UNIVERSITE SIDI MOHAMED BEN ABDELLAH
FACULTE DES SCIENCES ET TECHNIQUES
Département des Mathématiques

Licence Sciences et Techniques (LST)

MATHEMATIQUES ET APPLICATIONS

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du Diplôme de Licence Sciences et Techniques

Titre

Problèmes d'Ordonnancement De Projet et Métaheuristiques

Présenté par :

BADR EL-AZRAK

Encadré par :

Pr. EL HILALI ALAOUI AHMED **(FSTF)**
Pr. *KADRI NASSER **(FSTF)***

Soutenu Le 09 Juin 2016 devant le jury composé de:

- **Pr.** *KADRI NASSER **(FSTF)***
- **Pr.** EL HILALI ALAOUI AHMED **(FSTF)**
- **Pr.** EL KHOUKHI FATIMA (**FLSH de Meknès**)
- **Pr.** BEN CHEIKH GHIZLANE (**FSJES de Meknès**)

Stage effectué à FSTF

Année Universitaire 2015 / 2016

FACULTE DES SCIENCES ET TECHNIQUES FES – SAISS
B.P. 2202 – Route d'Imouzzer – FES
212 (0)5 35 61 16 86 – Fax : 212 (0)5 35 60 82 14
Site web : http://www.fst-usmba.ac.ma

REMERCIEMENT

Je tiens tout d'abord à remercier Dieu le tout puissant, qui m'a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, je tiens à remercier mes encadrants Mr EL HILALI ALAOUI AHMED et Mr KADRI NASSER, pour leurs précieux conseils et leur aide durant toute la période du travail.

Mes vifs remerciements également aux membres du jury pour l'intérêt qu'ils ont porté à ce travail en acceptant de l'examiner et de l'enrichir par leurs remarques. Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

TABLE DE MATIERE :

VI. Conclusion

I- INTRODUCTION

Un problème d'ordonnancement consiste à organiser l'exécution d'un ensemble d'activités soumises à des contraintes de temps et de ressources. Dans ce type de problèmes, l'un des plus importants est le problème d'ordonnancement de projet à moyens limités ou RCPSP ("Resource-Constrained Project Scheduling Problem").

Le RCPSP est un problème NP-difficile au sens fort qui recouvre un grand nombre de problèmes théoriques d'ordonnancement comme les problèmes d'atelier (Job-Shop, Flow-Shop, Flow-Shop Hybride. . .), les problèmes à machines parallèles ou les problèmes d'ordonnancement cumulatifs (CuSP). Il se rencontre dans un grand nombre d'applications : industrielles (par exemple la découpe de tissu), informatiques (gestion des processus sur des machines parallèles ou multiprocesseurs). Il suscite donc un intérêt réel dans la communauté des chercheurs opérationnels, qui lui a consacré une littérature importante.

La résolution de ce problème d'optimisation combinatoire a déjà fait l'objet de nombreuses études. Les travaux visant à résoudre ce problème portent sur les méthodes de résolution exacte, le calcul de bornes inférieures et les méthodes approchées.

Ces travaux s'appuient sur des outils théoriques divers comme la programmation linéaire en nombres entiers (PLNE), la programmation par contraintes, les heuristiques et méta-heuristiques, et les méthodes exactes.

Ainsi, ce rapport sera structuré de la manière suivante :

Premièrement on va traiter les données d'un problème d'ordonnancement puis ses différentes variantes, ensuite on s'intéresse aux problèmes d'ordonnancement de projet à moyennes limitées, et après on recouvre les différentes méthodes de résolution exactes et approchées.

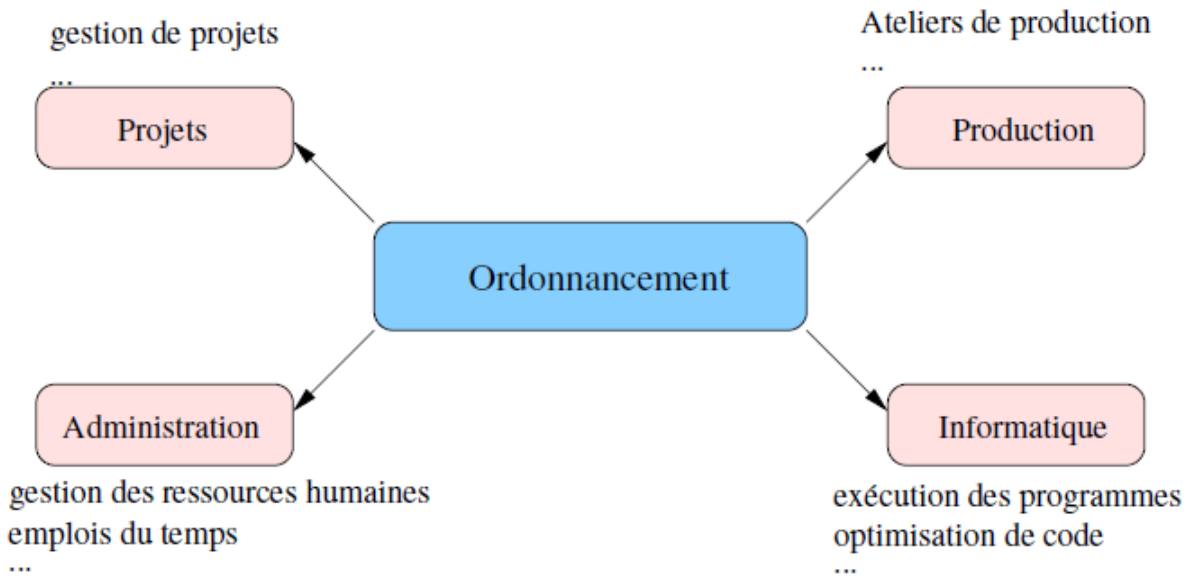
Et on termine ce manuscrit par une conclusion générale sur le sujet abordé et les avancées obtenues.

II. Les problèmes d'ordonnancement

II.1- Introduction

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de taches, compte tenu de contraintes temporelles (délais, contraintes d'enchainements…..) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises. [1]

Les problèmes d'ordonnancement touchent tous les domaines de l'économie : [03]



1. L'informatique (les tâches sont les programmes ; les ressources sont les processus, la mémoire).
2. La construction (suivi de projet).
3. L'industrie (activités des ateliers en gestion de production et problèmes de logistiques).
4. L'administration (emploi du temps).

Dans un problème d'ordonnancement interviennent deux notions fondamentales : les tâches et les ressources.

Une ressource est un moyen technique ou humain dont la disponibilité limitée ou non est connu à priori.

Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource.

II.2- Données d'un problème d'ordonnancement

1- Les taches et leurs caractéristiques.

2- Les contraintes potentielles.

3- Les ressources.

4- La fonction économique (fonction objectif).

II.2.1 –Tâches

On note en général **I**= {1,2, ……., n} l'ensemble des tâches et pi la durée de la tâche i si cette durée ne dépend pas des ressources qui lui sont allouées.

En plus de sa durée, une tâche a d'autres caractéristiques :

- ri : date de disponibilité, date avant laquelle la tâche i ne peut pas commencer.

- di : date échue, une tâche i doit être achevée avant sa date échue

- ti : date réelle de début de la tâche i, date qui sera déterminée uniquement pendant l'ordonnancement

- ci : date de fin réelle de la tâche i, date qui sera elle aussi calculée uniquement pendant l'ordonnancement.

II.2.1.1- Liens entre les tâches

Les tâches sont souvent liées entre elles par des relations d'antériorité. Si ce n'est pas le cas, on dit qu'elles sont indépendantes. La contrainte d'antériorité la plus générale entre deux tâches i et j, appelée contrainte potentielle, s'écrit sous la forme $t_j - t_i \geq a_{ij}$.

Quand la tâche j commence à la fin de la tâche i, dans ce cas, on dit qu'il y a succession simple.

$a_{ij}= p_i$ (p_i étant la durée de la tâche i)

$\forall$ I= {1,2, ……., n}, ri≤ ci ≤ di .

II.2.2- Ressources

On distingue deux types de ressources pouvant être requises pour les tâches : les ressources renouvelables et les ressources consommables. [01]

Une ressource est renouvelable si après avoir été allouée à une tâche, elle redevient disponible pour les autres (machines, processus, fichiers personnel).

Une ressource est consommable si après avoir été allouée à une tâche, elle n'est plus disponible pour les tâches restant à exécuter (argent, matières premières).

II.2.3-Contraintes

Les contraintes disjonctives et les contraintes cumulatives :

- Une contrainte disjonctive apparaîtra lorsque deux tâches, devant la même machine, ne pourront pas s'exécuter simultanément. [03]

- Une contrainte cumulative apparaîtra par exemple lorsque trois processeurs sont disponibles pour l'exécution de quatre tâches. On ne pourra exécuter plus de trois de ces tâches en même temps sans que l'on puisse savoir à l'avance laquelle sera retardée.

- Les contraintes de type potentiel : ce sont les contraintes de localisation temporelle (la tache i ne doit pas commencer avant telle date ou au contraire doit être achevée avant telle date).

- Les contraintes de successions ou positionnement entre les tâches ou d'antériorité (la tâche j ne peut pas commencer avant que la tâche i ne soit terminée).Ou simplement, parvenue à un certain degré d'achèvement.

- Les contraintes de type disjonctif : elles imposent la disjonction de deux intervalles de temps relatifs $]t_i,\ t_i + p_i[\cap]t_j,\ t_j + p_j[= \varnothing$ par exemple à l'exécution de deux tâches i et j qui ne peuvent être réalisées simultanément.

Disjonction d'inégalités de potentiels : $t_j - t_i \geq p_i ou\ t_i - t_j \geq p_j$

-Les contraintes de ressources : Elles sont liées à l'utilisation et la disponibilité des ressources requises par les taches.

On note : $A_t = \{i \in A | S_i \leq t \leq S_i + p_i\}$

L'ensemble des taches qui consomme la ressource k à l'instant t.

On traduit la limitation de la capacité de k par : ∀ t ; $\sum_{j \in A_t} r_{jk} \leq R_k$

II.2.4- Fonction objectif (fonction économique)

Il faut programmer les tâches de façon à optimiser un certain objectif qui sera, suivant les cas : [01]

- La minimisation de la durée totale (critère le plus fréquemment utilisé)
- Le respect des tâches de commandes (dates échues)

- La minimisation d'un coût (coût de production)

-La minimisation du nombre de ressources utilisées

-La minimisation des sommes des retards

-Maximiser le critère appelé NPV (Net Present Value) qui correspond à maximiser la valeur à tout instant du projet .Ce critère s'appuie sur la notion de couts et de gains associés à chaque tache du projet.

Etc ….

D'une manière générale, trois types d'objectifs sont essentiels dans la résolution des problèmes d'ordonnancement :

- L'utilisation efficace des ressources

- Un délai d'exécution des tâches aussi faible que possible

- Le respect des dates d'achèvement prescrites à l'avance

Les variables intervenant le plus souvent dans l'expression de la fonction économique (fonction-objectif) sont :

- La date c_i de fin d'exécution de la tâche i.
- Le retard $Ti = max\ (0,\ ci - di)$ de la tâche i,
- L'indicateur de retard Ui $(Ui = 0\ si\ ci \le di,\ Ui = 1\ sinon)$

Les critères usuels sont :

- La durée totale $C_{max} = max\ (ci)$
- Le plus grand retard $T_{max} = max\ T_i$,
- Le retard moyen pondéré $\sum w_i T_i$
- Ou les stocks d'encours $\sum w_i T_i$

III- Variante des problèmes d'ordonnancement

Selon la nature des variables et des contraintes mises en jeu, plusieurs variantes des problèmes d'ordonnancement sont proposées dans la littérature. On distingue deux grandes familles de problèmes d'ordonnancement, les problèmes d'ordonnancement de projet et les problèmes d'ordonnancement d'ateliers.

III.1- Ordonnancement de projet

Il s'agit d'un problème très général qui recouvre un grand nombre de situation d'ordonnancement et constitue une très vaste littérature [03]. Les questions les plus fréquentes dans un problème d'ordonnancement de projet, concernent la détermination de la durée totale du projet et la mise en évidence des marges temporelles exploitables pour minimiser le coût d'utilisation des ressources. L'ordonnancement de projet demeure central en ordonnancement car il permet de présenter des concepts fondamentaux et des algorithmes de base pour le traitement des contraintes temporelles.

III.2-Ordonnancement d'atelier

Une deuxième famille essentielle des problèmes d'ordonnancement, [03] ce sont les problèmes d'ateliers où on doit exploiter au mieux des moyens limités (des machines) pour réaliser un ensemble très varié de produits. La complexité réside alors non pas dans le processus de fabrication, qui est prédéterminé (par exemple sous forme de gammes opératoires), mais plutôt dans la combinatoire qui naît de la prise en compte de la limitation des ressources existences. Indifféremment des problèmes d'ordonnancement prévisionnels, qui prévoient des modalités d'exécution d'un ensemble de tâches par un ensemble de ressources sur un horizon fini ou infini (après avoir effectué des hypothèses préalables sur l'évaluation de la situation du système au cours du temps). On parle des problèmes d'ordonnancement d'ateliers à temps réel, quand il s'agit, d'adapter en permanence les modalités d'exécution d'un ensemble de taches par un ensemble de ressources à la situation réelle du système considéré. En plus des problèmes d'ordonnancement de projet et d'ateliers, on cite une famille de problèmes d'ordonnancements issus du monde pratique, à savoir les problèmes d'ordonnancement cycliques. Il s'agit d'organiser les activités de production en cherchant à répéter un cycle de base relativement bien optimisé, illustrant ainsi la notion même d'ordonnancement cyclique. Un ordonnancement cyclique est ensuite

construit par calcul des dates sur la période de référence et répétition cyclique de l'ordonnancement ainsi obtenu. Ces problèmes sont très présents dans le milieu industriel.

III.3- Gestion du personnelle

Les ressources humaines sont parmi les ressources les plus cruciales et les plus coûteuses pour la majorité des organisations [03]. Par conséquent, il est capital que chaque entreprise élabore une stratégie efficace de planification de ces ressources. Dans cette perspective, un ordonnancement efficace du personnel permet de générer des réductions des coûts, d'améliorer la qualité des services ou des produits offerts et de maximiser la satisfaction des employées.

Les problèmes d'ordonnancement du personnel sont des cas particuliers du problème d'allocation des ressources. Ils peuvent se présenter selon plusieurs configurations ou modèles en fonction des particularités du milieu organisationnel et de la durée de la période de planification. Généralement, ces problèmes portent sur la détermination du nombre d'employés requis pour répondre à une demande en produits ou en services, sur leurs affectations à des tâches précises le long d'intervalles de temps de durée variable, ou encore sur la détermination du lieu de travail pour chaque employé.

Ainsi définis, les problèmes d'ordonnancement du personnel peuvent être observés dans plusieurs types d'entreprises manufacturières ou de services. Ce sont des problèmes récurrents dans des domaines tels que : le transport; la santé ; l'enseignement; la production manufacturière; les centres d'appel; la restauration ou les services de protection et d'urgence. Dans les travaux recensés, on rencontre trois catégories de problèmes d'ordonnancement du personnel : le problème de planification des jours de repos, le problème d'élaboration des quarts de travail et le problème d'élaboration des patrons de travail.

IV -Les problèmes d'ordonnancement de projet

IV. 1 - Définition d'un projet

On définit un *projet* par un ensemble d'activités (ou tâches) ayant des caractéristiques propres, qui sont exécutées grâce à un ensemble de ressources [01]

De manière générale, une *activité* est caractérisée par sa *durée*, sa *consommation* en ressources, son *coût* et sa *priorité*.

IV.2- Avec contraintes de ressources

1 –Description du RCPSP

La réalisation d'un projet consiste en l'exécution, par des ressources, d'un ensemble d'activités (ou taches) de durées données et liées entre elles par des contraintes de précédence [04]. Dans sa forme classique, une instance du RCPSP est la donnée d'un ensemble A de n activités d'un projet et d'un ensemble R de m ressources. Les activités du projet sont non-interruptibles (ou non-préemptives).

Ainsi, une activité i termine son exécution à l'instant S_i+p_i, où Si est la date de début d'exécution de i et pi, sa durée. Les activités sont liées par des contraintes de précédence simples, de la forme i → j, interdisant de débuter l'exécution de la seconde activité j avant la fin de la première i. On modélise couramment le projet par un graphe values, orienté et sans circuit $G = (V;E; p)$, le graphe potentiel-taches, formé du graphe associé à la relation formée par les contraintes de précédence, auquel sont ajoutées deux activités fictives de durées nulles, représentant respectivement, le début et la fin du projet : 0 pour le début du projet dont on commence l'exécution à l'instant de référence

$S_0 = 0$, et S_{n+1} succédant à toutes les activités du projet.

L'objectif du RCPSP est de réaliser l'ensemble du projet en un temps minimale, autrement dit de déterminer un ordonnancement $S = (S_0; S_1;\ldots\ldots\ldots; S_{n+1})$ de durée S_{n+1} minimale , à la fois en respectant les contraintes de précédences , c'est-à-dire les inégalités de potentiels $S_j \geq S_i + p_i$,pour tout couple d'activité (i, j) ∈E, et en résolvant à tout instant t les conflits dans l'utilisation de chaque ressource k ; $\sum_{j \in A_t} r_{jk} \leq R_k$

Un tel ordonnancement est dit optimale, tandis qu'un vecteur S vérifiant les contraintes de précédences et les contraintes de ressources sans nécessairement minimiser Sn+1, est appelé ordonnancement réalisable.

A partir des données du problème, d'autres valeurs, qui seront utilisées tout au long de ce document, peuvent être initialisées à la toute première étape du processus d'ordonnancement :

-T ∈ N : l'horizon est une évaluation par excès de la durée minimale d'un ordonnancement : $S_{n+1} \leq T$. Elle peut être initialisée par exemple avec la somme des durées des activités.

- $ES_i \in N$ (resp. $EF_i \in N$) La date de début (resp. de fin) au plus tôt de l'activité i pour « earliest starting time » (resp. « earliest finish time. ») Avant laquelle i ne peut commencer (resp. finir) $ES_i \leq S_i$ et $EF_i \leq S_i + p_i$. Elle est initialisée à 0 (resp. p_i) pour toute activité i.

- $LS_i \in N$ (resp. $EF_i \in N$) La date de début (resp. de fin) au plus tard de l'activité i pour « latest starting time » (resp. « latest finish time ») après laquelle i ne peut commencer (resp. finir) $S_i \leq LS_i$ et $LF_i \geq S_i + pi$. Elle est initialisée à T-pi (resp. T) pour toute activité i.

2- Formulation linéaire

On distinguera donc les modèles en temps continu, où les dates de début des activités sont modélisées par des variables réelles (et la séquencement entre les activités par des variables binaires), des modèles en temps discrétisé, où les variables binaires liées aux activités sont aussi indicés par les instants.

- ### Temps continue

La formulation conceptuelle [04].

$$(P_1)\begin{cases} MIN\ S_{n+1} & (1) \\ sujet\ à & \\ S_j - S_i \geq p_i\ , \quad \forall\ (i,j) \in E & (2) \\ \sum_{j \in A_t} r_{jk} \leq R_k, \forall k \in IR\ , \forall t \in [0,T] & (3) \\ S_i \geq 0\ , \forall\ i \in V & \end{cases}$$

$$Avec\ A_t = \{i \in A \mid S_i \leq t \leq S_i + p_i\}$$

Les variables S_i représentent les dates de début des activités. L'objectif (1) est la minimisation de la date de début de la dernière activité fictive. (2) sont les contraintes de précédence. Les contraintes de ressources (3) signifient qu'à tout instant t et pour toute ressource k, la somme des consommations de k sur l'ensemble A_t des activités en cours à l'instant t est inférieure à la capacité de k.

- Ensemble critique minimaux

Pour le RCPSP en modélisant les contraintes de ressources (3) au moyen des ensembles critiques d'activités minimaux C ∈ C_m définis formellement par

(1) C×C∩E=∅ , (2) $\exists k \in$, $\sum_{j \in C} r_{jk} > R_k$ et (3) $\forall C' \subsetneq C$, ∀k ∈IR

$$\sum_{j \in C'} r_{jk} \leq R_k \,. \quad [04]$$

$$(P_2) \begin{cases} \min S_{n+1} \\ \text{sujet à} \\ X_{ij} = 1 & (4) \\ X_{ij} + X_{ji} \leq 1 & (5) \\ S_j - S_i \geq -M + (p_i + M)X_{ij} & (6) \\ \sum_{(i,j) \in C \times C} X_{ij} \geq 1 & (7) \\ X_{ij} \in \{0,1\};\ X_{ii} = 0 \\ S_i \geq 0 \end{cases}$$

X_{ij} est égal à 1 si l'activité i précède l'activité j, 0 sinon. (4) modélisent les contraintes de précédence initiales du problème, et les contraintes (7) imposent qu'au moins deux activités d'un ensemble critique minimal soient exécutées l'une après l'autre. Les contraintes (5)

garantissent l'absence de cycles dans le graphe de précédence. En fin, les contraintes (6) lient les deux types de variables du modèle : pour une valeur de M suffisamment large (par exemple M = T), les contraintes imposent que, pour toute précédence x_{ij} = 1, l'activité j doit débuter son exécution après la complétion de i. Autrement, si x_{ij} = 0, la distance $S_j - S_i$ n'est pas contrainte ($S_j - S_i \geq -M$).

- ## Temps discrétisé

Les formulations en temps discrétisé contiennent un nombre de variables dépendant de l'horizon T qu'on suppose donc connu. On pose $H = \{0,,,,, T\}$ l'ensemble des instants possibles de début des activités [04].

La première modélisation du RCPSP en un programme linéaire a été donnée par Pritsker et al. Et ne contient qu'un seul type de variables, binaires pour figurer la date de fin des activités. Nous reportons ici le modèle équivalent avec les dates de début : y_{it} = 1 si i débute à l'instant t et 0 si non. Au moyen de ces variables, on peut désormais caractériser $A_t = \{i \in A / \exists \tau \in \{t - p_i + 1, ..., t\}, X_{i\tau} = 1\}$ à tout instant t et donc modéliser la contrainte de ressources (3) par (11). Les contraintes de précédence (10) et de non-préemption (9) et l'objectif (8) se traduisent facilement avec la correspondance :

$$S_i = \sum_{t \in H} t.y_{it}$$

$$(P_3)\begin{cases} \min \sum_{t\epsilon H} t y_{(n+1),t} & (8) \\ sujet\ à & \\ \sum_{t=0}^{T} y_{it} = 1 \quad \forall\, i \,\epsilon\, V & (9) \\ \sum_{t=0}^{T} t(y_{jt} - y_{it}) \geq p_i \quad \forall (i,j)\epsilon E & (10) \\ \sum_{i\epsilon V} r_{ik} \sum_{h=t-pi+1}^{t} y_{ih} \leq R_k \ \forall k \in \mathrm{IR}, \forall t \in H & (11) \\ y_{it} \in \{0,1\} \qquad \forall i \in V, \forall t \in H & (12) \end{cases}$$

Description des contraintes

La fonction-objectif est donnée par l'équation (8). Elle consiste à minimiser la date de début de l'activité fictive de fin de projet *n*+1, sachant que celle-ci est l'activité jalon de durée opératoire nulle (p_{n+1} = 0), marquant la fin du projet. Par conséquent, chercher à minimiser cette date de début, revient donc à minimiser la date de fin du projet (*makespan*).La minimisation de cette fonction objectif est soumise aux contraintes suivantes :

– la contrainte (10) représente la contrainte de précédence ; elle modélise les relations de précédences entre les différentes activités, en stipulant que si la paire d'activités (*i*, *j*) appartient à l'ensemble des précédences *E*, alors la date de début de l'activité *j* sera ultérieure ou égale à la date de début de l'activité *i*, augmentée de la durée opératoire de l'activité *i*.

– la contrainte (11) représente la contrainte de ressource ; elle est aussi appelée *contrainte cumulative*, elle s'assure, pour chaque ressource que la somme des consommations des activités en cours d'exécution ne dépasse pas la capacité de la ressource.

– La *contrainte de non-préemption* des activités, modélisée par l'équation (9), impose à chaque activité de ne débuter qu'une et une seule fois, sur toute la durée du projet.

Proposition 0

La formulation conceptuelle est équivalente à la formulation a temps discrétisée (DT). [03].

Preuve

Si (i, j) ∈ E, l'activité i démarre au temps t, si et seulement si y_{it}=1.Ainsi, $\sum_{t\grave{o}H} t.y_{it}$ =t en vertu des contraintes (9) et (12). On a alors

$\sum_{t\grave{o}H} t.y_{jt} \geq \sum_{t\grave{o}H} t.y_{it} + p_i \Longleftrightarrow \sum_{t\grave{o}H} t.y_{jt} \geq t+p_i$. Ce qui revient à dire que pour tout t'∈ H,

$y_{it'}$ =0 si t'≤ t+ p_i. En d'autres termes, l'activité j ne peut débuter qu'après l'achèvement de l'activité i. Les contraintes de ressource de cette formulation sont aussi équivalentes à celles de la formulation conceptuelle.

En retrouve aisément la formulation conceptuelle du RCPSP à travers ce modèle, en traduisant les contraintes de précédence (10), de non-préemption (9) et la définition de l'objectif grâce à la correspondance $S_i = \sum_{t\grave{o}H} t.y_{it}$

- **Représentation d'une solution à l'aide d'un graphe**

Exemple 1

Considérons un problème à 6 activités et 2 ressources dont l'instance est donnée, tableau 1. Une solution de ce problème est proposée dans le tableau 2. Et le graphe associé est donnée, figure 3.[04]

Activités	Durée	Consommation de ressource 1	Consommation de ressource 2	Prédécesseurs
1	2	5	0	----------
2	3	3	1	----------
3	2	1	1	----------
4	2	4	1	2, 3
5	1	0	3	2
6	3	3	2	4

TAB.1 – exemple d'instance pour un RCPSP à 6 activités et 2 ressources

Activités	Début au plus tôt	Début au plus tard
1	0	0
2	2	2
3	2	3
4	5	5
5	5	6
6	7	7

TAB. 2- dates au plus tôt et au plus tard

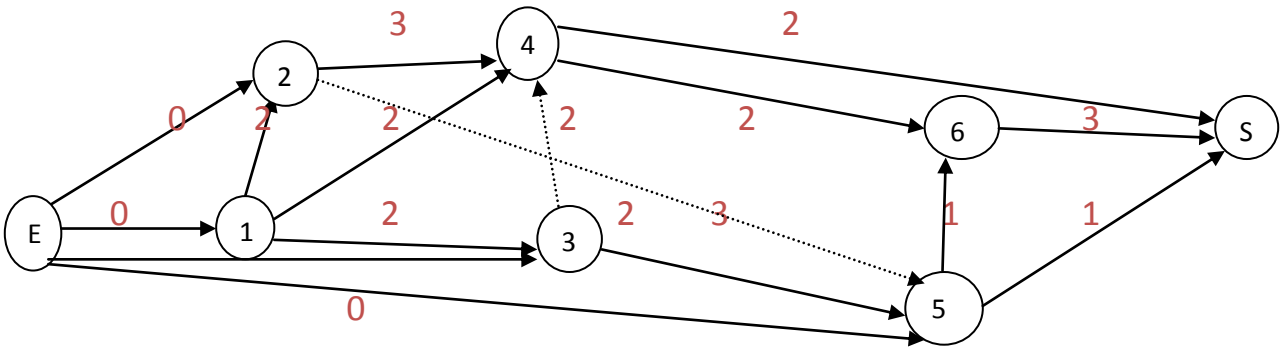


Fig.3- Représentation par un graphe. Les flèches en pointillés représentent une contrainte de précédence

IV.3- Sans contraintes de ressources

Lorsque les contraintes représentant les ressources ne sont pas prises en compte, les seules contraintes dans le cadre de la minimisation de la durée totale du projet sont les contraintes temporelles [04]. Il s'agit donc d'effectuer un ordonnancement à partir des contraintes de localisation temporelle et de précédence. Ce problème est celui qu'on obtient lorsqu'on construit un ordonnancement à capacité de ressource infinie.

1-Objectif :

Calculer la durée minimale du projet, les ressources étant supposées illimitées

=) Minimiser ($S_{n+1} - S_0$) sous les contraintes de potentiels.

=) Déterminer les dates de début au plus tôt et au plus tard des taches.

=) Déterminer les taches critiques.

2-Formulation mathématique

Déterminer (S_0, S_1,……, S_n, S_{n+1}) de façons à minimiser (S_{n+1}- S_0)

Sous les contraintes :

Contraintes de potentiels : $S_j - S_i \geq a_{ij}$

Contraintes de non-négativité S_0, S_1,……, S_n, $S_{n+1} \geq 0$

C'est un programme linéaire.

3-Exemple de programmation mathématique

Soit un ensemble de 7 tâches, figure 1, il est question de calculer t_i pour i=1,..., 7, la date de début d'exécution de la tâche i, en minimisant le C_{max} et en respectant les contraintes, d_i étant la durée opératoire de la tâche i [04].

Contraintes des données	Contraintes de précédence	Disjonction d'inégalité de potentiel
$d_1=3$	$t_1+d_1 \leq t_3$	$(t_1+d_1 \leq t_2)$ ou $(t_2+d_2 \leq t_1)$
$d_2=2$	$t_1+d_1 \leq t_4$	$(t_3+d_3 \leq t_5)$ ou $(t_5+d_5 \leq t_3)$
$d_3=4$	----------	----------
$d_4=1$	$t_3+d_3 \leq t_6$	$(t_3+d_3 \leq t_7)$ ou $(t_7+d_7 \leq t_3)$
$d_5=8$	$t_4+d_4 \leq t_7$	$(t_7+d_7 \leq t_5)$ ou $(t_5+d_5 \leq t_7)$
$d_6=3$	$t_2+d_2 \leq t_5$	----------
$d_7=7$	----------	$(t_4+d_4 \leq t_6)$ ou $(t_6+d_6 \leq t_4)$

Figure 1 -**Exemple de programmation mathématique**

V -Méthode de résolution des problèmes d'ordonnancement

1- introduction

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, procédure par séparation et évaluation, théorie des graphes) [11].

Ces méthodes garantissent en général l'optimisation de la solution fournie. Mais les algorithmes dont la complexité n'est pas polynomiale ne pouvant pas être utilisés pour des problèmes de grande taille, d'où la nécessité de construire des méthodes de résolution approchée, efficaces pour ces problèmes souvent NP-difficiles.

L'analyse d'un problème particulier permet d'obtenir des propriétés sur la structure des solutions optimales, propriétés à partir desquelles on peut ou bien caractériser les solutions optimales ou bien réduire l'espace des solution à l'explorer (notion de sous-ensemble dominant qui contient au moins une solution optimale).

2- méthode de résolution exacte

2.1- Diagramme de Gantt

PRINCIPE :

Ce type de diagramme a été mis au point par un américain Henry Gantt [06]

On représente au sein d'un tableau en ligne les différentes taches et en colonne les unités de temps (exprimés en mois, semaines, jours….)

La durée d'exécution d'une tache est matérialisée par un trait au sien du diagramme.

REALISATION :

Les différentes étapes de réalisation d'un diagramme de Gantt sont les suivantes :

Première étape : On détermine les différentes tâches (ou opérations) à réaliser et leur durée.

Deuxième étape : on définit les relations d'antériorité entre tâches.

Troisième étape : on représente d'abord les tâches n'ayant aucune antériorité, puis les tâches dont les tâches antérieures ont déjà été représentées, et ainsi de suite…

Quatrième étape : on représente par un trait parallèle en pointillé à la tâche planifiée la progression réelle du travail.

Exemple 2 :

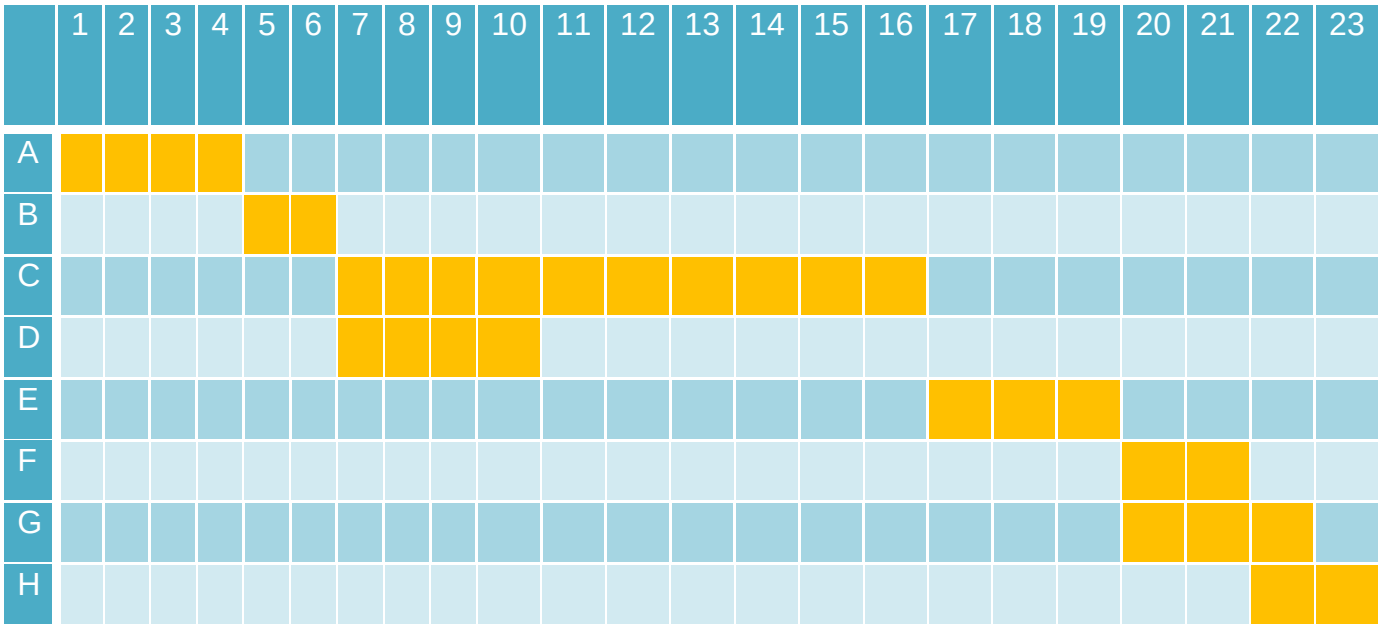


Tableau n°2 : Le diagramme de Gantt [14]

Remarque :

- Chaque colonne représente une unité de temps.
- Les durées d'exécution prévues des taches sont représentées par trait épais (10 unités de temps pour c).
- Les contraintes de succession se lisent immédiatement :
  - ✓ La taches B et C succèdent à la tache A.
  - ✓ D succèdent à B.

Exemple 3 :

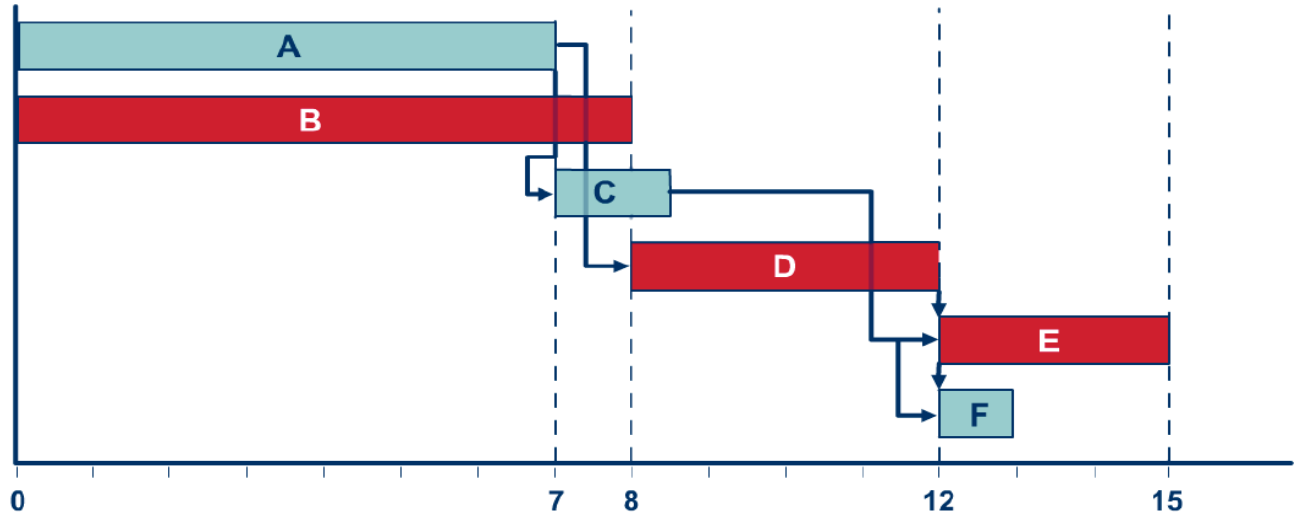


Tableau 3 : ***Exemple de diagramme de Gantt. [14]***

2.2- Méthode PERT

C'est une modélisation du problème central de l'ordonnancement par un graphe, elle permet d'évaluer la durée de réalisation d'un projet complexe et de détecter les parties de ce projet ne supportant aucun retard. Ce graphe porte le nom de graphe PERT (**P**rogram **E**valuation and **R**eview **T**echnique) ou graphe potentiel-étape. Nous donnons ici quelques éléments sur cette modélisation.

Dans cette représentation, les arcs sont associés aux tâches; ils sont valus par la durée des tâches, et les sommets représentent certains événements qui regroupent en général la fin de certaines tâches et le début d'autres.

2.2.1- modélisation en graphe

Le graphe orienté et value G = (X, U) (un graphe où chaque arc de U est associée une valeur réelle de la durée d'une tâche) défini par : [14]

-A chaque tache x on associe un sommet i ∈ X de départ et un sommet

j ∈ X de fin tel que i<j.

-On définira un arc (i, j) de longueur d_{ij} pour chaque tache x avec d_{ij} la durée d'exécution de la tache.

Le graphe reflète les précédences requises dans l'exécution des différentes tâches du projet. Ce graphe est sans circuit du fait que l'existence d'un circuit impliquerait une contradiction dans les précédences; une tâche devant en même temps précéder et succéder à une autre. Il

est moins facile à représenter, il faut définir les événements correspondant aux sommets. Certaines contraintes de succession nécessitent l'introduction de tâches fictives. Enfin, la prise en compte de contraintes qui ne sont pas des contraintes de succession peut être plus délicate.

Supposons par exemple que l'on ait les tâches suivantes A, B, C, D avec :

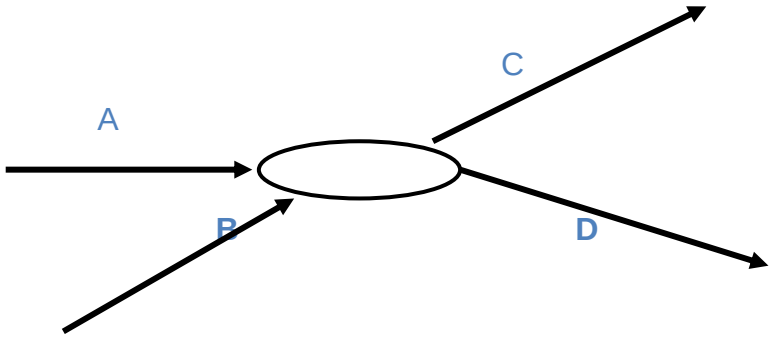
A précède C et D

B précède D

A ces 4 tâches sont associés 4 arcs : A, B, C, D.

A précède C et D se traduit par : l'extrémité de l'arc correspondant à la tâche A coïncide avec l'origine de l'arc correspondant à la tâche C et avec l'origine de l'arc correspondant à la tâche D.

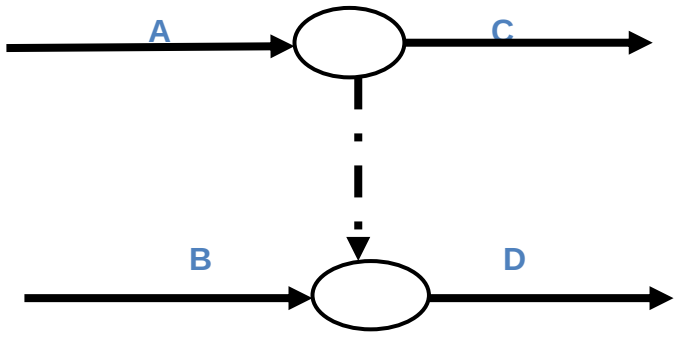
B précède D est traduit par : l'extrémité de l'arc correspondant à la tâche B coïncide avec l'origine de l'arc correspondant à la tâche D. Cela peut conduire à la représentation suivante



Dans cette représentation, C et D ont même origine ce qui impose la contrainte :

B précède également C qui n'était pas dans les données du problème.

Il faut alors introduire une **tâche fictive** de longueur 0.



2.2.2- Ordonnancement au plus tôt

On appelle **date de début au plus tôt** d'une tâche la plus petite date à laquelle elle peut être lancée.

Le calendrier de l'ensemble des tâches est appelé **"ordonnancement au plus tôt".**

Proposition 1

La date de début au plus tôt d'une tâche est égale à la longueur du plus long chemin entre le sommet "début" et le sommet représentant cette tâche dans le graphe. [05]

Preuve

A chaque **contrainte de précédence** on associe un arc (i, j) : la contrainte

$t_j \geq t_i + d_{ij}$ est associée à l'arc (i, j) de **longueur** d_{ij}, où ti représente la date de début de la tâche i.

Si on additionne ces inégalités pour tous les arcs du chemin, on arrive à $t_i \geq t_{débuts}$ + somme des longueurs des arcs de n'importe quel chemin reliant le sommet début au sommet i.

Comme $t_{débuts} = 0$, on en déduit que la date de début de la tâche i est au moins égale à la longueur du plus long chemin.

Ce résultat est vrai pour tous les chemins reliant le sommet "début" au sommet i, la date de début au plus tôt de la tâche i est égale à la longueur du plus long chemin du sommet "début" au sommet i.

Calcul des dates au plus tôt

D'après la proposition précédente, le calcul des dates au plus tôt revient à calculer celui de la longueur d'un plus long chemin. On peut donc utiliser les algorithmes adaptés à la détermination de plus longs chemins. [05]

2.2.3- Ordonnancement au plus tard

Il s'agit en l'occurrence de déterminer la date à laquelle chacune des tâches doit impérativement avoir commencé si on veut que la durée totale des travaux soit respectée.

Définition :

On appelle **date de début au plus tard** d'une tâche la date à laquelle elle doit impérativement avoir commencé afin que la date de fin de travaux soit respectée.

Le calendrier correspondant est l'**ordonnancement au plus tard.**

Proposition 2 :

La date de début au plus tard d'une tâche est égale à la différence entre la date de fin des travaux et la longueur du plus cour chemin du sommet représentant cette tâche dans le graphe au sommet "fin".

Calcul des dates au plus tard :

Pareillement au calcul de date au plus tôt, le calcul des dates au plus tard revient à un calcul de plus long chemin dans un graphe valu. Plus précisément, on détermine suivant un ordre de sommet décroissant, le plus long chemin entre chaque sommet i et le sommet "fin" à l'aide d'un algorithme du plus long chemin. On en déduit ensuite la date au plus tard par une simple soustraction (d'après la proposition 2).

2.2.4-Marges libres et Marges totales

Définition :

La **marge libre** d'une tâche représentera concrètement le retard maximal qu'on pourra prendre dans la réalisation d'une tâche sans retarder le début des tâches suivantes, on la notera **ML**. [13]

La **marge totale** d'une tâche représentera concrètement le retard maximal qu'on pourra prendre dans la réalisation d'une tâche sans retarder l'ensemble du projet, on la notera **MT**.

Calcul des marges :

Soit ij la tâche allant du sommet i au sommet j, donc la marge libre et la marge totale d'une tâche ij sont calculé respectivement :

$$\mathbf{ML_{ij} = t_j - t_i - d_{ij}}$$

$$\mathbf{MT_{ij} = T_j - t_i - d_{ij}}$$

2.2.5- Tâches critiques et chemins critiques

On qualifiera de **critique**, [14] une tâche dont la marge totale est nulle, c'est en quelque sorte une tâche "urgente", une tâche sur laquelle il ne faut pas prendre de retard si l'on ne veut pas augmenter la durée totale du projet.

Le **chemin critique** est le plus long chemin du sommet "début" au sommet "fin". Il est constitué des arêtes associées à des tâches critiques. Il n'est pas nécessairement unique.

2.2.6- Principe de la méthode PERT

Pour établir les données de l'ordonnancement des tâches, la méthode PERT procèdent en trois phases : [12]

La première phase consiste à effectuer un parcours (*propagation avant* aussi appelée *forward recursion*) du réseau représentant le projet dans le sens des contraintes de précédence pour déterminer les dates au plus tôt des tâches. Il s'agit en fait de déterminer la première tâche du graphe (celle qui n'a pas de prédécesseur) à laquelle on affecte la date au plus tôt à laquelle elle peut commencer, puis d'appliquer un algorithme de calcul de plus long chemin entre cette tâche et toutes les autres tâches du problème. Dès lors les dates de début au plus tôt de chacune des tâches sont établies. Ce sont en effet les dates qui correspondent à la date au plus tôt de la première tâche plus la longueur du plus long chemin entre la première tâche et la tâche en question. Dès cette phase terminée, on connaît la durée minimale du projet dans sa globalité puisque cette durée est la différence entre la date de fin de la dernière des tâches et la date de début de la première.

La seconde phase (propagation arrière appelée backward recursion) est similaire à la première. Cependant comme ce sont les dates de début ou de fin au plus tard qui désirées, on commence par la dernière tâche du projet. Étant donné que l'objectif est de minimiser la durée du projet, on ne veut pas changer la date de fin du projet, donc on pose l'égalité entre la date de fin au plus tard de la dernière des tâches et sa date de fin au plus tôt. Il suffit ensuite de rechercher les chemins les plus longs entre la dernière tâche et toutes les autres pour obtenir pour chaque tâche sa date de fin au plus tard : elle est égale à la date de fin au plus tard de la dernière tâche moins la longueur du plus long chemin de la dernière tâche à la tâche en question.

Finalement, la troisième phase consiste à déterminer les marges des tâches et les chemins critiques. La marge d'une tâche est égale à la différence entre sa date de début au plus tard et sa date de début au plus tôt. Les chemins critiques d'un projet sont les chemins du début du projet à sa fin qui ne sont constitués que de tâches critiques, c'est-à-dire de tâches dont la marge est nulle. Ces tâches critiques sont en fait les tâches qu'on ne peut retarder sans augmenter la durée du projet.

2.3-Méthode des Potentiels Métra (MPM)

Définition :

La Méthode des Potentiels et antécédents Métra (MPM) [08] est une méthode d'ordonnancement basée sur la théorie des graphes, et visant à optimiser la planification des tâches d'un projet. Semblable au PERT, les principales différences entre les deux méthodes reposent essentiellement dans la construction du graphe.

Les conditions préalables à la construction du graphe MPM

La méthode MPM suit une démarche logique qui impose au préalable de satisfaire les étapes suivantes :

- Etablir une liste des tâches à réaliser et déterminer la durée de chaque tâche.
- Pour chacune des tâches, déterminer les tâches précédentes (relations d'antécédence et de succession).
- Identifier les tâches dépendantes (qui ne peuvent commencer que si certaines autres tâches sont exécutées partiellement ou terminées).
- Identifier les tâches pouvant être réalisées simultanément (sous réserve d'une disponibilité des ressources nécessaires).

Modélisation en Graphe

Le graph MPM se présente tel comme suit : [13]

- Chaque tâche est représentée par un sommet, et les arcs entre les sommets traduisent uniquement les relations d'antériorité des tâches.
- chaque tâche (ou sommet) est renseignée sur la date à laquelle elle peut commencer au plus tôt (date de début au plus tôt) et terminer au plus tard (date de fin au plus tard) pour respecter le délai optimal de réalisation du projet.
- A chaque arc est associée une valeur numérique qui représente soit une durée d'opération, soit un délai, et la longueur des arcs n'est pas proportionnelle à cette durée.
- le graphe commence et termine sur 2 sommets, respectivement appelés « Début » et « Fin » symbolisant les début et fin des opérations. Ces deux sommets ne correspondent pas une tâche.

- Le graphe se lit de gauche à droite (du sommet "DÉBUT" à celui de "FIN").

2.4-Conclusion

Plusieurs modèles d'ordonnancement et de planification des tâches d'un projet ont été exposés ; le réseau PERT, le Diagramme de GANTT, et les modèles MPM et CPM. Les étapes nécessaires à la construction du réseau PERT ont été données en détail, la détermination des plus longs chemins est l'étape la plus difficile.

Nous allons aborder dans la suite la résolution par les méthodes approchées qui fournissent une solution acceptable en temps raisonnable.

3- Méthodes de résolution approchées

Ces méthodes sacrifient le caractère optimal de la solution pour obtenir, en un temps de calcul raisonnable, des solutions sous-optimales de bonne qualité. Ces méthodes reposent généralement sur un mécanisme de déplacement (aléatoire ou non) dans l'espace des solutions. Elles ne sont pas exactes, mais permettent en général d'obtenir des solutions proches de l'optimum.

3.1- Les heuristiques

Une heuristique [03] désigne un algorithme qui résout un problème d'optimisation donné, sans garantie d'optimalité mais dans des temps de calcul raisonnables (un exemple connu est les algorithmes gloutons).

Les algorithmes gloutons sont parmi les schémas heuristiques les plus simples et les plus rapides. Ils construisent une solution de manière itérative sans jamais remettre en cause les décisions prises à l'itération antérieure. Ces algorithmes construisent une solution élément par élément. A une itération donnée, on détermine l'élément à inclure dans la solution partielle en évaluant le coût de la nouvelle solution partielle qui inclut cet élément. L'élément engendrant la plus petite augmentation du coût est choisi (voir algorithme 1).

Malheureusement, il n'est pas possible de connaître, a priori, l'impact des décisions prises à une itération donnée sur le long terme. En outre, il est possible qu'à une itération donnée aucun élément ne soit insérable (par exemple à cause de contraintes qui se retrouvent violées), dans ce cas, l'algorithme échoue.

Algorithme 1 : L'algorithme glouton déterministe [03]

```
Sorties : S ou échec

1  échec ← faux;

2  S← ∅  ; (objet vide);

Tant que (S incomplète et échec = faux) faire
4       Construire la liste L des éléments insérables dans S;
5       Si L ≠ ∅ ; alors
6             Evaluer le coût incrémental des éléments de L;
7             Insérer dans S l'élément ayant le coût incrémental le plus faible
8       sinon
9             échec ←  vrai ;
```

3.2- Les Méta-Heuristique

Une méta-heuristique désigne un schéma algorithmique général qui peut s'appliquer à différents problèmes d'optimisation combinatoire [03]. Plus précisément, elle utilise des stratégies qui guident la recherche dans l'espace des solutions, ces stratégies étant indépendantes du problème auquel on les applique. Le but est d'explorer le plus efficacement possible l'espace des solutions afin de ne pas rester bloqué dans les minima locaux et de se diriger rapidement vers les régions les plus prometteuses. Il existe un grand nombre de méta-heuristiques allant de schémas très simples (qui mettent en œuvre des processus de recherche basiques, comme la descente), à des schémas beaucoup plus complexes (avec des processus de recherche élaborés comme les colonies de fourmis).

Il existe plusieurs méta-heuristique comme la recherche locale, le recuit simulé, la recherche tabou, l'optimisation par colonies de fourmis, les algorithmes génétique.

3.2.1- Algorithmes génétiques

Les algorithmes génétiques [05] s'inspirent de la théorie de l'évolution et des règles de la génétique qui expliquent la capacité des espèces vivantes à s'adapter à leur environnement par la combinaison des mécanismes suivants :

– **la sélection naturelle** : fait que les individus les mieux adaptes a l'environnement tendent à survivre plus longtemps et ont donc une plus grande probabilité de se reproduire ;

– **la reproduction par croisement** : fait qu'un individu hérite ses caractéristiques de ses parents, de sorte que le croisement de deux individus bien adaptes a leur environnement aura tendance à créer un nouvel individu bien adapte a l'environnement ;

– la mutation : fait que certaines caractéristiques peuvent apparaitre ou disparaitre de façon aléatoire, permettant ainsi d'introduire de nouvelles capacités d'adaptation à l'environnement, capacités qui pourront se propager grâce aux mécanismes de sélection et de croisement.

Les algorithmes génétiques reprennent ces mécanismes pour définir une méta-heuristique de résolution de problèmes d'optimisation combinatoire. L'idée est de faire évoluer une population de combinaisons, par sélection, croisement et mutation, la capacité d'adaptation d'une combinaison étant ici évaluée par la fonction objective à optimiser. La figure 1 présente un schéma de fonctionnement général de l'algorithme génétique. **[06]**

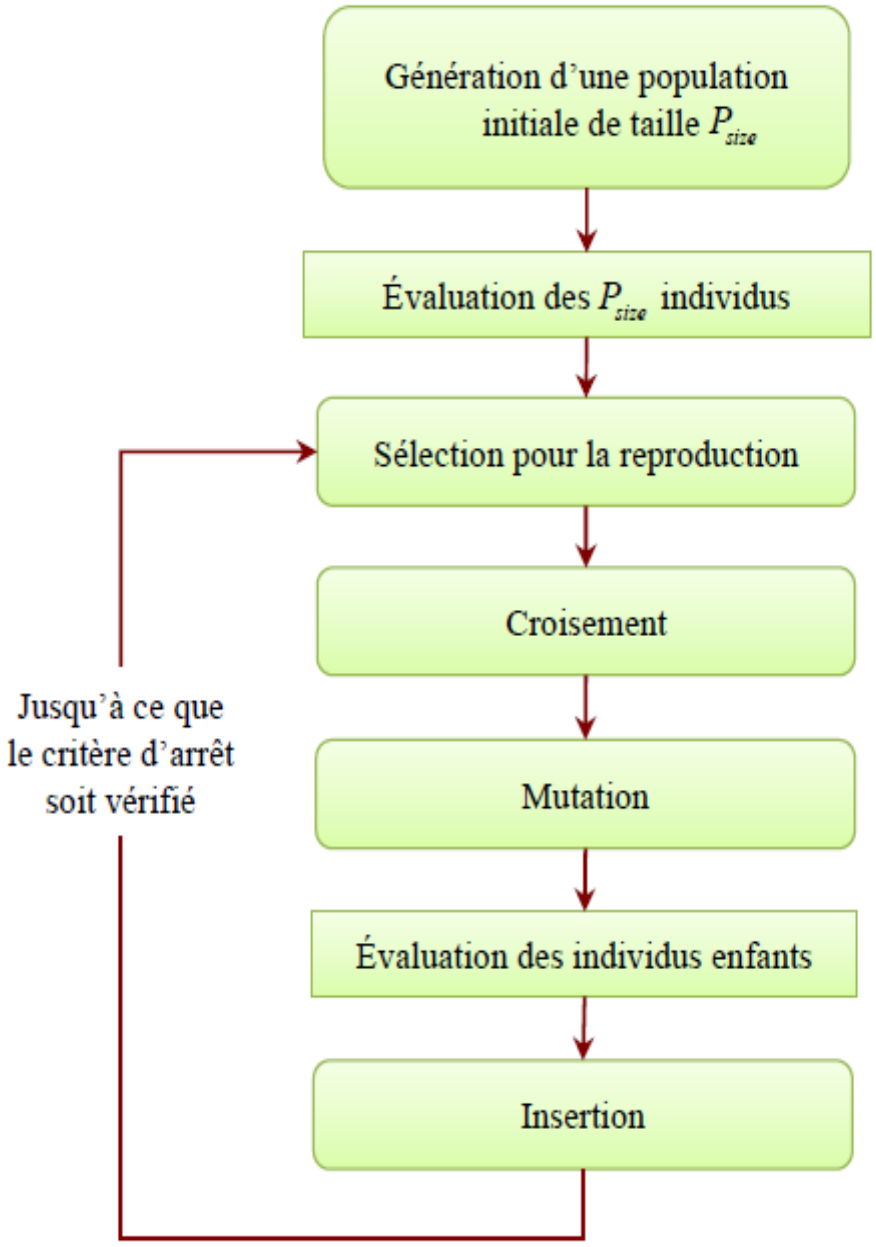


FIG .3- Fonctionnement générale de l'algorithme génétique

L'algorithme 2 décrit ce principe général, dont les principales étapes sont détaillées ci-après.

Algorithme 2 : *Algorithme génétique* ***[06]***

Initialiser la population avec un ensemble de combinaisons de E

Tant que« critères d'arrêt non atteints »faire

Sélectionner des combinaisons de la population

Créer de nouvelles combinaisons par croisement et mutation

Mettre a jour la population

Retourner la meilleure combinaison ayant appartenu à la population

Initialisation de la population : en général, la population initiale est générée de façon aléatoire, selon une distribution uniforme assurant une bonne, diversité des combinaisons.

Sélection : cette étape consiste à choisir les combinaisons de la population qui seront ensuite croisées et mutées. Il s'agit là de favoriser la sélection des meilleures combinaisons, tout en laissant une petite chance aux moins bonnes combinaisons. Il existe de nombreuses façons de procéder à cette étape de sélection. Par exemple, la sélection par tournoi consiste à choisir aléatoirement deux combinaisons et a sélectionner la meilleure des deux (ou bien a sélectionner une des deux selon une probabilité dépendant de la fonction objectif).

Croisement : cette opération consiste à générer de nouvelles combinaisons, à partir des combinaisons sélectionnées. Là encore, il existe de nombreux opérateurs de croisement. Une méthode simple consiste à choisir aléatoirement un point de croisement, à couper chaque combinaison parente en ce point, puis à reformer deux enfants en échangeant les parties composant les parents de part et d'autre du point de croisement.

Mutation : cette opération consiste à modifier de façon aléatoire certains composants des combinaisons obtenues par croisement.

Mise à jour de la population : cette étape consiste à remplacer certaines combinaisons de la génération précédente par certaines combinaisons issues des opérations de croisement et de mutation, formant de la sorte une nouvelle génération. Là encore, il existe différentes stratégies de remplacement, favorisant plus ou moins la diversité, et plus ou moins élitistes.

On peut par exemple choisir de ne garder que les meilleurs individus, qu'ils soient issus de la nouvelle génération ou de l'ancienne, ou bien ne garder que les individus de la nouvelle génération, indépendamment de leur qualité.

Critères d'arrêt : le processus d'évolution est itéré, de génération en génération, jusqu'a ce qu'une combinaison de qualité suffisante soit générée, ou bien jusqu'a ce qu'une limite de temps soit atteinte.

Analogies entre la génétique naturelle et les problèmes d'ordonnancement [06]

Génétique Naturelle	*Problèmes d'ordonnancement*
Population	Ensemble de solutions : ensemble des ordonnancements faisables
Individu (chromosome = séquence finie de gènes) parmi la population	Une solution : un ordonnancement parmi les ordonnancements faisables
Gene (une suite d'allèles)	Une tâche de l'ordonnancement (l'ensemble de paramètres codant la tâche)
Allèle	Un paramètre attribué à la tâche (ex : temps de traitement, position de la tâche dans la séquence ment des opérations)

Tableau 4 : Analogie entre la génétique et les problèmes d'ordonnancement

3.2.2-Recuit Simulé

Le *recuit simulé* est une méthode inspirée de principes de thermodynamique et de processus utilisés en métallurgie, qui alternent des cycles de refroidissement lent et de réchauffage tendant à minimiser l'énergie du matériau. Appliquée à l'optimisation, elle permet de trouver les extrema d'une fonction.

Algorithme 3 : Schéma algorithmique du recuit simulé [07]

Entrées : θ_0 // meilleur solution trouvée

Sorties : S*

Initialiser une solution S ;

$\theta \leftarrow \theta_0$;

Tant que ' critère d'arrêt non atteint ' faire

Choisir S' voisin de S ;

Si f(S') <f(S) alors

S ← S' ;

Sinon

Si exp ($-\frac{f(S')-f(S)}{\theta}$) ≥p // p : nombre aléatoire entre 0 et I

Alors

S ← S' ;

Faire décroître θ ;

S* ← la meilleure solution trouvée ;

Elle fonctionne comme suit. Initialement on fixe la température θ à une température donnée θ_0 et on génère une solution S à l'aide d'une heuristique quelconque ; S devient la solution courante. A chaque itération on choisit une solution S0 dans un voisinage de S, si S0 est meilleure que S alors S0 devient la solution courante sinon S0 devient la solution courante avec une probabilité dépendante de θ et de la différence de coût entre S et S0 (voir algorithme 3). Plus la température est haute, plus la probabilité d'accepter une transformation qui dégrade la solution courante est élevée. Au fur et à mesure des itérations, la température diminue de sorte qu'il devient de moins en moins probable d'accepter une solution dégradante. L'efficacité de cet algorithme dépend bien sûr, entre autre, de la stratégie adoptée pour faire décroître θ.

3.2.3 Recherche Taboue

1-Principe

La recherche taboue est également une des méta-heuristiques les plus connues, elle a été introduite par Glover en 1986 [02]. Elle utilise un historique de manière à interdire à l'algorithme de revenir sur ses pas. Cet historique se traduit par la présence d'une liste dite

tabou qui garde une trace des dernières solutions visitées, ainsi l'algorithme ne pourra plus explorer ces solutions (du moins à court terme, tout dépend de la taille de la liste tabou).

Dans le cas d'un problème d'ordonnancement, la liste taboue peut contenir, par exemple, les informations suivantes : la solution (ordonnancement +affectation) le cout de la solution et les attributs de mouvements utilisés pour aboutir à cette solution.

Les systèmes de voisinage utilisés sont le changement de deux pièces contigües, deus pièces quelconque et insertion d'une pièce dans une autre position.

L'algorithme fonctionne comme suit. Initialement la liste taboue est vide et on génère une solution S à l'aide d'une heuristique quelconque ; S devient la solution courante. A chaque itération, on choisi le meilleur voisin S0 de S qui n'est pas déjà dans la liste tabou, S0 devient la solution courante S et est ajouté à la liste tabou (voir algorithme 4). Si la taille de la liste taboue dépasse la taille maximale autorisée, on supprime de cette liste l'élément le plus ancien (stratégie FIFO - First In First Out). Notons que la liste taboue permet d'éviter les cycles en interdisant de choisir une solution dans le voisinage de la solution courante qui aurait déjà était explorée. En outre, en pratique, on ne stocke pas les solutions dans leur intégralité (trop coûteux en temps et en espace) mais seulement une signature de ces solutions.

Les Tabous

La définition et la gestion des tabous jouent bien entendu un rôle primordial dans la RT. Comme nous l'avons plus haut, ceux-ci correspondent à une sorte de mémoire à court terme qui indique à la procédure où elle a déjà été, lui permettant donc de diriger son exploration vers des régions du domaine non encore visitées. [10]

Mouvement tabou : un mouvement non souhaitable, comme si on redescendait à un minimum local d'où on vient juste de s'échapper.

Mouvement non améliorateur : un mouvement qui nous sortirait d'un minimum local S* en nous amenant à une solution voisine S' pire que l'actuelle.

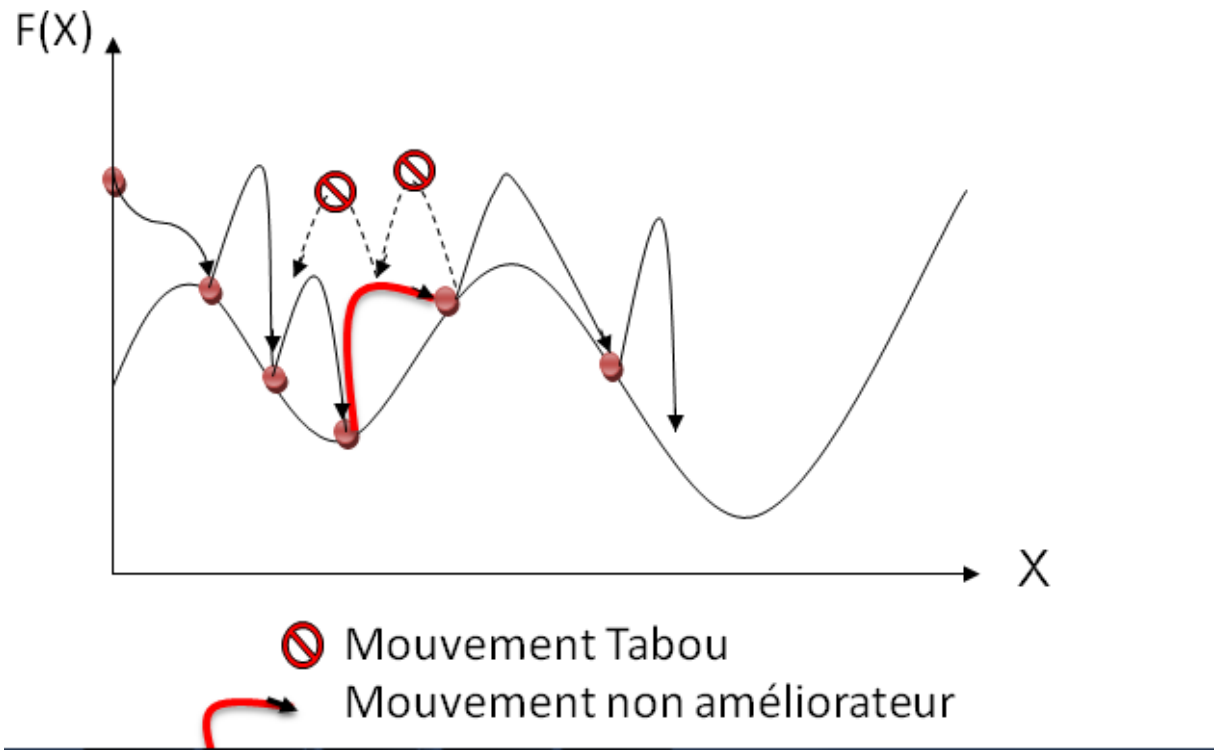


FIGURE 5 : principe de Tabou

2-Algorithme

Algorithme 4 : Schéma algorithmique de la recherche tabou [02]

(1) Générer une solution initiale S de manière aléatoire

(2) $S^* \leftarrow S$; $C^* \leftarrow f(S)$ // S^* la meilleure solution rencontrée, C^* est sont cout, et f la fonction objectif

La liste taboue est vide ($T=\emptyset$)

(3) Ajouter S à la liste tabou ; k=0 ;

(4) **Tant que** << critère d'arrêt non atteint >> **Faire**

(5) choisir parmi le voisinage de S_k, $V(S_k)$, le mouvement qui minimise f et qui n'appartient pas à la liste taboue, meilleur S_k.

(6) $S_{k+1} \leftarrow$ meilleur(S_k)

(7) Si la liste taboue est pleine alors

Remplacer le dernier élément de la liste taboue par S_{k+1}

Sinon

Ajouter S_{k+1} à la liste taboue

Fin Si

Si ($C(S_{k+1}) < C^*$) alors

$S^* \leftarrow S_{k+1}$, $C^* = C(S_{k+1})$;

Fin Si

Fin Tant que

FIN ALGORITHME

3-Diverses améliorations

A partir de cet algorithme initial, certaines adaptations ont été élaborées a fin de pallier à des problèmes constatés dans l'analyse de l'exploration de l'espace de recherche on recense toutes ces techniques :

- La stratégie d'intensification
- La stratégie de diversification
- La stratégie d'aspiration

La stratégie d'intensification

Il s'agit de répéter les éléments faisant partie des meilleures solutions trouvées, qui seront utilisées pour générer de nouvelles solutions.

La stratégie de diversification

D'une manière symétrique, lorsque le processus de recherche parcourt une branche sur une langue période, il est possible de le stopper et de diversifier la recherche sur une autre zone de l'espace.

L'algorithme reprend généralement sur une autre solution générée aléatoirement.

La stratégie d'aspiration

L'utilisation de points tabous peut empêcher, dans certains cas, la méthode taboue d'atteindre une solution intéressante, le critère d'aspiration a été introduit par Glover à cet effet.

Il consiste à enlever le statut tabou associé à une transformation si celle-ci permet d'aboutir à une solution meilleure que toutes les solutions trouvées jusqu'à présent.

Mais ce critère ne ce limite pas à ce cas particulier et il est «également possible d'utiliser une fonction d'aspiration, dont le but est de toujours aller d'une solution à une solution meilleure.

3.3- Comparaison des méthodes

Le tableau suivant résume la comparaison des méthodes [10]

	A. du Recuit Simulé	Méthode taboue	A. génétique	A. à colonies de fourmis
Facilité d'adaptation		-	-	-
Connaissance		+	+	+
Qualité	+	++	+	+++
Rapidité	-	-	--	--

Figure 5 : comparaison des méthodes

Les critères de comparaison retenus sont les suivants :

- Faciliter d'adaptation au problème
- Possibilité d'intégrer des connaissances spécifique au problème
- Qualité des meilleures solutions trouvées
- Rapidité c'est-à-dire temps de calcul nécessaire pour trouver une telle solution

VI- Conclusion

Dans ce travail, on a traité les problèmes d'ordonnancement. La majeure partie concerne le problème d'ordonnancement de projet à moyens limités (RCPSP).

On a également abordé la résolution d'une extension du RCPSP consistant à prendre en compte des ressources particulières, qui peuvent être consommées en début d'exécution de chaque activité, mais aussi produites à leur fin : il s'agit du RCPSP avec consommation et production de ressources. Afin d'effectuer une comparaison expérimentale entre différents modèles, on a proposé une adaptation de nos formulations basées événements, des formulations à temps discret de Pritsker et de Christofides, et de la formulation à temps continu basée sur les flots (proposé par Artigues). Globalement, les résultats montrent que les formulations basées événements obtiennent de meilleurs résultats sur un bon nombre de types d'instances.

Dans une seconde partie on a également proposé des méthodes de résolution exacte comme le Diagramme de GANTT, La Méthode PERT, la méthode de potentiel métra (MPM), et quelques heuristique et méta-heuristique comme les algorithmes gloutons, les algorithmes génétiques, la recherche tabou etc……

Bibliographie et Web graphie

❖ Bibliographie

[1] EL Hilali Alaoui Ahmed et al, « Initiation à la recherche opérationnelle »,2009

[2] P. Soriano M. Gendreau « fondement et application des méthodes de recherche tabou » 1997.

[3] Oumar koné, doctorat de l'université de toulouse, Titre : « nouvelles approches pour la résolution du problème d'ordonnancement de projet à moyens limités » 7 décembre 2009

[4] Sophie demassey « Thèse : méthodes hybrides de programmation par contraintes et programmation linéaire pour le problème d'ordonnancement de projet à contrainte de ressource » 18 décembre 2003.

[5] Edmond Maurel, Daniel Roux et Daniel Dupont « Techniques Opérationnelles d'Ordonnancement » mars 1977 - 342 pages Edition. EYROLLES.

[6] Joseph Liouville « Méthodes d'Optimisation– Page 25 jusqu'à 57. Laboratoire de Mathématiques Pures et Appliquée Université du Littoral » Septembre 2011.

[7] Groupe Gotha - Modèles et algorithmes de recherche opérationnelle - Ellpises 2004.

[8] F. Roubellat, G. Fontan «potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités » 1979.

[9] Lucas Grèze et al « une méthode heuristique pour l'ordonnancement de projets avec contraintes de ressources et chevauchement d'activité » CIRRELT-2012-18.

❖ Web graphie

[10] http://www.sews-cabind.it/it

[11] http://www.gantt.com/fr/

[12] http://ressources.aunege.fr/nuxeo/site/esupversions/2b1c56b6-109d-488a94a33ea525f8beef /ModAidDec/cours/l5/l5.pdf

[13] http://www.logistiqueconseil.org/Articles/Logistique/Methode-potentiel-metra.htm

[14] https://aurga.wordpress.com/2013/04/08/la-methode-du-chemin critique-critical-path-method-en-anglais-ou-cpm/