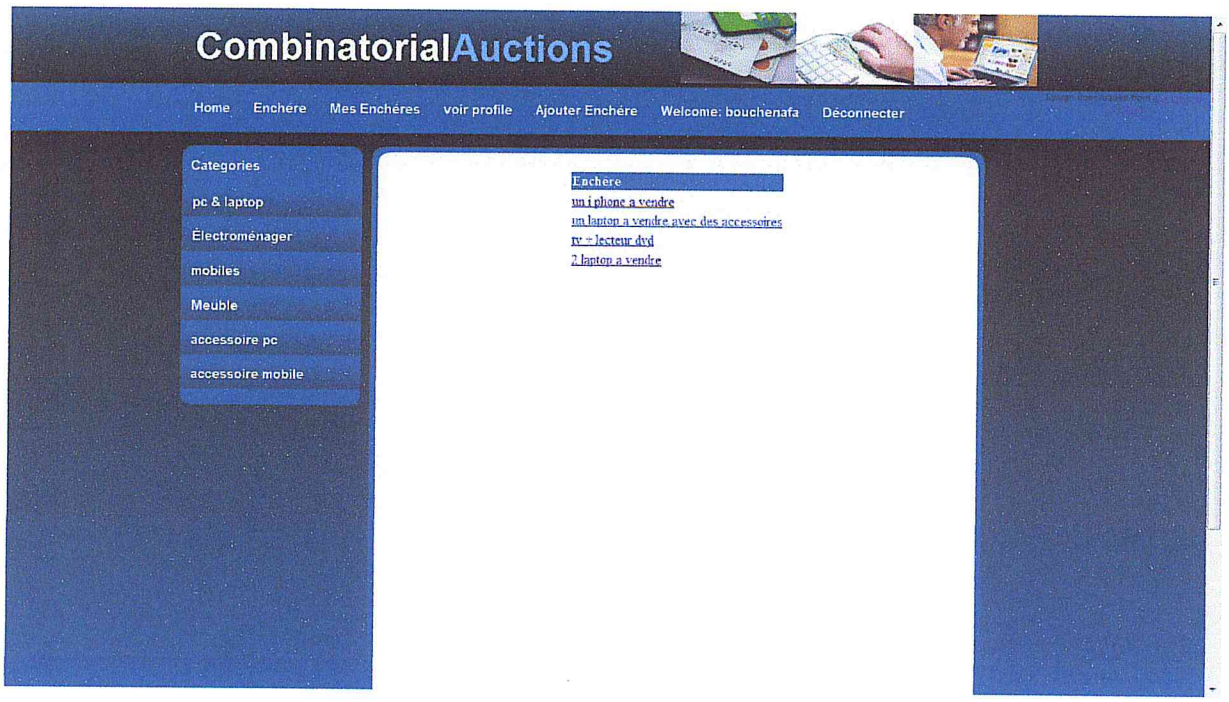

Figure S1.II.12 Page liste de mes enchères.

- Cette page est désignée pour les vendeurs, afin de voir leurs propres enchères. Un click sur une enchère la page proposition apparait, pour voir les propositions dans cette enchère.

Interface items par catégorie :

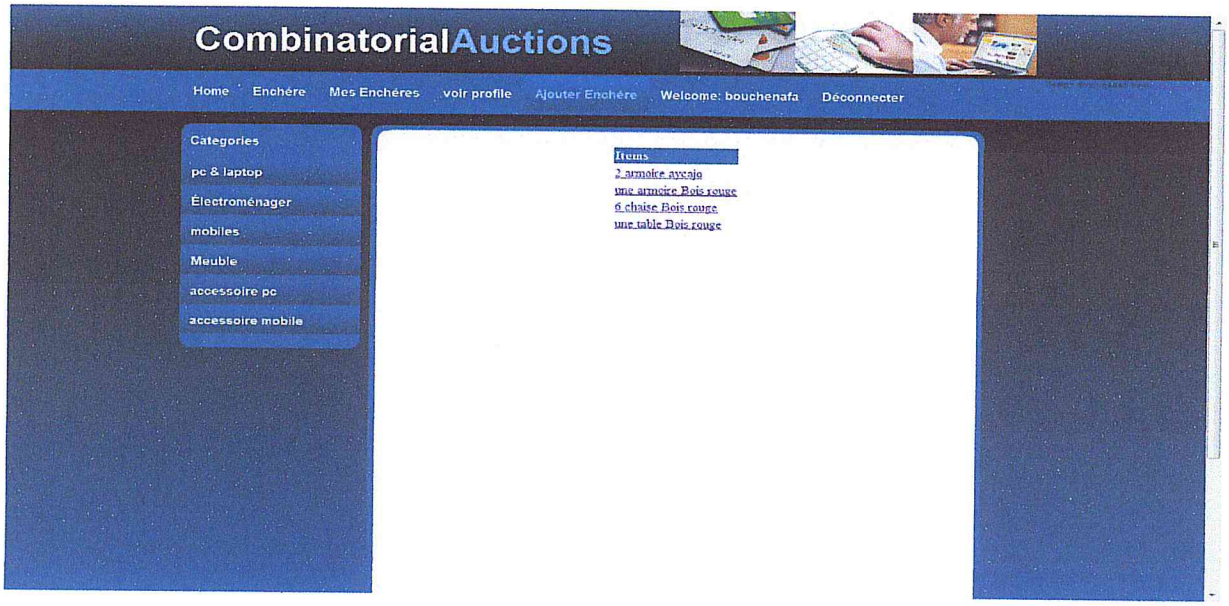

Figure S1.II.13 Page item par catégories.

- Cette page sert à lister les items par leur catégorie c.-à-d. elle ramène tous les items qui appartiennent à une catégorie donnée comme paramètre.

La liste des catégories est affiché dans le coté gauche de la page et dans toutes les autres pages du site.

Interface profil utilisateur :

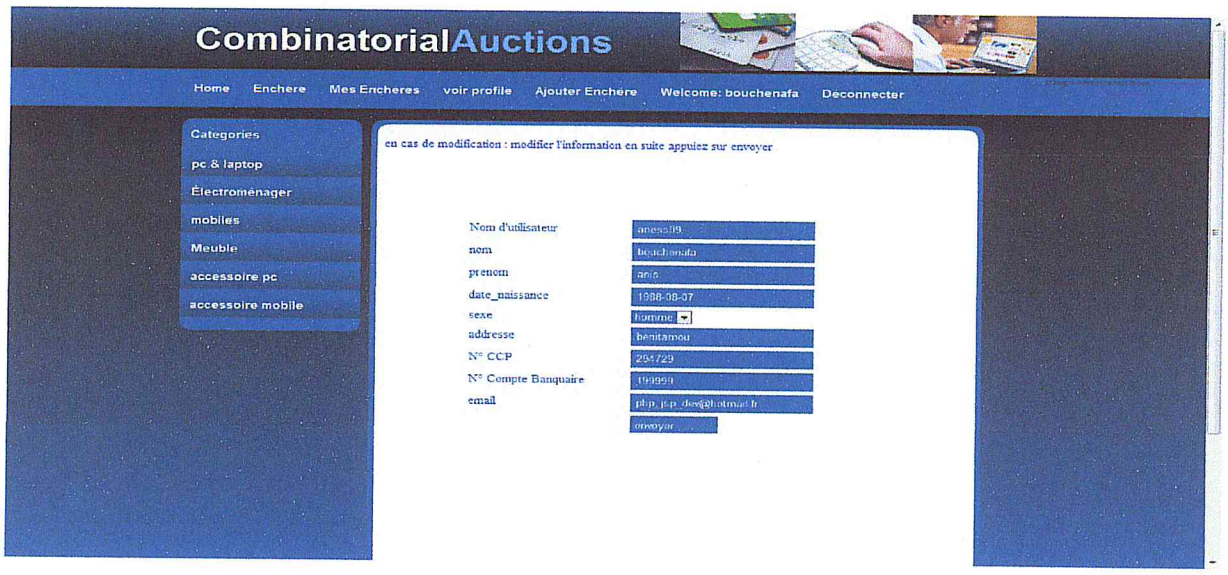

Figure S1.II.14 Page profil utilisateur.

- C'est le profil de l'utilisateur, elle contient les informations sur un utilisateur donné. Si l'utilisateur est connecté, il click sur Voir Profile pour voir ses information. Maintenant s'il veut modifier une information il, n'aura qu'à la modifier après un click sur envoyer.

3. Conclusion :

Dans ce chapitre on a parlé sur l'algorithme de résolution et ces phases en suite sur notre diagramme de classe d'implémentation sa et son description détaillé. Suite à ça nous nous avons défini les outils qu'on utilisé dans l'implémentation. Et on a terminé avec un aperçu de notre site web.

Chapitre 5 :
Tests et résultats

1. Introduction :

Dans ce chapitre des tests seront effectué afin d'évaluer d'une part les performances de notre application, et d'autre part afin d'examiner son comportement face à la variation des instances. Les instances de tests utilisées ont était générer en utilisant le logiciel CATS (Combinatorial Auctions Test Suite) qui est devenue la référence standard pour évaluer les algorithmes dédié à la résolution du problème de détermination du gagnant « WDP » dans les enchères combinatoires.
Tous les tests ont été effectués sur une machine doté d'un processeur Intel® Core™2 Duo CPU T5870 @2.00 GHZ, 2GHZ avec une mémoire vive de 2 Go, sous un environnement Linux (Ubuntu 10.04).

2. Instances de tests [BNS09]:

Pour conduire ces tests on a généré des instances en variant parallèlement le nombre d'articles et d'offres pour plusieurs distributions.nous donnons une description pour chacune des distributions utilisées.

Legacy Data Distributions : Avec la première vague d'algorithmes pour le problème de détermination du gagnant WDP, plusieurs distributions ont été proposées [Sandholm 1999; Fujishima et al. 1999;Hoos et Boutilier 2000]. Ces distributions ont été largement utilisées par d'autres chercheurs, y compris beaucoup de ceux cités ci-dessus. Chacune de ces distributions peut être vue comme une réponse à deux questions: quel est le nombre d'articles couvert par une offre, et quel prix proposer pour un paquet. Étant donné un nombre requis d'articles, toutes les distributions sélectionnent les articles à inclure dans une offre de façon aléatoire uniforme sans remplacement. Nous donnons ici les noms utilisés pour les identifier comme des distributions "legacy" Leyton-Brown et al. [2000].

A cause de la nature de certaine distribution la génération de l'ensemble de tests prend un temps extrêmement long c'est pourquoi on s'est contenté des 4 distributions de type Legacy suivantes :

- L2 : la distribution aléatoire pondérée proposée par Sandholm [1999] choisit un certain nombre d'articles g uniformément aléatoire entre [1, m] Et lui attribue un prix tiré de manière uniforme dans [0, g].

- L4 : la distribution Decay Sandholm [1999] commence avec une taille de paquet de 1, et incrémente les dimensions du paquet jusqu'à ce qu'un tirage aléatoire uniforme dans [0, 1] dépasse un paramètre α.

- L6 : la distribution exponentielle de Fujishima et al. [1999], les demandes de produits, avec une probabilité de $C\, e^{-g/q}$ et attribue un prix a l'offre établit de manière aléatoire uniforme dans
[0.5g, 1.5g].

- L7 : la distribution binomiale de Fujishima et al. [1999], donne à chaque article une probabilité indépendante de p d'être inclus dans un paquet, et attribue un prix a l'offre établit de manière aléatoire uniforme dans [0,5 g, 1,5 g] où g est le nombre de produits sélectionnés.

Les distributions ci-dessus ont été critiquées à plusieurs reprises pour leurs manques de signification économique. Pour cette raison CATS essai de générer un ensemble réaliste de distributions de tests.

Un graphe est généré représentant des relations d'adjacence entre les biens, et est utilisée pour dériver les propriétés de complémentarité entre les biens et les propriétés de substituabilité des offres. Dans le monde réel la complémentarité concernent certaines situations basées sur la contiguïté dans l'espace (physiques ou conceptuelles), tandis que la complémentarité est basée sur le temps de corrélation. Les distributions utilisées dans les tests sont décrite ci dessous.

paths : modélise une vente aux enchères des liens de transport entre les villes ou la vente aux enchères de bande passante, ou plus généralement des arêtes dans un graphe planaire près. Les offres concernent les ensembles d'articles qui correspondent aux chemins entre des paires de nœuds choisis aléatoirement. Les offres substituables sont ceux qui relient les mêmes paires de nœuds, les prix dépendent de la longueur du chemin.

regions : modélise une vente aux enchères de biens immobiliers, ou plus généralement de tous les biens sur lesquels l'adjacence est la base de la complémentarité. Les offres concernent les biens qui sont adjacents dans un graphe planaire.

arbitrary : est similaire à regions, mais relaxe l'hypothèse de planéité et modélise les complémentarités entre des produits discrets tels que des pièces d'électronique ou objets de collection.

scheduling : modélise la planification d'un domaine d'ordonnancement distribué job-shop et aussi son application aux ventes aux enchères de production d'électricité.

Dans ce qui suit nous retranscrivons les résultats obtenus. Pour chaque distribution un histogramme décrit les temps d'exécution obtenus selon les variations du nombre d'articles et d'offres. Pour chaque couple de valeurs Item/Bid nous avons pris la moyenne des temps d'exécution pou 5 instances.

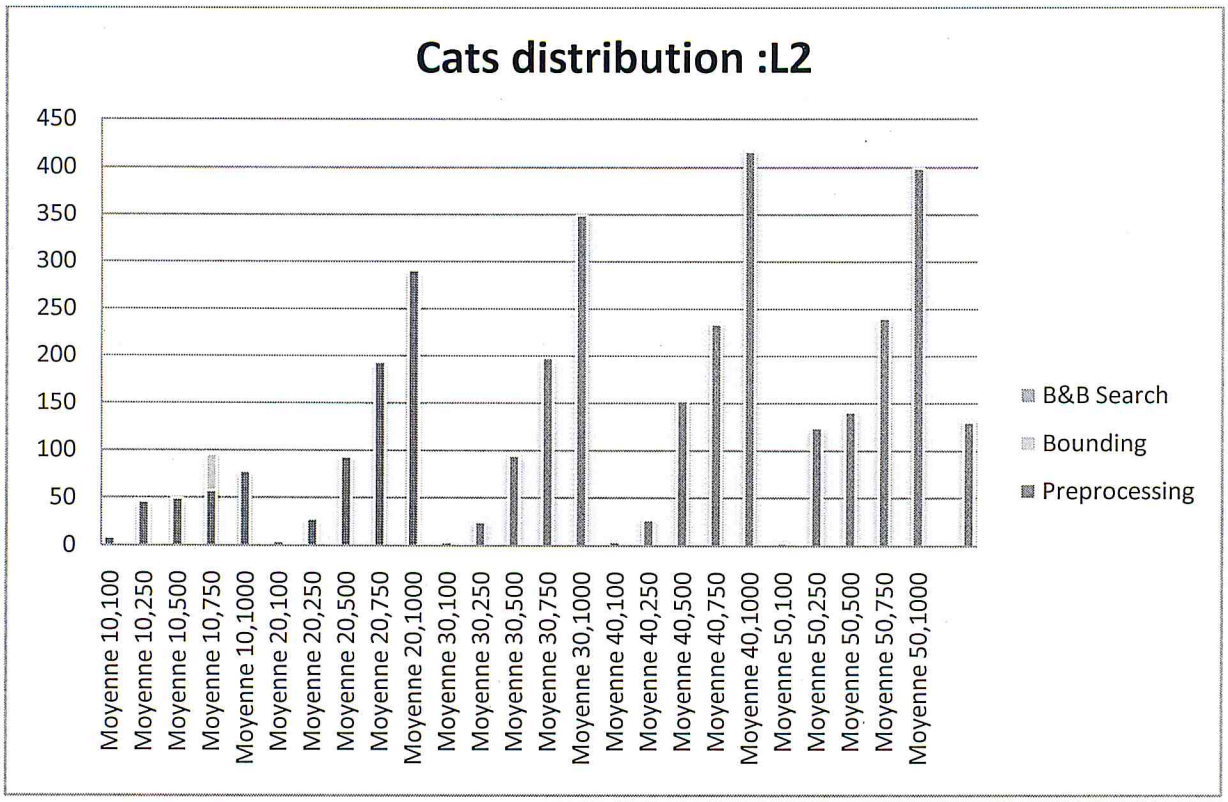

Figure 5.2.1

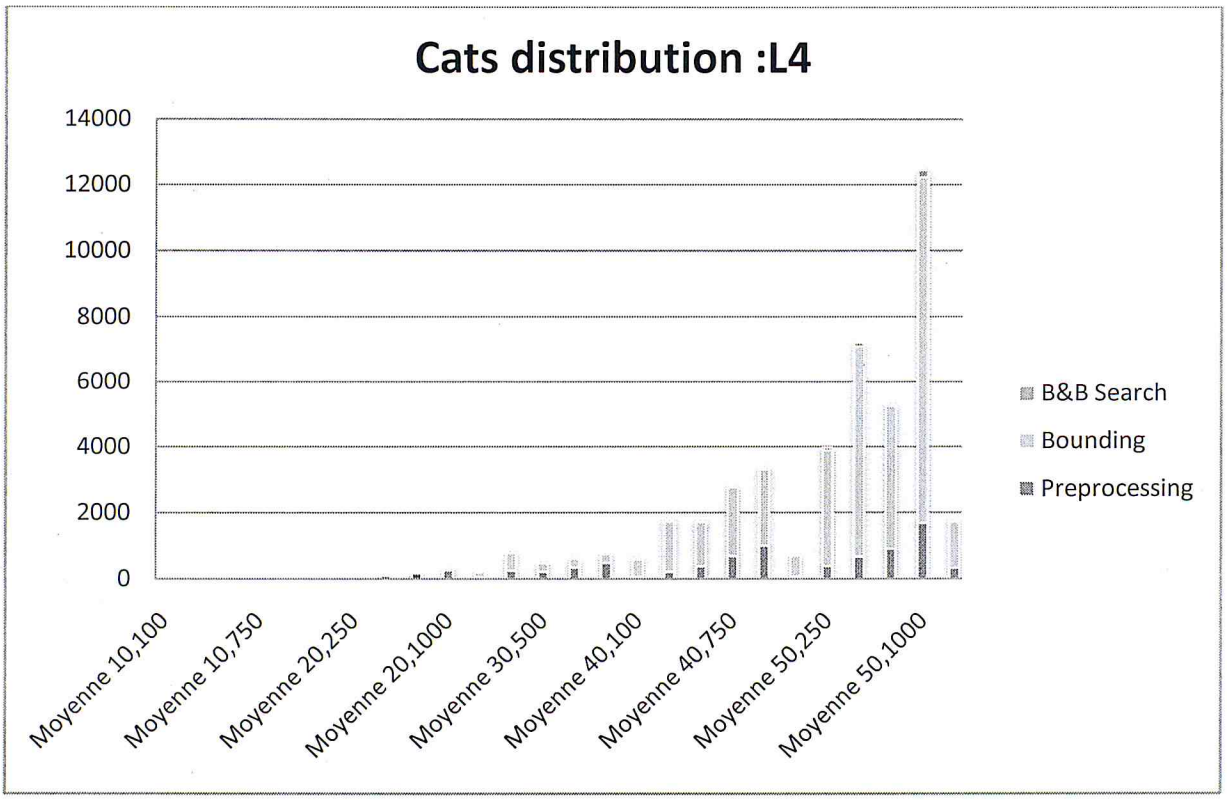

Figure 5.2.2

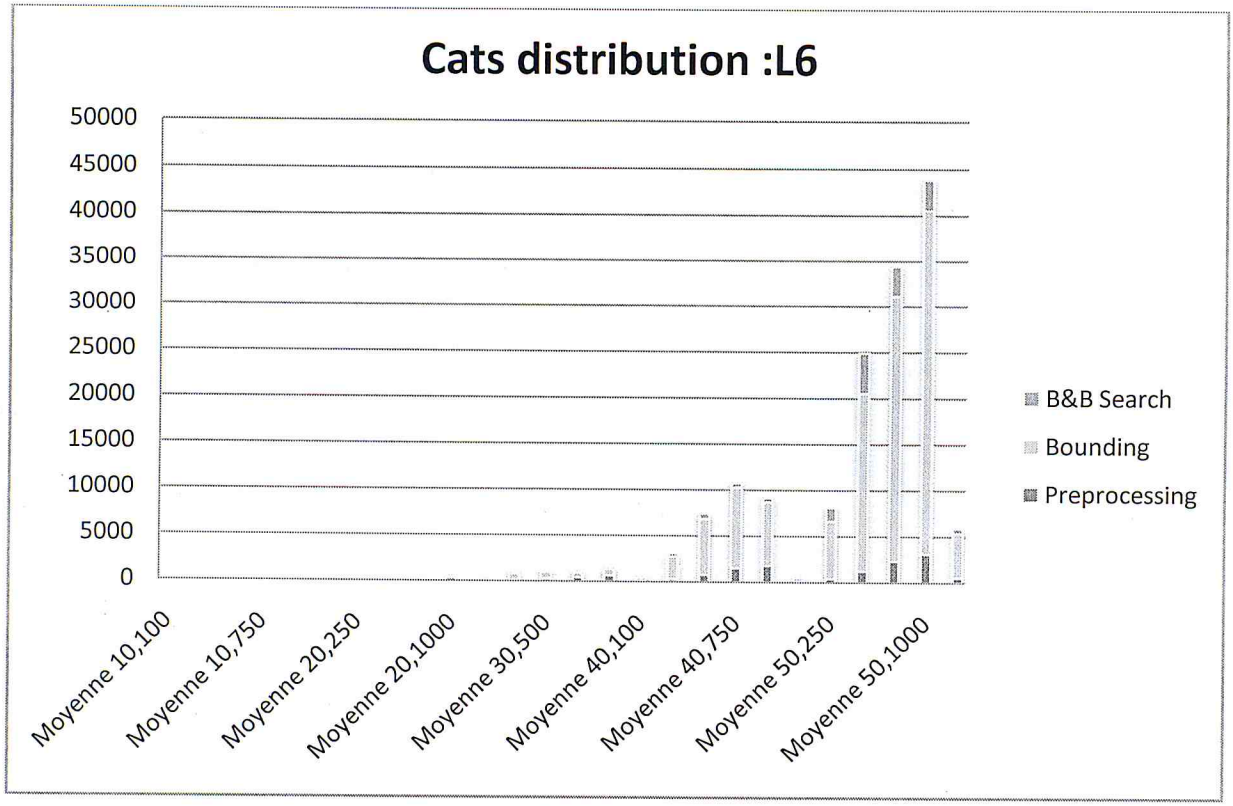

Figure 5.2.3

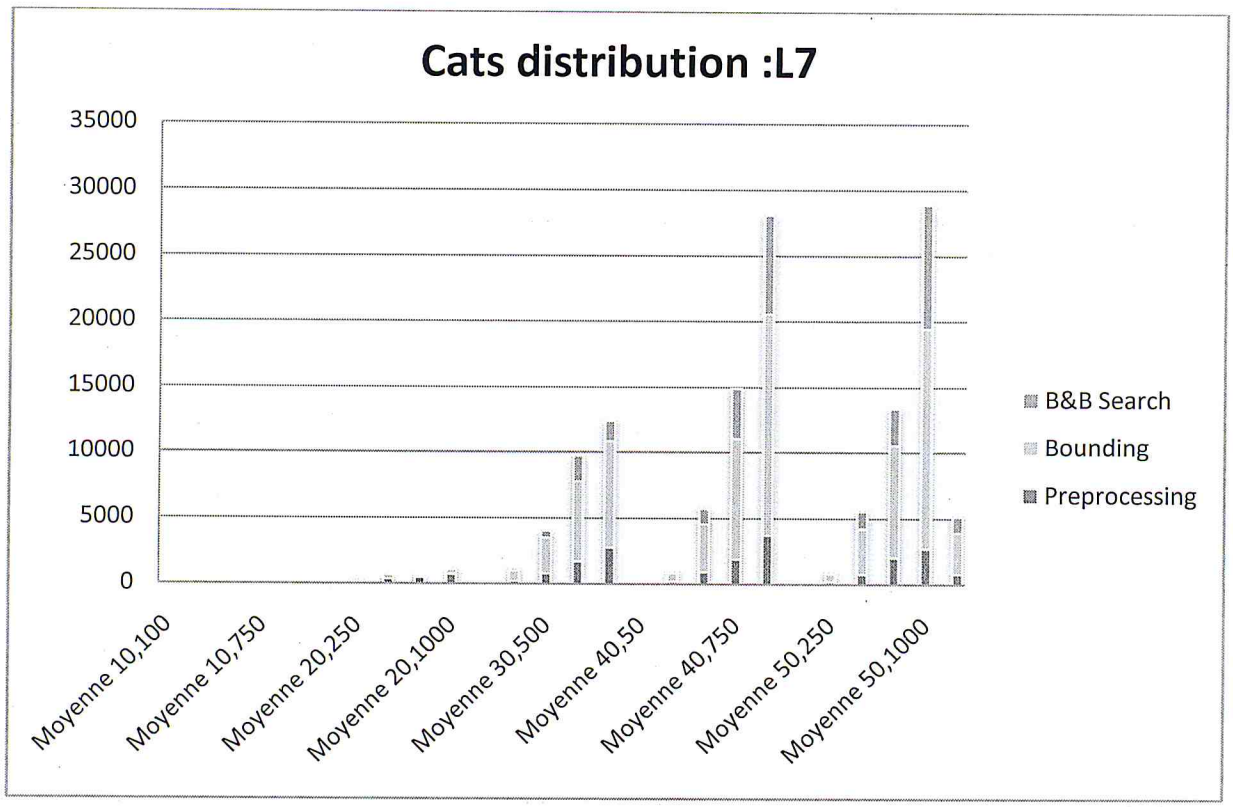

Figure 5.2.4

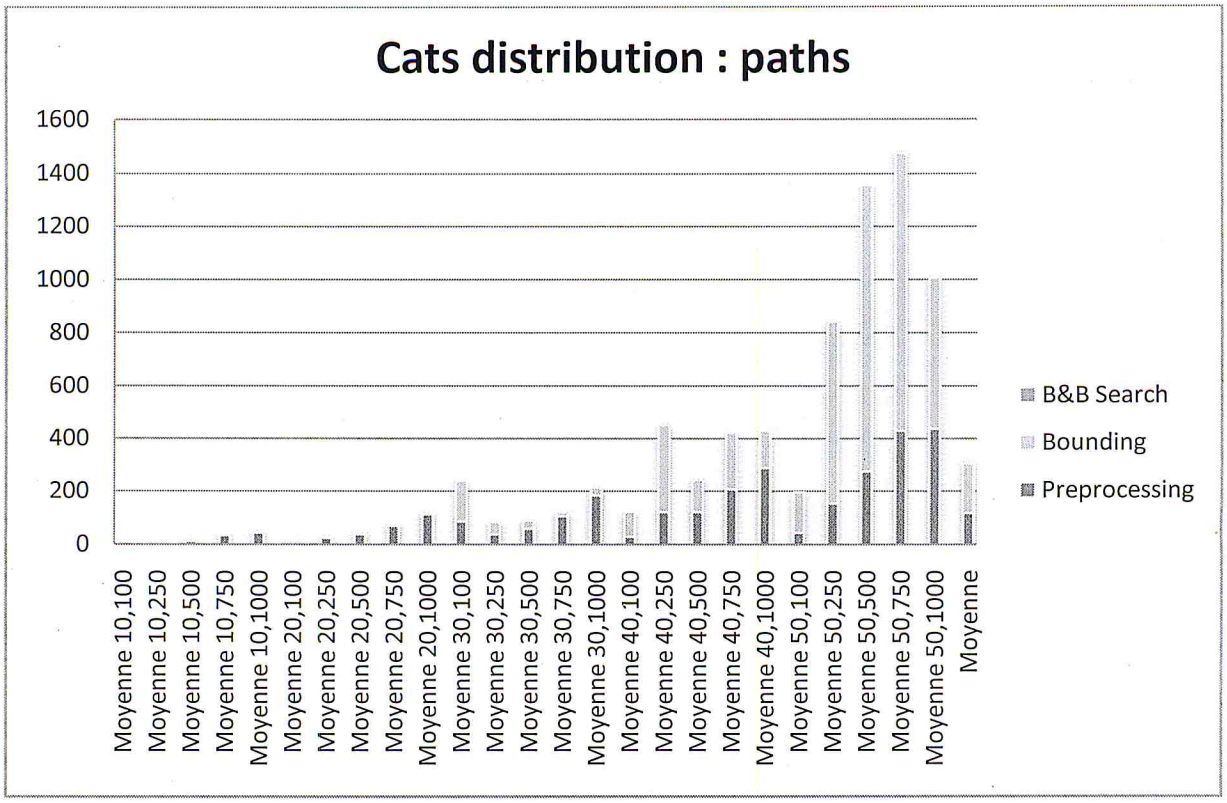

Figure 5.2.5

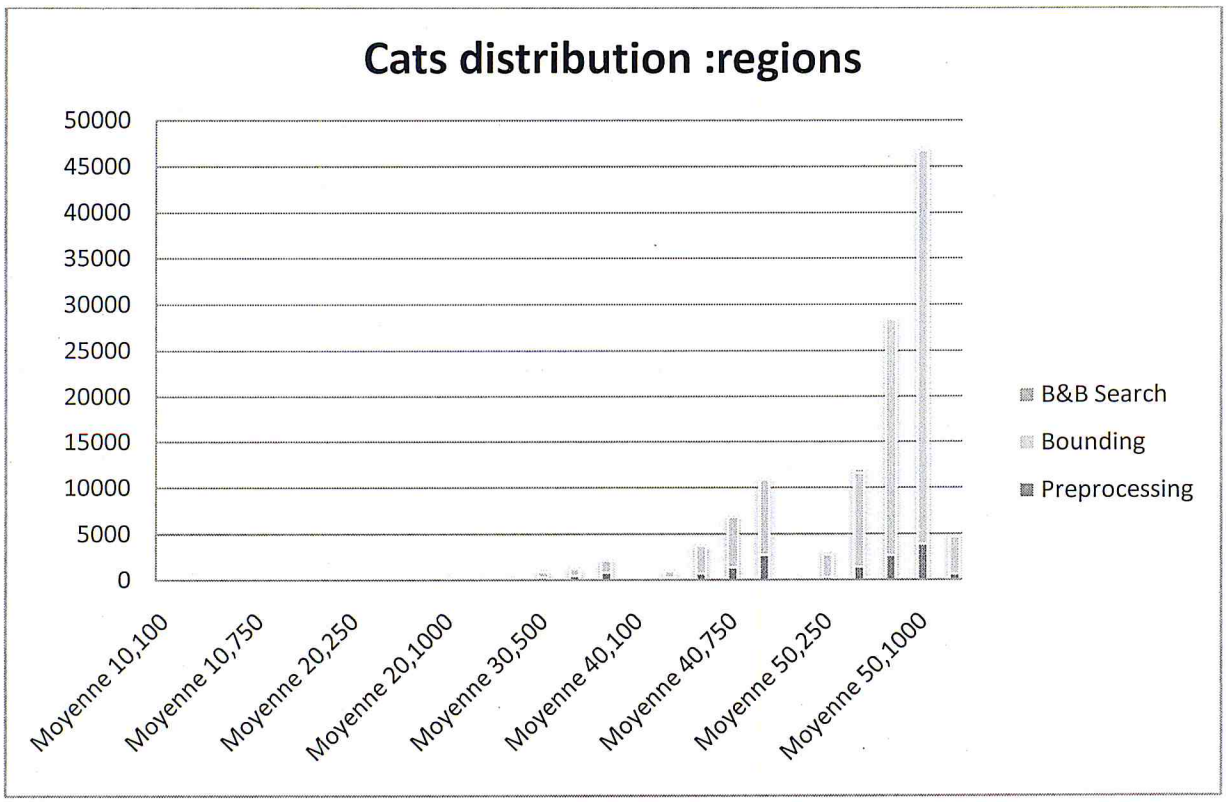

Figure 5.2.6

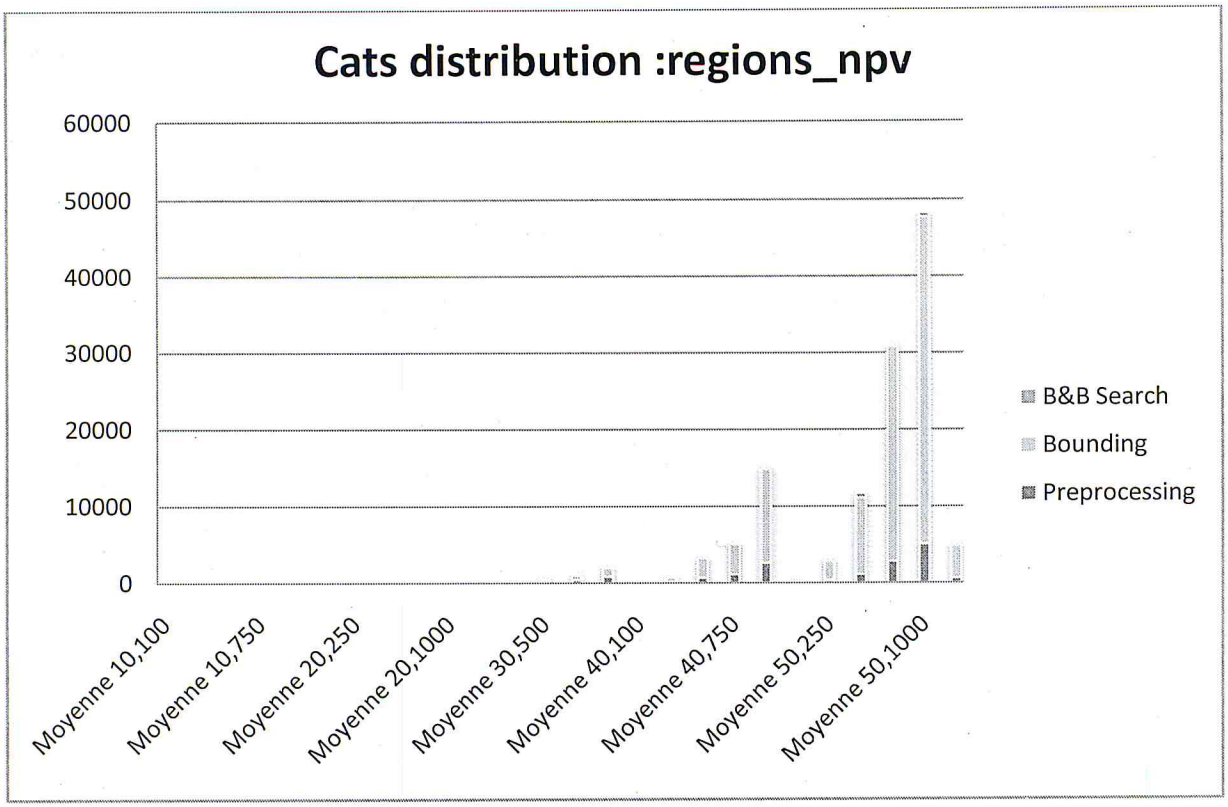

Figure 5.2.7

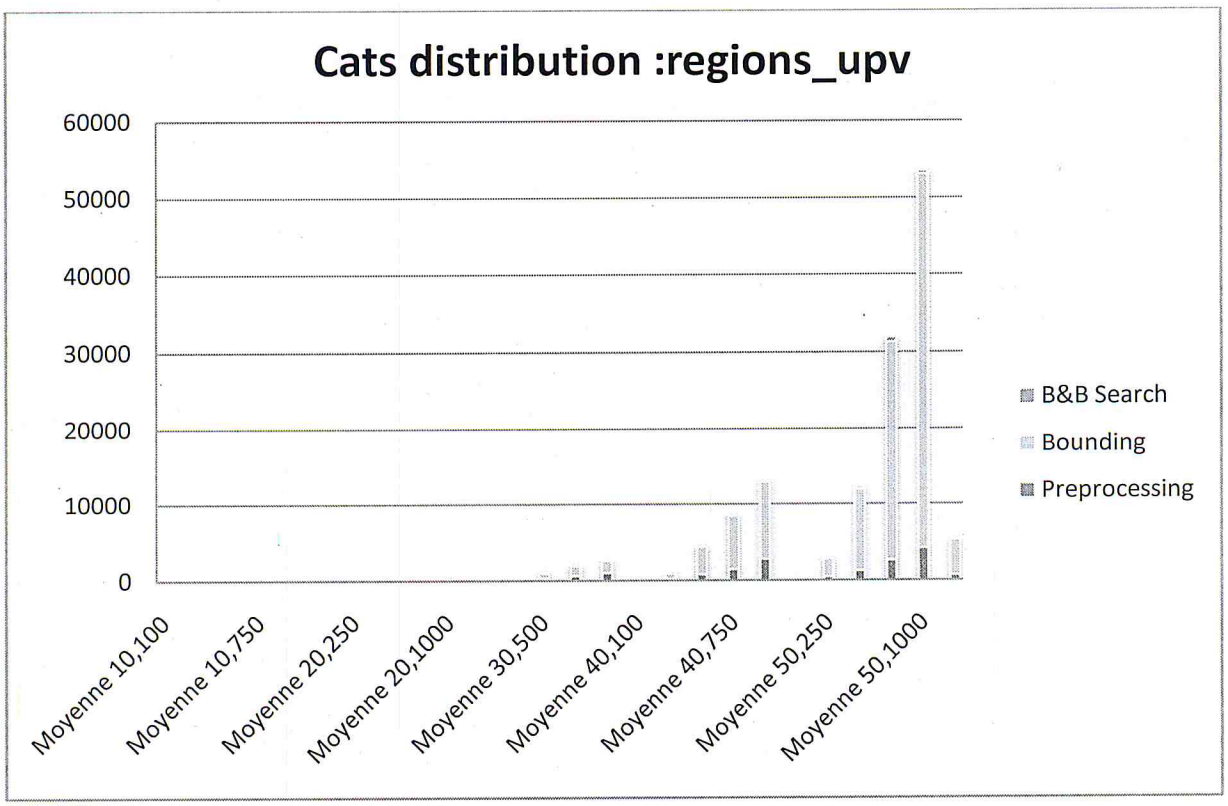

Figure 5.2.8

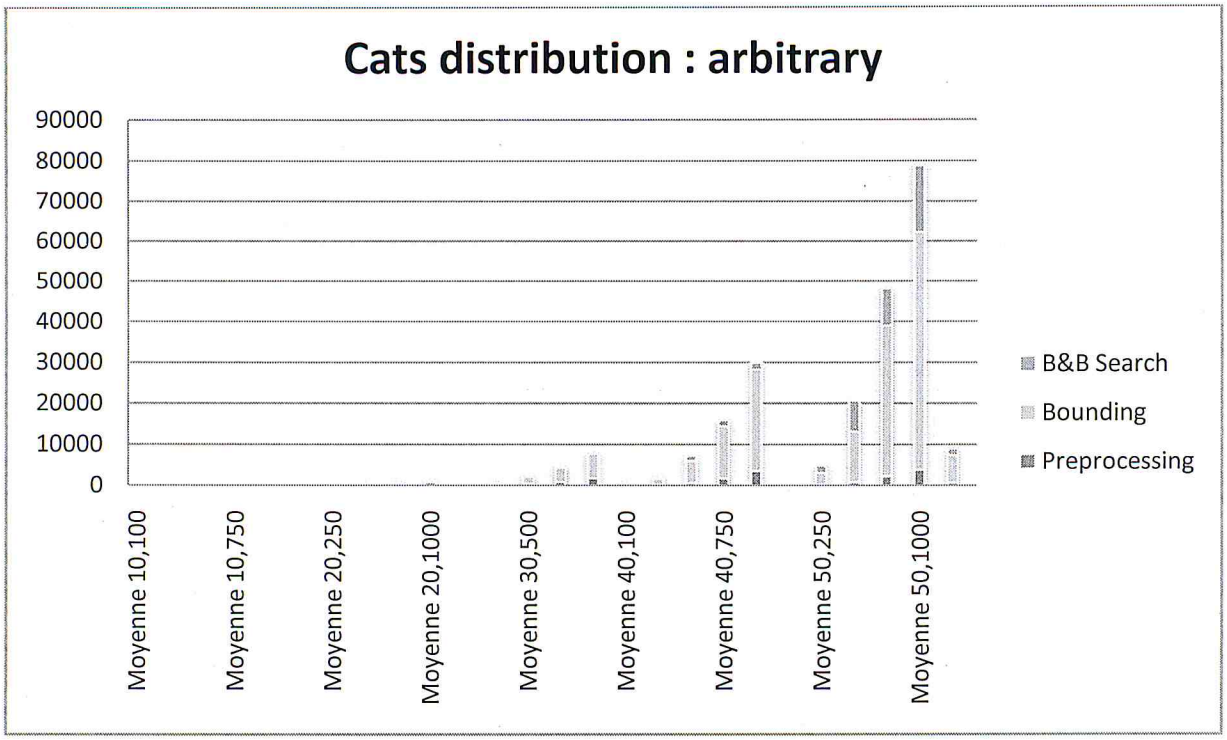

Figure 5.2.9

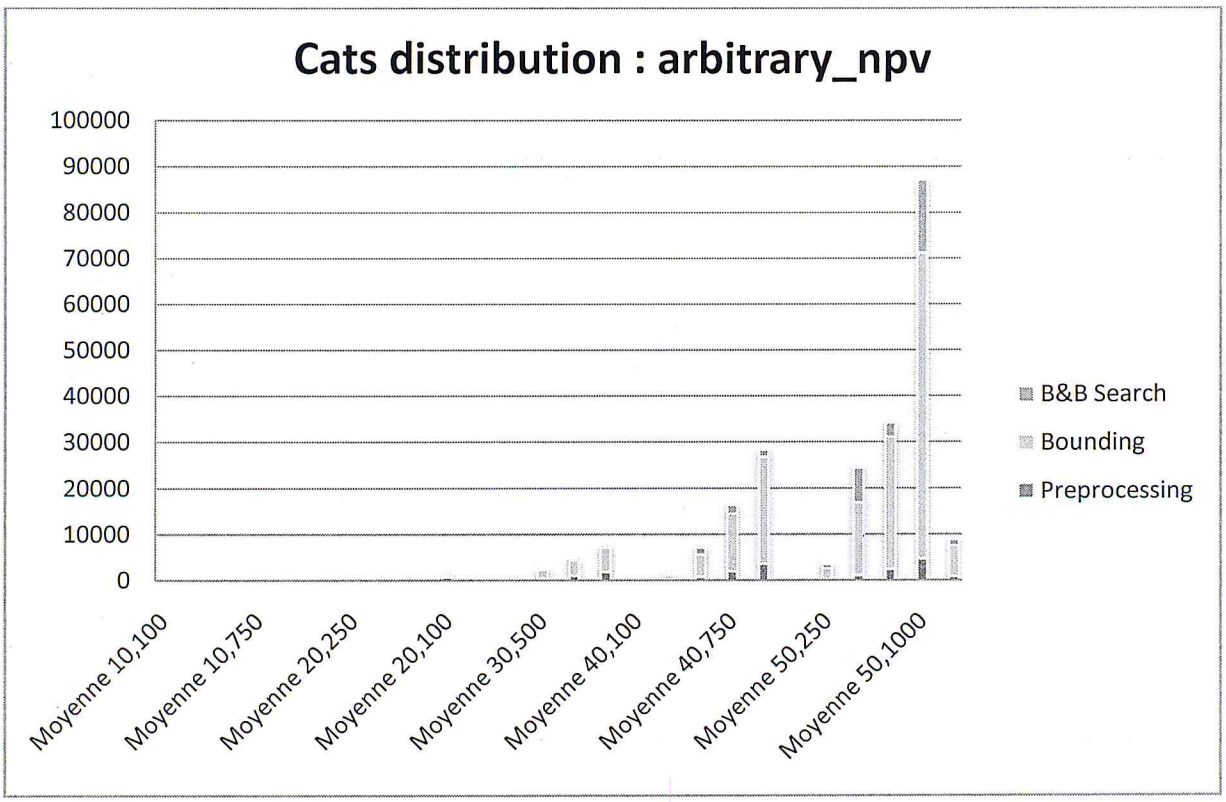

Figure 5.2.10

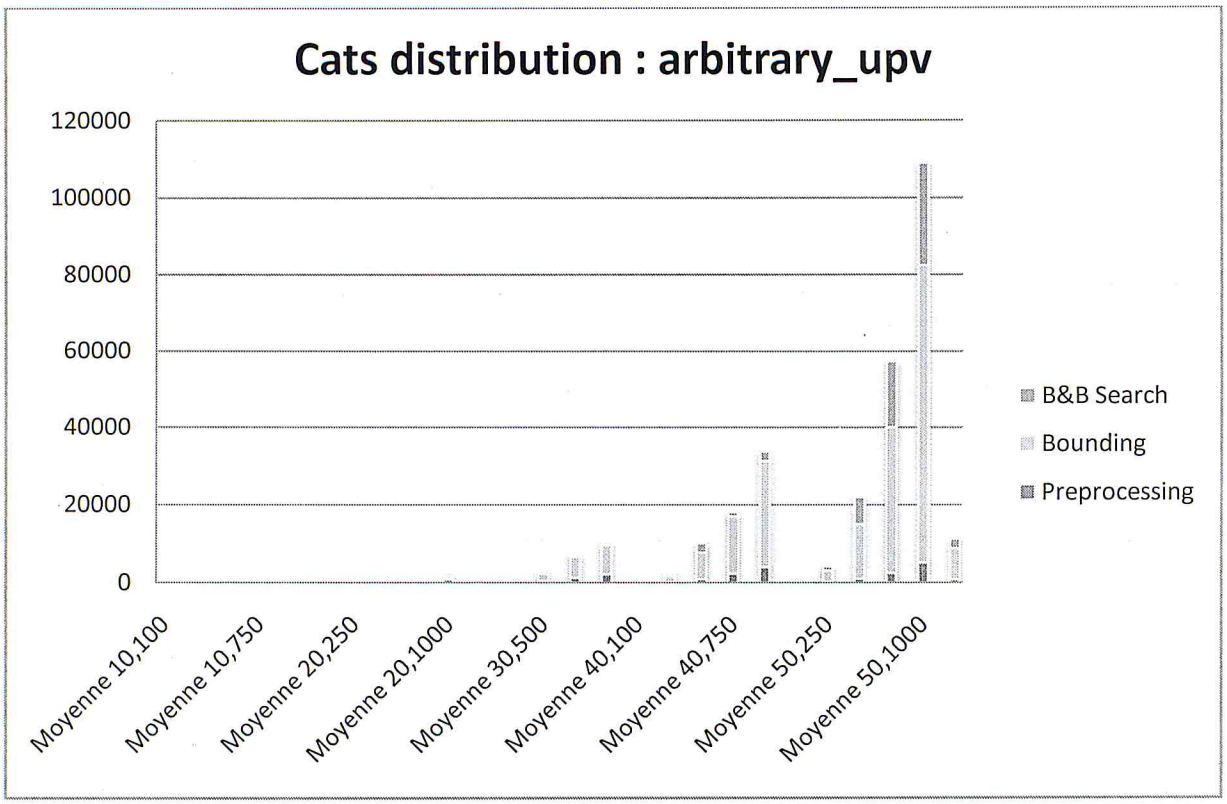

Figure 5.2.11

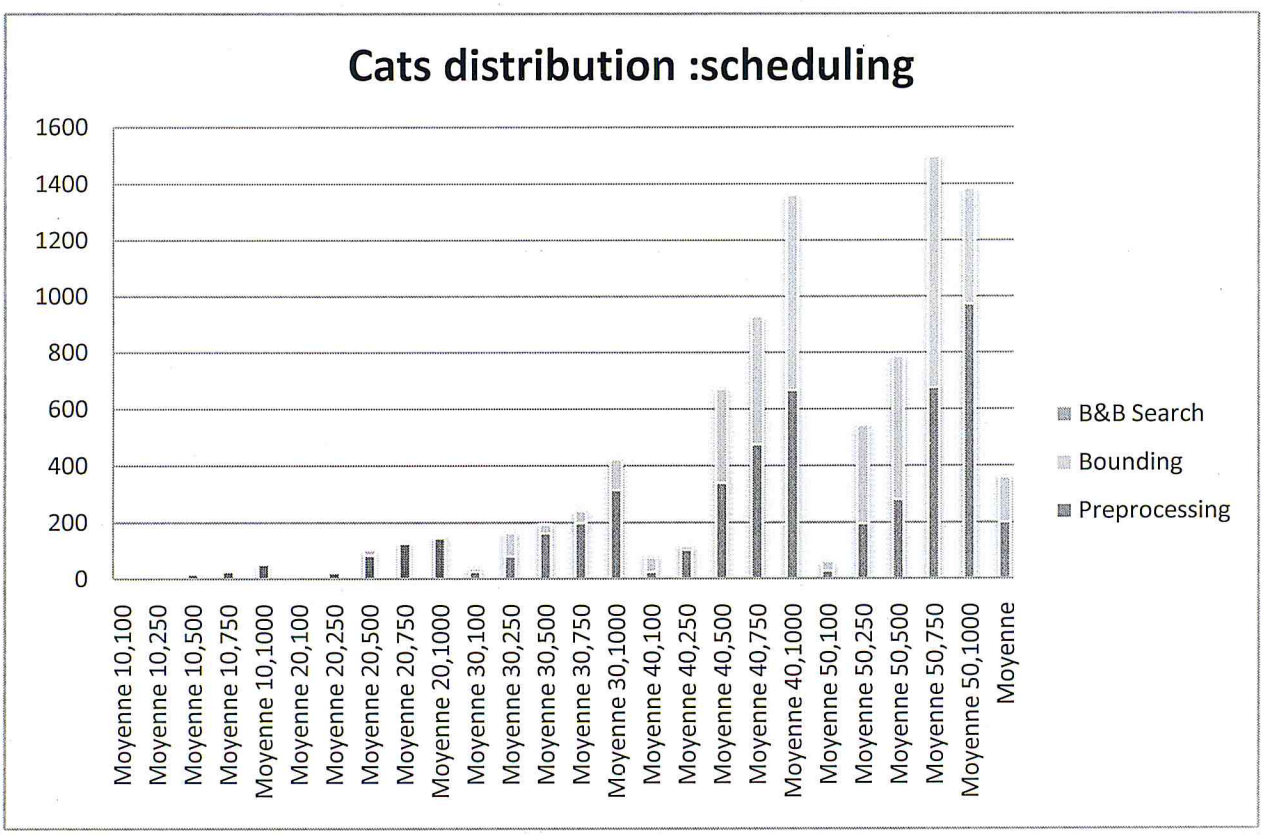

Figure 5.2.12

Ce qui ressort de chaque histogramme est une nette augmentation du temps d'exécution avec l'augmentation du nombre d'articles et/ou nombre d'offres. Cependant, l'influence du nombre d'articles reste prédominante, cela s'explique par le fait que plus le nombre d'articles augmente plus le nombre d'offres compatibles (pour un nombre d'article fixe) entre elles augmente. Aussi, on note que le plus grand du temps d'exécution est consommé dans la phase 2 « Bounding », cependant le temps que prendrait une exécution sans cette phase serait au mieux équivalent et pour bien des cas que nous avons testé, non relaté ici la performance est médiocre.

Pour toute les instances testé le temps d'exécution maximal ne dépasse pas les 2 minutes ce qui pour le contexte de notre application « site web dédié d'enchères » reste raisonnable.

Conclusion générale :

Conclusion générale :

Notre travail a porté sur deux flancs complémentaires pour le développement d'un site web dédié aux enchères combinatoires : le premier flanc s'est porté sur la conception et la réalisation du site web en tant que partie visible pour les utilisateurs (acheteurs et vendeurs), le second flanc ayant comme objectif de fournir une méthode de résolution pour le problème complexe des enchères combinatoires.

Nous entrevoyons quelques perspectives intéressantes surtout pour ce qui est de la méthode de résolution :

- Paralléliser l'implémentation pour un gain en temps de calculs.
- Utilisation du Revised simplex à la place de la méthode du simplexe standard utilisée dans notre implémentation.
- Combiner une méthode exacte pour la recherche et une méthode approximative pour le calcul des bornes.

Pour ce qui est de la partie web nous jugeons intéressant d'utiliser Ajax pouvant être implémenté en utilisant Google web toolkits pour une plus grande interactivité.

Références Bibliothécaires :

Références Bibliographiques :

[BNS09] KEVIN LEYTON-BROWN, EUGENE NUDELMAN and YOAV SHOHAM. Empirical Hardness Models: Methodology and a Case Study on Combinatorial Auctions. Journal of the ACM, 2009

[DAT08] Ekaterina Lebedeva. Hypertree Decompositions for Optimal Winner Determination in Combinatorial Auctions. Wien, 10. March 2008.

[DEV07] Isabelle Devarenne, Etude en recherche locale adaptative pour l'optimisation combinatoire, thèse doctorat, université de Franche-Comté, 2007.

[HAOxx] Optimisation parallèle & coopérative : Application au problème de recouvrement d'ensembles, thèse de magister, 18mars 2002, USTHB.

[HAD02] J. K. Hao. P. Galnier et M. Habib, Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes, Revue de l'intelligence artificielle, Vol : No.1999.

[Hol75] J. H. Holland. Adaptation In Natural And Artificial Systems, The University Of Michigan Press : Ann Arbor, Mi, 1975.

[LAM07] Vincent Lamareille, "La psychanalyse décrite en UML", 2001.

[LANxx] Etienne Langlet, Guide d'administration du serveur JavaEE sous windows et Linux, edition ENI.

[LEY03] Kevin Leyton-Brown. Resource Allocation in Competitive Multiagent Systems. PhD thesis, Stanford University, August 2003.

[MUxx] Victor MUNOZ and Javier MURILLO. CABRO: Winner Determination Algorithm for Single-unit Combinatorial Auctions. *University of Girona.*

[Pap82] C. Papadimitriou & K. Steiglitz. Combinatorial Optimization : Algorithms And Complexity Printice-Hall, 1982.

[RPH98] Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. Computationally manageable combinational auctions. Manage. Sci.,44(8):1131–1147, 1998.

[San02] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence, 135(1-2):1–54, 2002.

[San03] Tuomas Sandholm. Combinatorial Auctions, chapter Optimal Winner Determination Algorithms, pages 337 – 368. The MIT Press, 2006.

[San05] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: a fast optimal algorithm for winner determination in combinatorial auctions. Management Science, 51(3):374–391, 2005.

[San06] Tuomas Sandholm. Combinatorial Auctions, chapter Optimal Winner Determination Algorithms, pages 337 – 368. The MIT Press, 2006.

[TAL09] El-Ghazali TALBI, Metaheuristics From design to implementation, WILEY Editions, 2009.

[TAL95] E-G Talbi. Algorithmes Génétiques Parallèles : Techniques Et Application. Edition Hermes, 1995.

[Thi03] Cyril Thibaud. Installation, mise en oeuvre et programmation, 2003.

[Tho08] Theodoros Evdoridis "University of the Aegean, Greece, Security Threats in WebPowered Databases and Web Portals" (2008)

[Wiki11] Wikipedia the free Encyclopedia : Séparation et evaluation, disponible sur: http://fr.wikipedia.org/wiki/Séparation_et_évaluation