**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**
**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**
**UNIVERSITE MOHAMED BOUDIAF - M'SILA**

**FACULTE** Mathématiques et Informatique
**DEPARTEMENT** Informatique
**N° :**……..

**DOMAINE :** Mathématiques et Informatique
**FILIERE :** Informatique
**OPTION :** OMID

**Mémoire présenté pour l'obtention**
**Du diplôme de Master Académique**

**Par**: BOUREZG et Souheyla

**Intitulé**

# Etude sur l'algorithme de Yuvan Cohen pour la réduction des taches fictives dans le réseau PERT

**Soutenu devant le jury composé de :**

| | | |
|---|---|---|
| …… | Université de M'sila | Président |
| Dr. N.E.MOUHOUB | Université de M'sila | Rapporteur |
| …… | Université de M'sila | Examinateur |

**Année universitaire : 2016 /2017**

## REMERCIEMENT

Pour commencer, je veux adresser mes remerciements à mon directeur de mémoire, **Dr. NASSER EDDINEMOUHOUB** pour sa grande disponibilité et ses encouragements tout au long de la rédaction de ce mémoire.

Enfin, j'adresse mes remerciements à ma famille qui m'ont soutenue durant toutes mes années d'études et qui m'ont toujours encouragée à aller plus loin, mes collègues pour la qualité de leur travail et leur bonne humeur à toutes les saisons scolaires.

# TABLE DES MATIERES

## CHAPITRE 02

## LE PROBLEME CENTRAL DE L'ORDONNANCEMENT

## CHAPITRE 03

## L'ALGORITHME DE COHEN ET SADEH POUR LA CONSTRUCTION DU GRAPHE AOA



## CHAPITRE 04

## L'OPTIMISATION DE L'ALGORITHME DE COHEN



## CHAPITRE 05

## IMPLEMENTATION

# LISTE DES TABLEAUX

**Nom de table** **N° page**

# LISTE DES FIGURES

**Nom de figure** **N° page**

# INTRODUCTION GENERALE

## 1. Introduction

La recherche opérationnelle (RO) est la discipline des mathématiques appliquées qui traite des questions d'utilisation optimale des ressources dans l'industrie et dans le secteur public. Depuis une dizaine d'années, le champ d'application de la RO s'est élargi a des domaines soit stratégiques comme le choix d'investir ou pas, le choix d'une implantation, le dimensionnement d'une flotte de véhicules ou d'un parc immobilier, etc… ou opérationnels comme l'ordonnancement, la gestion de stock, l'affectation de moyens (humains ou matériels) à des tâches, les prévisions de ventes…

La gestion de projet est une composante très importante de la communauté de recherche opérationnelle. De nombreux travaux traitent de l'ordonnancement et de la gestion de projet, mais aussi de logistique, de planification, et de problèmes d'emploi du temps.

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises. Alors il est défini par le planning d'exécution des tâches (« ordre » et « calendrier ») et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs. Pour résoudre un problème d'ordonnancement il faut passer par l'étape de modélisation.

La modélisation est une étape importante dans les domaines de recherche opérationnelle, car elle est la conception d'un modèle. Selon son objectif et les moyens utilisés, elle est dite mathématique, géométrique, informatique...

Un modèle est une traduction de la réalité pour pouvoir appliquer des outils, des techniques et des théories mathématiques, puis en traduisant les résultats obtenus en prédictions ou opérations dans le monde réel.

Donc, pour résoudre un problème il faut les modéliser. Parmi les vastes types de modélisation on cite la modélisation graphique qui peut représenter la structure, la connexion (la relation entre ses éléments), la modélisation graphique constitue une méthode de pensée qui permet de modéliser une grande variété de problèmes sous forme de sommets et d'arcs. L'informaticien peut faire le graphe d'une manière correcte. La modélisation graphique permet donc la compréhension du problème, mais pas sa résolution.

Alors, pour résoudre un problème il faut trouver une solution dans l'espace algorithmique. Nombreux sont les outils de théorie de graphe pour décrire les algorithmes, les étudier, exprimer leurs qualités, pouvoir les comparer entre eux.

## 2. Problématique

Le problème qui se pose est de comprendre l'algorithme de Cohen et Sadeh et de réaliser une étude bien détaillée sur ses étapes pour trouver la pénurie qui ne permet pas l'accès à un graphe PERT AoA minimal.

## 3. Objectifs

Notre objectif à travers ce travail est de comprendre l'algorithme de « Cohen & Sadeh » et d'essayer de l'améliorer pour donner un graphe PERT AoA avec le nombre minimal de tâches fictives.

## 4. Organisation du mémoire

La mémoire est organisée en cinq parties :

- Une introduction générale.
- Chapitre 01 « L'optimisation combinatoire » : Dans ce chapitre, nous allons donner des définitions de l'optimisation des algorithmes, des problèmes d'optimisation combinatoires et des problèmes de décisions et leur complexité. Ensuite, nous aborderons l'étude de la fonction d'ordonnancement avec ses éléments de base ainsi que ses caractéristiques générales. Enfin, nous conclurons le chapitre par la présentation des méthodes de résolution d'ordonnancement.
- Chapitre 2 « Le problème central de l'ordonnancement » : Dans ce chapitre, nous présenterons le projet et la gestion de projet, les problèmes central de l'ordonnancement ou on présente les trois méthodes de représentation : Gantt, AoN, AoA, et la résolution de ce genre de problèmes.
- Chapitre03 « L'algorithme de Cohen et Sadeh » : Dans ce chapitre, nous présenterons l'algorithme de Cohen et ses étapes bien détaillées, avec une application d'algorithme sur quelques graphes.

- Chapitre04 « Le graphe AoA minimal » ： Dans ce chapitre, nous traiterons de l'état de l'art concernant la construction du graphe AoA minimal. En outre, nous présenterons l'algorithme de Cohen sur un exemple. Ensuite, nous proposerons une modification de l'algorithme pour améliorer le graphe AoA Pert avec un nombre minimal de tâches fictives.
- Chapitre05 « Implémentation » : Nous présenterons une application qui propose de résoudre le problème de cette mémoire, à l'aide de langage de programmation C# (Sharp).
- Enfin on termine notre mémoire avec une conclusion générale de notre travail où nous répondrons à notre problématique présentée dans cette introduction et présenterons notre contribution personnelle suivie des perspectives futures qui tracent le chemin des travaux futurs.

# CHAPITRE 01

# L'OPTIMISATION COMBINATOIRE

## 1. Introduction

L'informatique permet d'avoir un outil d'aide à la décision et aussi aux planifications de la production. Cette planification est d'autant plus efficace que l'algorithme d'ordonnancement qu'elle utilise est performant. Le but étant d'atteindre un ordonnancement optimal qui répartit au mieux la charge du travail et tient compte des diverses contraintes de production.

Les problèmes d'ordonnancement sont très variés. On peut les rencontrer dans de très nombreux domaines : les systèmes industriels de production « activités des ateliers en gestion de production et problèmes de logistique », les systèmes informatiques « les tâches sont les programmes et les ressources sont les processeurs, la mémoire... », Les systèmes administratifs « Gestion du personnel, emplois du temps,… », Les systèmes de transport, la construction, etc…. C'est pour cette raison qu'ils ont fait et continuent de faire l'objet de nombreux travaux de Recherche [16].

## 2. La recherche opérationnelle

La recherche opérationnelle peut être définie comme l'ensemble des méthodes et techniques rationnelles orientées vers la recherche du meilleur choix dans la façon d'opérer en vue d'aboutir au résultat visé ou au meilleur résultat possible.

Elle fait partie des *« aides à la décision* » dans la mesure où elle propose des modèles conceptuels en vue d'analyser et de maitriser des situations complexes pour permettre aux décideurs de comprendre et d'évaluer les enjeux et d'arbitrer ou de faire les choix les plus efficaces.

Ce domaine fait largement appel au raisonnement mathématique (logique, probabilités, analyse des données) et à la modélisation des processus. Il est fortement lié à l'ingénierie des systèmes, ainsi qu'au management du système d'information [17].

### 2.1 Problèmes d'optimisation

Les graphes sont un outil de modélisation puissant et très intuitif. Ils sont naturellement utilisés pour représenter des réseaux de transport, de communication, et plus généralement des flux de matières ou d'informations. Ils sont également utilisés pour modéliser des problèmes dans lesquels des objets sont en relation, les sommets du graphe représentant ces objets et les

arêtes (ou arcs si une orientation est considérée), les relations entre ces objets [22]. Le problème d'optimisation considéré est modélisé par les outils de la théorie des graphes ensuite une solution est cherchée qui est généralement un algorithme.

Parmi les problèmes modélisés par la théorie des graphes, on cite : le problème d'affectation, de flot maximum ou minimum, de transport, de sac à dos, de voyageur de commerce, de localisation et d'ordonnancement, etc…. Ce dernier regroupe l'ordonnancement dans la gestion de projet et l'ordonnancement dans la gestion de la production. [18]

**2.1 .1** Les problèmes d'optimisation combinatoires

Un problème d'optimisation combinatoire *(POC)* est un problème de minimisation ou de maximisation auquel on associe un ensemble d'instances.

Les problèmes d'optimisation combinatoire sont présents dans plusieurs domaines d'applications industrielles, économiques et scientifiques. On les retrouve dans l'ordonnancement de tâches dans un système de production. Ce sont, en général, des problèmes faciles à définir, mais difficiles à résoudre [16].

**2.1.2** Problème d'optimisation combinatoire et Problème de décision

Un problème d'optimisation combinatoire est un problème qui consiste à chercher une meilleure solution parmi un ensemble de solutions réalisables.

Un problème de décision est un problème qui consiste à apporter une réponse "oui" ou "non" à une question.

Un problème d'optimisation combinatoire est au moins aussi difficile que le problème de reconnaissance associé. De plus, on peut généralement prouver que le problème de décision n'est pas plus facile que le problème d'optimisation combinatoire. En d'autres termes, cela signifie qu'un problème d'optimisation combinatoire est souvent du même niveau de difficulté que le problème de décision associé. [21]

## 3. Complexité

L'expérience montre que certains problèmes sont plus faciles que d'autres à résoudre.

Une théorie de la complexité a été développée et permet mathématiquement de classer les problèmes faciles et difficiles en deux classes : les classes P et NP.

On suppose que, pour chaque problème que l'on veut résoudre, l'on dispose d'une mesure de la taille du problème. Par exemple, on utilisera comme mesure le nombre de tâches lorsqu'il s'agit d'un problème d'ordonnancement de *n* tâches. Le nombre d'opérations élémentaires effectuées par un algorithme est donc une fonction qui dépend de *n*. Cette fonction s'appelle la complexité. [19]

Déterminer la complexité d'un algorithme donné n'est pas toujours si simple.

Déterminer la complexité d'un problème apparaît moins ambitieux, puisqu'il s'agit de déterminer quelle est la plus faible complexité d'un algorithme de résolution. [11]

## 4. Complexité d'un algorithme

On désigne par complexité d'un algorithme, le nombre d'opérations nécessaires à celui-ci pour s'exécuter.

Bien évidemment, ce nombre peut varier en fonction de ce que l'on appelle les données d'entrées, c'est-à dire les paramètres que l'on donne à l'algorithme.

On mesure l'efficacité d'un algorithme par une expression mathématique qui indique le nombre d'opérations indispensables à l'exécution de l'algorithme en fonction de la taille des données en entrées tout en supposant le pire des cas. [21]

## 5. L'algorithme efficace

La complexité d'un algorithme est notée par *O*. Ainsi, si le nombre d'itérations nécessaires pour obtenir une solution optimale est décrit par $4n3 – 6n2 + 5n – 1$, l'algorithme est dit en $O(n 3)$, ce qui caractérise une complexité polynomiale en la taille du problème. Par contre si la complexité est de l'ordre $O(2n)$ on parle de complexité exponentielle. Un algorithme sera dit efficace si sa complexité est bornée par un polynôme ayant la taille des données comme variable.

Cependant, il faut bien comprendre que l'on s'intéresse au comportement général de l'algorithme face à des problèmes de grande taille.

Ainsi, ce n'est pas utile de compter toutes les opérations dans le détail, ni de considérer le langage de programmation.

## 6. Problèmes faciles et problèmes difficiles

Tout d'abord, on fait une distinction entre les problèmes décidables et les problèmes indécidables**,** les problèmes décidables sont ceux pour lesquels il existe au moins un algorithme pour les résoudre, Les problèmes indécidables sont ceux pour lesquels aucun algorithme, quel qu'il soit, n'a été trouvé pour les résoudre. [21]

### 6.1 La classe P

Un Problème est dit appartenant à la classe P s'il existe un algorithme polynomial pour le résoudre. On dit que les problèmes de la classe P sont faciles.

### 6.2 La classe NP

La classe des problèmes NP (NP pour Non détermination Polynomiale) est la classe des problèmes des décisions qui peuvent être résolus par une machine de Turing non déterministe en temps polynomial. Parmi la classe de problèmes NP, on distingue deux grandes classes:

- La classe des problèmes polynomiaux (la classe P) et la classe des problèmes NP-Complets (la classe NPC).

### 6.3 La classe NP-Complet

Un problème de décision est dit NP-Complet si tout problème de la classe NP peut se rendre polynomialement à lui.

### 6.4 La classe NP-difficile

Tous les problèmes NP-difficiles ne sont pas de difficulté identique. On rencontre par exemple des problèmes qui ne peuvent pas être résolus en temps polynomial avec un codage binaire, mais qui peuvent l'être avec un codage unaire, ces problèmes sont dits NP-difficiles au sens ordinaire ou simplement NP- difficiles, les algorithmes pour cette classe de problèmes sont appelés pseudo-polynomiaux, pour d'autres problèmes, on peut ne pas trouver d'algorithme polynomial, quelque soit le codage, ceux-là sont dits NP- difficiles au sens fort. [21]

<table>
<tr><td rowspan="4">Décidable</td><td rowspan="3">Classe NP</td><td>Classe P (plus court chemin, arbre de poids min, flot maximum …)</td><td rowspan="4">Classe NP-Difficile</td></tr>
<tr><td></td></tr>
<tr><td>Classe NP-Complet (PVC, sac à dos…)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Indécidable</td><td></td><td></td><td></td></tr>
</table>

**Tableau 1.1** Tableau récapitulatif de classification des problèmes combinatoires

# 7. La fonction ordonnancement

## 7.1 Définition

Selon [6] « Ordonnancer c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution. » Un ordonnancement donc, selon [14] constitue une solution au problème d'ordonnancement. Il décrit l'exécution des tâches et l'allocation des ressources au cours du temps et vise à satisfaire un ou plusieurs objectifs.

L'ordonnancement apparaît dans tous les domaines de l'économie: l'informatique, la construction, l'industrie et l'administration.

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches. Pour cela, il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches.

Un problème d'ordonnancement peut être considéré comme un sous-problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches planifiées.

Le mariage des deux approches n'est que bénéfique pour la résolution de ce genre de problèmes.

Les paragraphes suivants précisent ces notions de tâche, ressource, objectif,… et introduisent quelques notations.

## 7.2 Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et une date de fin d'exécution et qui consomme des ressources avec des quantités déterminées. Un coût (ou poids) est attribué à une tâche pour estimer sa priorité, son degré d'urgence, ou son coût d'immobilisation dans le système.

Une tâche qui peut être exécutée par morceaux est appelée tâche morcelable, le problème constitué de tâches morcelables est appelé problème préemptif. Si une tâche, une fois qu'elle démarre l'exécution, ne peut pas être interrompue, on dit que le problème est non préemptif [20].

## 7.3 Les ressources

La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche et disponible en quantité limitée.

Plusieurs types de ressources sont à distinguer :

Une ressource est renouvelable si après avoir été allouée à une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'équipement en général), la quantité de ressource utilisable à chaque instant est limitée [25].

### 7.4 Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre certaines variables. On distingue deux types de contraintes : les contraintes temporelles et les contraintes de ressources.

**7.4.1** Les contraintes temporelles

Les contraintes temporelles comprennent les contraintes de temps alloué, qui correspondent généralement aux impératifs liés aux tâches (délais de livraisons, disponibilité des approvisionnements) ou encore à la durée totale d'un ordonnancement.

Elles comprennent les contraintes d'antériorité ou de précédence qui correspondent à des contraintes de cohérence technologique qui positionnent les tâches les unes par rapport aux autres. Elles comprennent aussi les contraintes de calendrier qui correspondent, par exemple, aux plages horaires de travail, etc.

**7.4.2** Les contraintes de ressources

Les contraintes de ressources, quant à elles, traduisent la disponibilité des ressources et le fait qu'elle soit en quantité limitée. Deux types de contraintes de ressources, liées à la nature cumulative ou disjonctive des ressources, peuvent alors être distingués.

Les ressources disjonctives ne peuvent être utilisées que par une tâche à la fois. Les ressources cumulatives, quant à elles, peuvent être utilisées par plusieurs tâches simultanément, comme dans le cas d'un ensemble de ressources. [25]

L'ordonnancement de projet consiste à placer dans le temps les tâches d'un projet en respectant ses contraintes. Les plus répandues des techniques d'ordonnancement en gestion de projet sont basées sur une modélisation du problème du type AoN ou AoA. Dans ce cadre là, ces techniques visent à donner un plan pour l'exécution du projet, c'est-à-dire trouver une affectation complète des dates de début des tâches.

## 9. Méthodes de résolution des problèmes d'ordonnancement

Les méthodes d'optimisation peuvent être réparties en deux grandes classes de méthodes pour la résolution des problèmes :

- Les méthodes exactes.
- Les méthodes approchées.

### 9.1 Les méthodes exactes

Le principe des méthodes exactes consiste à rechercher, souvent de manière implicite, une solution, la meilleure solution ou l'ensemble des solutions d'un problème.

L'optimisation exacte concerne toutes les méthodes permettant d'obtenir un résultat dont on sait qu'il est optimal à un problème précis. Cela va des méthodes du simplex aux méthodes de Lagrangien en passant par la programmation dynamique. On peut classer les méthodes exactes en quatre grandes classes :

- La programmation dynamique.
- La programmation linéaire continue ou en nombres entiers.
- La programmation non linéaire avec ou sans contraintes.
- Les méthodes de recherche arborescente (Branch & Bound) [15].

**9.2 Les méthodes approchées**

Les méthodes approchées fournissent une solution approchée au problème traité. Elles sont en général conçues de manière à ce que la solution obtenue puisse être proche de la valeur optimale : de telles méthodes permettent d'obtenir des bornes inférieures ou supérieures de la valeur optimale telles que :

- Méthodes Heuristiques.
- Méthodes Métaheuristiques.

**9.2.1** Les méthodes Heuristiques

Une heuristique est plutôt une méthode qui cherche (une stratégie) sans garantir le résultat. Destiné à un problème spécifique, le temps de calcul est raisonnable sans garantir la faisabilité ou l'optimalité. [01]

En recherche opérationnelle, les heuristiques sont des règles empiriques simples qui ne sont pas basées sur l'analyse scientifique (différents algorithmes). Elles sont basées sur l'expérience, les résultats déjà obtenus et sur l'analogie pour optimiser les recherches suivantes. Généralement, on n'obtient pas la solution optimale mais une solution approchée. Parmi les heuristiques, on peut citer l'algorithme glouton. [05]

**9.2.2** Les méthodes Métaheuristiques

Une métaheuristique est constituée d'un ensemble de concepts fondamentaux (par exemple, la liste taboue et les mécanismes d'intensification et de diversification pour la métaheuristique tabou), qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation. Ainsi les Métaheuristiques sont adaptables et applicables à une large classe de problèmes. [20]

Les métaheuristiques se subdivisent en deux sous-classes :

- Les méthodes de voisinage.
- Les algorithmes évolutionnaire.

## 10. Conclusion

La majorité des problèmes d'ordonnancement sont NP-complets, ça veut dire que, dans la pratique, la complexité croît exponentiellement avec le nombre de tâches et de ressources. Il n'est pas donc envisageable de résoudre de tels problèmes avec les méthodes exactes. C'est pour cela qu'il faut développer des heuristiques dont l'objectif est de fournir des solutions aussi proches que possible de la solution exacte en un temps raisonnable.

# CHAPITRE 02

# LE PROBLEME CENTRAL DE L'ORDONNANCEMENT

## 1. Introduction

L'ordonnancement dans un projet consiste à placer dans le temps les tâches du projet en respectant ses contraintes. Les plus répandues des techniques d'ordonnancement en gestion de projet sont basées sur une modélisation du problème du type potentiel-tâches (AoN) ou potentiel- étapes (AoA).

## 2. Qu'est-ce qu'un projet?

C'est une démarche spécifique qui permet de structurer méthodiquement et progressivement une réalité à venir [22]

Un projet est défini et mis en œuvre pour élaborer une réponse au besoin d'un utilisateur, d'un client ou d'une clientèle, et il implique un objectif et des actions à entreprendre avec des ressources données.

Un projet est donc caractérisé par :

- l'introduction d'une novation dans un système existant ou créer un nouveau système,
- l'organisation temporaire, c'est-à-dire avoir un début et une fin, marqués par des évènements identifiables,
- une action spécifique et nouvelle qui structure une réalité à venir pour laquelle on n'a pas encore d'équivalent exact.

En outre, un projet présente le plus souvent une grande complexité et fait intervenir des disciplines multiples, étrangères les unes aux autres, dont il faut coordonner les activités parfois contradictoires. [22]

## 3. C'est quoi la gestion de projet?

On peut presque considérer la gestion de projet comme une discipline universelle puisqu'elle englobe un large éventail d'activités : de l'élaboration de nouveaux produits et services, à l'organisation de conférences et d'ateliers, en passant par la modernisation de bureaux d'une entreprise. Si on dispose d'un budget illimité et de tout temps nécessaire, la gestion de projet serait plutôt facile. En réalité, le temps et l'argent sont des facteurs déterminants et, par conséquent, la gestion de projet devient une compétence importante à maîtriser [10]

Selon [CGR 08], dans la vie d'aujourd'hui où l'on compte une infinité de projets qui se réalisent, on remarque que ces projets n'atteignent pas souvent leurs objectifs à cause du dépassement de délais (surcoûts importants, ou bien la qualité technique du produit est insuffisante). Notons également que les projets se déroulent dans un milieu complexe (acteurs divers dans une entreprise : étude, production, marketing, environnement extérieur non maîtrisable : marché, social, politique, concurrence, etc.). C'est pour ces raisons qu'il faut avoir une démarche claire et rigoureuse pour la gestion d'un projet.

## 4. Caractéristiques d'un projet

Un projet réussi doit contenir les particularités suivantes :

- *Des objectifs précis :* Les projets les plus réussis ont des objectifs définis clairement dès le départ.
- *Un plan de projet bien établi :* Un plan conçu avec minutie est utile pour deux raisons. D'abord, il permet à chaque participant de comprendre et de contribuer au projet. Il précise les responsabilités de chacun et évalue combien d'argent, de personnes, de matériel et de temps sont nécessaires à l'achèvement du projet. Ensuite, il sert d'outil de suivi et permet d'adopter des mesures correctives tôt dans le processus si les choses tournent mal.
- *La communication :* Le projet repose sur la collaboration entre toutes les personnes qui y prennent part. Une communication réelle et continue doit s'établir entre les parties, si elles veulent œuvrer ensemble à la réussite du projet.
- *Une envergure maîtrisée :* Tout au long du projet, le chef fait face à plusieurs situations qui ne contribuent pas toutes aux objectifs tracés. Il importe que le chef de projet porte attention à ses priorités, avec une perte minimale de temps et de concentration.

- *Le soutien des intervenants :* D'ordinaire, les projets sont le fait de plusieurs parties prenantes. Il importe que celles-ci accordent leur soutien pour toute la durée du projet de façon à ce que l'équipe atteigne ses objectifs.

## 5. L'objectif de projet

Quand on doit choisir la manière d'aborder un projet, il existe 3 notions fondamentales qu'il faut connaître et évaluer : la qualité, le coût et le délai.

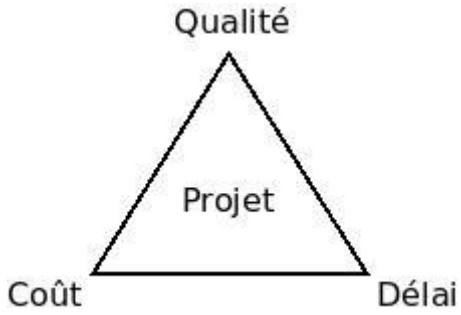



**Figure 2.1** Le triangle Qualité, Coût, Délai

- **Qualité** : Il s'agit du soin qui est apporté à la réalisation fonctionnelle et technique du projet. Un projet de médiocre qualité remplira les besoins immédiats du client, en s'autorisant un certain nombre de raccourcis. Un projet de bonne qualité aura été spécifié pour couvrir certains besoins futurs identifiables, et offrira une ergonomie adaptée, des performances homogènes, une évolutivité étudiée, une documentation complète.
- **Coût** : Un client est prêt à dépenser une certaine somme pour un projet donné. La valeur du projet peut éventuellement s'adapter à un certain nombre de critères, mais il y a forcément un seuil au-delà duquel il est impossible de le rentabiliser. La notion de coût englobe aussi bien les frais d'étude (en fonction du temps passé aux spécifications fonctionnelles et techniques) et de réalisation (suivant le nombre de développeurs nécessaires, le matériel mis à leur disposition, la présence d'une équipe de test et de validation, …), que les frais d'exploitation (matériel nécessaire pour faire tourner le projet en production, salaire de l'opérateur de maintenance, …).

- **Délai** : Savoir combien de temps doit durer la réalisation d'un projet n'est pas aisé, même si cela fait partie du travail d'un ingénieur. Certains projets ne sont pas urgents, ni même importants, mais ils comportent forcément une deadline à partir de laquelle ils deviennent caducs.

## 6. Cycle de vie d'un projet

Un projet se démarque par son cycle de vie, qui est généralement présenté comme étant constitué de phases. Le nombre de phases ainsi que leur appellation peuvent varier d'une application à une autre, d'un domaine d'application à un autre et d'un auteur à un autre. L'ingénieur responsable d'un projet devra parfois définir les phases du projet dont il a la responsabilité en tenant compte des paramètres propres au projet ou à la culture d'entreprise. Ces différences ne limitent en aucune façon la validité ni la pertinence du modèle ci-dessous en quatre phases qu'il est proposé à l'ingénieur de suivre. [24]

1. *Phase d'identification* **:** la demande est clarifiée, les objectifs précisés et le projet globalement identifié en ce qui a trait au produit ou au service à livrer, aux contraintes à respecter et à la stratégie de réalisation.

2. *Phase de définition* **:** le contenu du projet est défini de façon plus précise, une planification détaillée est établie pour sa durée; les échéances, les ressources et les dépenses, ainsi que les politiques et les procédures de gestion sont circonscrites.

3. *Phase de réalisation* **:** le produit ou le service est effectivement réalisé suivant le plan prévu et en conformité avec les exigences du demandeur.

4. *Phase de clôture* **:** le produit ou le service est remis au demandeur, le projet est évalué et sa clôture administrative effectuée.

On distingue différents cycles de vie en fonction d'un projet, non seulement selon les auteurs, mais aussi selon les domaines, comme la construction, la recherche universitaire et le génie logiciel (méthode Scrum).

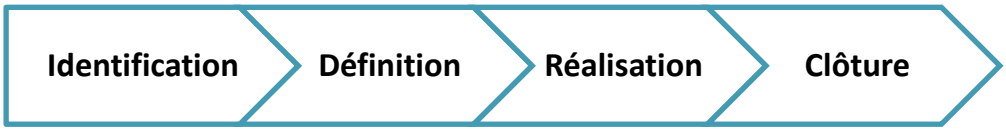



**Figure 2.2** Les phases d'un projet

## 7. L'ordonnancement dans la gestion de projet

L'ordonnancement se situe exactement dans la phase planification. Il réalise le suivi opérationnel du projet : gestion de ressources, suivi de l'avancement, lancement des activités. Techniquement, ordonnancer un projet consiste à programmer dans le temps l'exécution des tâches, tout en respectant les contraintes et de manière à optimiser les critères de performance retenus. C'est plus particulièrement à ce stade qu'interviennent les techniques d'ordonnancement de projet présentées dans ce qui suit.

### 7.1 Le problème central de l'ordonnancement

Pour le problème central de l'ordonnancement, il s'agit d'ordonnancer en une durée minimale un ensemble I = {1, 2, …, n} de tâches soumises à des contraintes temporelles de type inégalité de potentiel.

Donc, dans le problème central de l'ordonnancement, on ne tient compte que des contraintes de potentiels qui sont d'antériorité et de durée. Les contraintes disjonctives et cumulatives sont ignorées. Les ressources sont supposées à quantité illimitée.

- Modélisation du problème central :

Il existe actuellement trois méthodes de modélisation du problème central : *le diagramme de Gantt*, *la méthode des potentiels* et *la méthode PERT*.

Ces deux dernières utilisent comme outil de modélisation la théorie des graphes ; et plus particulièrement le réseau.

Nous définissons alors *un graphe conjonctif* G = (X, U), qui est value ayant une racine α et un puits ω tel qu'il existe un chemin de valeur positive entre α et tous les autres sommets, et un chemin de valeur positive entre tout sommet différent du puits ω et ω.

Une condition nécessaire et suffisante pour appliquer l'une des méthodes de modélisation des problèmes d'ordonnancement sur un graphe conjonctif est que ce graphe n'ait pas de *circuit* et les valeurs sur les arcs (les *ti*) sont de valeurs positives ou nulles. De part la nature du problème, une tâche ne peut se succéder à elle-même, donc le graphe ne contient pas de boucles.

### 7.1.1 Le diagramme de Gantt [08]

Au début du siècle dernier, le gouvernement américain déclara la guerre à l'Allemagne, Gantt développa alors, une représentation graphique de déroulement des projets, qui prit plus tard le nom de diagramme de Gantt du nom de son inventeur.

Donc, pour faire le planning de l'exécution du projet et pour en contrôler le déroulement on utilise le diagramme à barres (Bar Charte) ou diagramme de Gantt.

Chaque tâche est symbolisée par un rectangle dans lequel sont inscrits le code et la durée de la tâche. Commençant par représenter les tâches avec antériorité α, α une tâche est portée dans le diagramme lorsque toutes ses antériorités y sont déjà portées. Ceci est toujours possible, car une tâche ne se succède pas à elle même, les rectangles sont ainsi toujours placés à droite (c'est le même principe que celui des niveaux d'un graphe sans circuit). Cette représentation indique bien le déroulement des travaux mais moins bien les antériorités.

On introduit cette technique à l'aide d'un exemple du problème de réalisation d'une construction tiré de :

| **La tâche** | **Désignations** | **durée** | **Antériorités Immédiates** |
|---|---|---|---|
| A | Etablissement des plans | 5 | - |
| B | Terrassement | 4 | - |
| C | Creuser les fonds de piliers | 6 | - |
| D | Faire les fonds de piliers | 3 | A, B, C |
| E | Placement arrivée et évacuation assainissement | 7 | A,B, C |
| F | Dalle | 5 | A,B, C |
| G | Cloisonnage | 8 | D,E |
| H | Carrelage | 6 | D,E |
| I | Plomberie | 5 | F |
| J | Installation chauffage | 3 | F |
| K | Electricité | 5 | G, H, I, J |
| L | Pose faïence | 2 | G, H, I, J |
| M | Boiserie | 1 | K, L |
| N | Peinture | 4 | K, L |

**Tableau 2.2** Table initiale d'ordonnancement

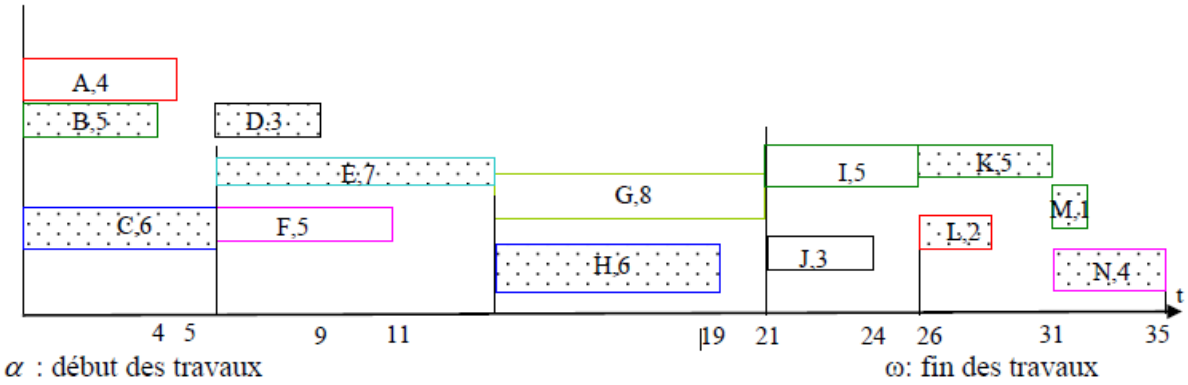



**Figure 2.3 Le** diagramme de Gantt de la table T

### 7.1.2 Le graphe PERT

PERT (Program Evaluation and Review Technique – technique d'ordonnancement et de contrôle des programmes) et le CPM ont été indépendamment développés vers la fin des années 50. Depuis, ils ont été parmi les plus largement appliqués.

Les versions originales de PERT et du CPM ont eu quelques différences importantes. Cependant, elles ont également eu beaucoup de commun, et les deux techniques ont été graduellement fusionnées au cours des années. En conséquence, les praticiens emploient maintenant généralement les deux noms l'un pour l'autre, ou les combinent dans l'acronyme simple PERT/CPM.

La méthode PERT a été mise au point par la marine américaine qui devait conduire à la réalisation des missiles à ogive nucléaire Polarisa. Cette technique a permis de coordonner les travaux de près de 6000 constructeurs, 250 fournisseurs et 9000 sous-traitants dans un délai de réalisation de 7 ans.

L'utilisation de la méthode PERT a permis de ramener la durée globale du projet de 7 à 4 ans.

Bien avant la méthode PERT, les problèmes d'ordonnancement ont été abordés par le diagramme de Gantt. Comme on l'a constaté précédemment s'il constituait un moyen simple et commode pour transmettre l'information relative à un ordonnancement, c'est un moyen médiocre pour concevoir ce même ordonnancement malgré qu'il puisse exprimer d'une manière stricte l'exigence de postériorité.

Les Caractéristiques de PERT sont les suivantes :

- Les tâches sont représentées par des flèches.
- Le réseau visualise des dépendances entre tâches.
- Calcul les dates « plus tôt » et « plus tard » pour lancer chaque tâche.
- Le Chemin critique : chemin dont la durée la plus tôt = durée plus tard.
- Les marges : La marge totale (Mt). La marge libre (Ml).

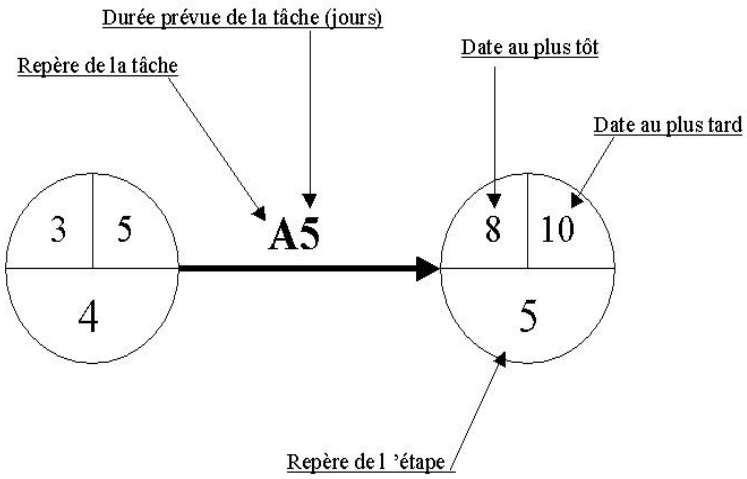



**Figure 2.4** Représentation de tâche A dans le réseau PERT.

**7.1.3** Le graphe des potentiels

La méthode des potentiels a été développée vers la fin des années 50 parallèlement à la méthode PERT. Elle est appelée également la méthode MPM (méthode des potentiels Metra) ou encore la méthode des potentiels –tâches.

Les tâches sont symbolisées par des sommets auxquels on donne le même code, les 2 sommets u et v sont reliés par un arc de u vers v si et seulement si la tâche u précède la tâche v.

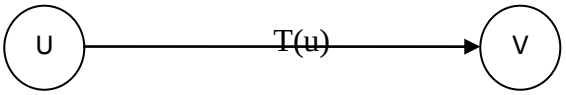



**Figure 2.5** La tâche u, de durée t (u), précède la tâche v

Considérons l'exemple de réalisation précédent :

| Codes | Durées | Antériorités |
|---|---|---|
| α | 0 | - |
| A | 4 | α |
| B | 5 | α |
| C | 6 | α |
| D | 3 | A, B, C |
| E | 7 | A,B, C |
| F | 5 | A,B, C |
| G | 8 | D,E |
| H | 6 | D,E |
| I | 5 | F |
| J | 3 | F |
| K | 5 | G, H, I, J |
| L | 2 | G, H, I, J |
| M | 1 | K, L |
| N | 4 | K, L |
| Ω | 0 | M, N |

**Tableau 2.3** Table de priorité

On porte en suite sur chaque arc incident extérieurement à un sommet u la durée de la tâche u correspondante de sorte que les valeurs sur les arcs de même extrémité initiale soient égales. Nous verrons cependant plus loin que ces valeurs vont différer en rajoutant des contraintes sur les tâches autres que les antériorités.

Un chemin , A1, A2, ….., Ak, A du graphe de $\alpha$ à une tâche A correspond alors à une suite de tâche A1, A2 , ….., Ak, précède Ai+1, i=1,…, k-1, et Ak précède A. La durée d'un tel chemin est la somme des durées des tâches qui le composent. Le dessin du graphe est très simple puisqu'il suffit de disposer les sommets aléatoirement et de les relier par les arcs. Le graphe étant sans circuit et sans boucle (une tâche ne se succède pas à elle-même).

La réorganisation des sommets en niveaux montre clairement les antériorités. Elle se fait comme suit :

- **niveau I:** la tâche source $\alpha$ sans antécédent ;
- **niveau II :** tâches n'ayant que le prédécesseur $\alpha$ ;
- **niveau III :** tâches n'ayant que les prédécesseurs du niveau II ;
- **niveau IV :** tâches n'ayant que les prédécesseurs du niveau III ;
- etc.
- le dernier niveau est ω qui n'a aucun successeur.

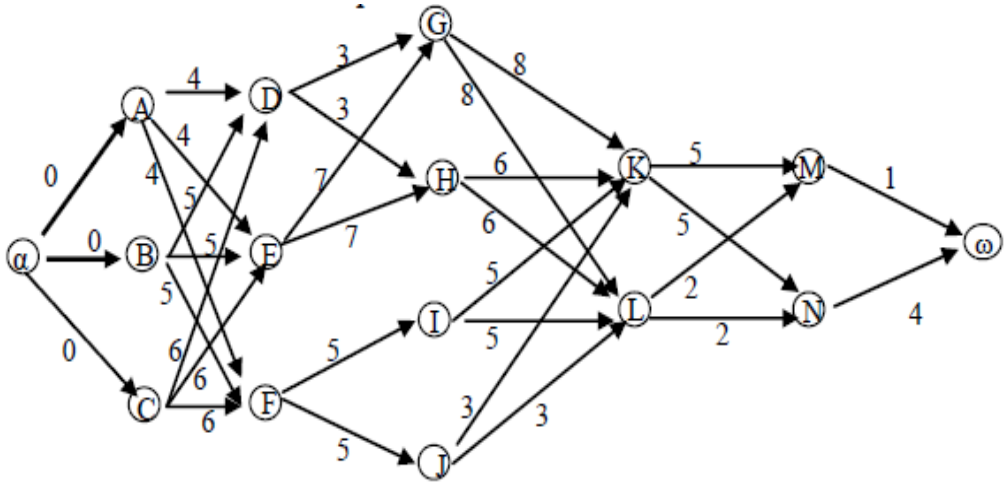



**Figure 2.6 L**e graphe des potentiels avec les sommets réorganisés en niveaux

Enfin, notons que le mot potentiel associé au graphe est issu de l'analogie avec la différence de potentiels entre deux nœuds d'un réseau électrique.

## 8. Notion de tâche fictive

Si le graphe de la méthode des potentiels et celui de la méthode PERT sont très proches, ce n'est pas toujours le cas. La construction du graphe PERT pose des problèmes qui amènent à ajouter des arcs fictifs (virtuels ou artificiels) qui ne correspondent à aucune tâche.

Supposons maintenant que le problème d'ordonnancement comporte 4 tâches a, b, c et d telles que a précède c sans précéder d et b précède c et d.

Dans le graphe PERT, la représentation n'est pas possible sans une légère modification. En effet, a et b précèdent c, les extrémités terminales de a et b devraient coïncider avec l'extrémité initiale de c. Or, l'extrémité initiale de d devrait coïncider aussi avec l'extrémité terminale de b donc de a mais a ne précède pas d. Ce problème est résolu par l'introduction *d'une tâche fictive f de durée* 0 dont le principe est de séparer les extrémités terminales de a et b en éclatant le sommet s, tout en gardant les contraintes de succession.

L'introduction de la tâche fictive modifie le tableau des antériorités lequel donne la correspondance dans le graphe des potentiels. [04]

| Code (s) | Antériorité |
|---|---|
| c | a, b |
| d | b |

**Tableau 2.4** (a) Un sous-tableau des antériorités de c et d

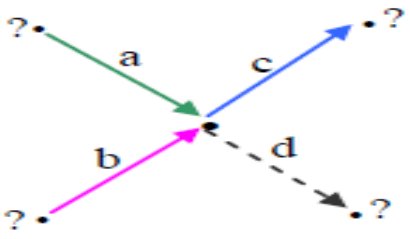



**Figure 2.7** (a) Problème de Représentation

| Code(s) | Antériorité |
|---|---|
| C | a, f |
| D | b |
| f | b |

**Tableau 2.5 (**b) Nouveau sous tableau des antériorités c, d et f

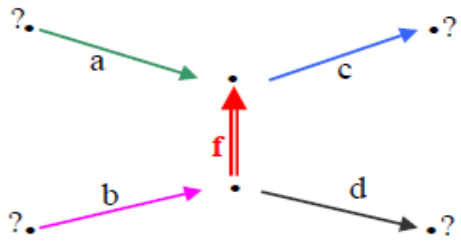



**Figure 2.8** (b Introduction de la tâche fictive f) et représentation dans le graphe PERT

L'introduction des tâches fictives permet de solutionner certaines situations et de lever des ambiguïtés. Elles ne mettent en jeu aucun moyen matériel ou financier. Considérons l'exemple de construction simplifié suivant :

| Codes | Durées | Antériorité |
|---|---|---|
| α | 0 | - |

| | | |
|---|---|---|
| A | 2 | α |
| B | 2 | α |
| C | 2 | H |
| D | 3 | α |
| E | 4 | B, G |
| F | 2 | C, I |
| G | 3 | A, D |
| H | 4 | B, D |
| I | 5 | H |
| J | 3 | C |
| ω | 0 | E, J, F |

| Niveau | Sommet |
|---|---|
| Niveau 1 | α |
| Niveau 2 | A, B, D |
| Niveau3 | G, H |
| Niveau4 | C, E, I |
| Niveau5 | F, J |
| Niveau6 | ω |

**Tableau 2.6** La table d'ordonnancement T et l'organisation des sommets en niveaux

Le dessin du graphe PERT se fait exactement comme l'exemple précédent :

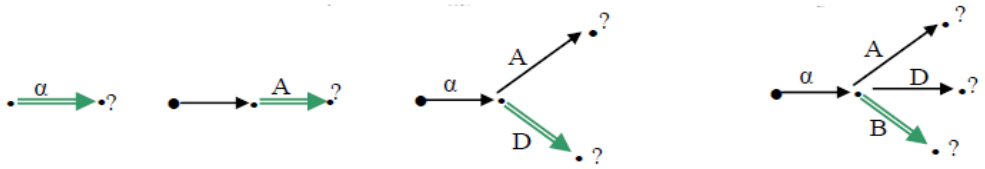



**Figure 2.9** étape (a)

Pour dessiner la tâche G il faut que les arcs des tâches A et D coïncident dans leurs extrémités terminales, or la tâche D est antérieure à une autre tâche (H), d'où la création d'une tâche fictive f1. Le même scénario se répète avec la tâche H : il faut que les extrémités de D et B coïncident. Or D est antérieure à G ; donc il faut créer une autre tâche fictive f2.

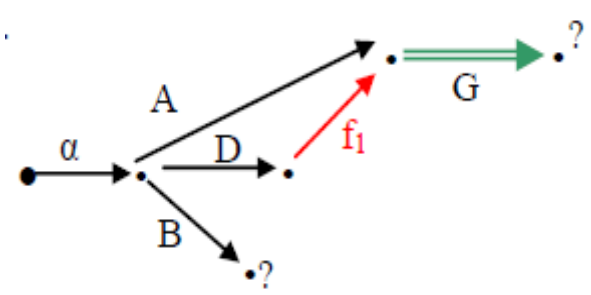



**Figure 2.10** étape(b)

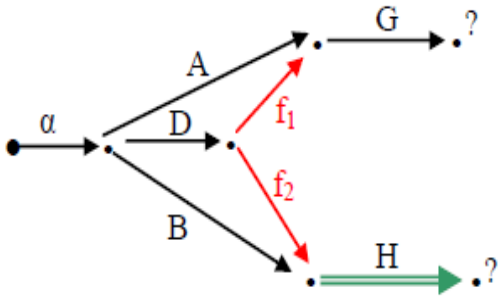



**Figure 2.11** étape(c)

Et nous continuons le même processus jusqu'à la dernière ligne de la table :

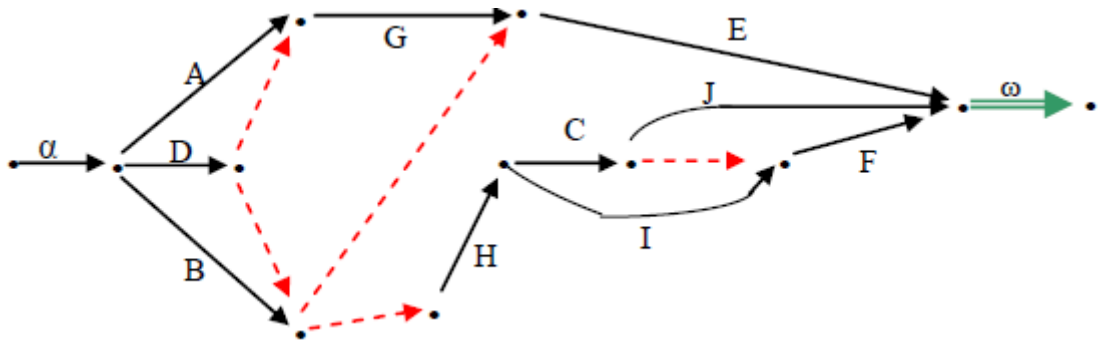



**Figure 2.12** le graphe PERT de table d'ordonnancement T

## 9. Conclusion

Le réseau PERT, le graphe de potentiels et le diagramme de Gantt sont les plus utilisés pour modéliser un problème central d'ordonnancement, mais il y a une infinité de graphes PERT à cause des tâches fictives qu'on peut introduire comme on veut et là ou on veut puisqu'elles ne nécessitent aucun moyen matériel ou financier qui influe sur la durée globale du projet. Le problème est de chercher à dessiner un graphe PERT (difficile à dessiner) à partir d'un graphe des potentiels (facile à dessiner) avec le minimum de tâches fictives.

# CHAPITRE 03

# ALGORITHME DE COHEN ET SADEH POUR LA CONSTRUCTION DU GRAPHE AOA

## 1. Introduction

La construction du graphe AON (Activities on Nodes) est relativement facile, les réseaux AOA peuvent contenir des arcs fictifs et leur construction est beaucoup plus problématique. Toutefois, en raison de leur importance, les réseaux AOA ont été l'objet de nombreuses tentatives de recherche pour générer des réseaux AOA avec un nombre minimum d'arcs fictifs. Il est cependant à noter qu'un seul réseau AON peut être représenté par plusieurs réseaux AOA différents avec le même indice de complexité minimale.

La signification d'un nœud AOA dans tous les documents ci-dessus est un événement dans le temps, qui ne correspond pas nécessairement à quelque chose exprimé par l'AON du même projet. Ainsi, le réseau résultant est alambiqué et il est difficile (et parfois impossible) de récupérer les contraintes initiales de priorité immédiate.

L'approche dans ce chapitre est différente. Nous proposons un réseau AOA dans lequel chaque nœud correspond à une contrainte de préséance.

L'algorithme proposé permet à la fois une traduction systématique du réseau AON vers le réseau AOA et un processus de construction systématique des réseaux AOA à partir des tableaux immédiatement précédents correspondants

## 2. Représentation du réseau AOA vs AON [12]

La première étape dans la construction d'un réseau AOA est d'énumérer les prédécesseurs immédiats de chaque activité et de séquencer les activités dans une table en fonction des contraintes de priorité. Le tableau de priorité résultant correspond directement à un schéma de réseau AON. Dans un diagramme AON, les activités sont représentées sur les nœuds et les contraintes de priorité par les arcs reliant les nœuds. Par conséquent, dans le graphe AON, le nombre de nœuds est égal au nombre d'activités, et le nombre d'arcs est le nombre de prédécesseurs immédiats. La représentation AON est naturelle et ne contient pas d'éléments artificiels sur le réseau.

En revanche, les réseaux AOA sont caractérisés par des arcs factices et des relations de préséance implicites. En conséquence, chaque projet peut avoir plusieurs représentations AOA. Les différentes représentations ont des effets réels et indésirables sur les mesures de la flexibilité de certaines activités.

D'autre part, la structure du réseau AOA est beaucoup plus appropriée pour certaines techniques analytiques et formulations d'optimisation. Par exemple, les activités AOA correspondent à des entrées dans une matrice d'origine à destination qui à son tour regroupe certaines formulations de programmation linéaire. La plupart des techniques d'optimisation de réseau sont basées soit sur des formulations d'arc de nœud, soit sur des formulations de matrice d'origine-destination (c'est-à-dire une représentation AOA et pas une représentation AON). De plus, la plupart des formulations de programmation linéaire pour trouver le chemin critique ou «planter» des activités critiques sont basées sur AOA. Un autre type de formulations utilise des soustractions de variables de nœud AOA pour exprimer la valeur d'arc (activité).

## 3. La Logique de l'algorithme [12]

Cette section décrit brièvement la logique de l'algorithme qui génère un réseau AOA à partir d'un réseau AON correspondant ou d'une table de priorité. On peut trouver aussi une discussion plus élaborée. Cependant, la logique et ses détails sont mieux expliqués dans la section 6.

- La première étape : de l'algorithme présenté ci-dessous est presque triviale: construire une table de préséance simple, qui répertorie les prédécesseurs immédiats (activités) pour chaque activité et la transformer en un réseau «Activités sur nœuds». Pour $n$ tâches, la complexité de la première étape est O ($n^2$).

- La deuxième étape : identifie les nœuds AOA finales. Chaque nœud AOA correspond à une contrainte de priorité unique. Chaque nœud dans AOA marque l'événement de départ des arcs émanant de celui-ci. Ainsi, si deux ou plusieurs activités ont la même contrainte de priorité, leurs arcs peuvent émaner du même nœud (ce qui correspond à cette contrainte). En combinant le point de départ de tous les arcs qui ont la même contrainte de priorité, en un seul nœud. Cela aide à minimiser le nombre de nœuds dans le réseau AOA. Dans la deuxième étape, l'algorithme génère une liste de contraintes de priorité uniques, dont chacune deviendra un nœud dans le diagramme

AOA final (y compris le nœud de départ pour les activités sans prédécesseurs). Enfin, si plus d'une activité n'a pas de successeur, un nœud fictif doit être ajouté symbolisant la fin du projet.

- La troisième étape : consiste à identifier les mannequins nécessaires. Un arc fictif est nécessaire chaque fois qu'une activité fait partie dans plus d'une contrainte de priorité unique (à partir de l'étape 2). La raison d'avoir besoin de l'arc factice est d'avoir plus d'un nœud final (chaque nœud représente les contraintes de priorité différentes de ses successeurs). Ainsi, la troisième étape de l'algorithme est d'identifier tous ces mannequins nécessaires.

Après la troisième étape, nous avons tous les nœuds AOA (à partir de l'étape 2) et tous les arcs AOA (un arc pour chaque activité et des arcs factices de l'étape 3). Il ne reste plus qu'à assigner chaque paire de nœuds de début et de fin à leurs arcs correspondants. Le reste des étapes algorithmiques atteindre cette fin. De plus, donner aux activités de nouveaux noms selon un schéma «de-là» facilite la formulation du problème d'optimisation et fait également partie de l'algorithme. L'algorithme contient un total de sept étapes. Les résultats de chaque étape sont résumés dans un tableau comme il est indiqué dans la section suivante. La septième étape se termine par une table appropriée à former un réseau "Activités sur Arcs".

## 4. Les sept étapes et leurs complexités sont les suivantes [13]

1. Construire la table des prédécesseurs immédiats, O ($n^2$).
2. Trouver une liste de relations de préséance uniques et les numéroter comme nœuds AOA, O ($n$).
3. Trouver les activités fictives nécessaires, O ($n$).
4. Ajouter des activités fictives au tableau, O($n$).
5. Associer chaque nœud AOA à ses arcs entrants, O($n^2$).
6. Associer chaque nœud à ses arcs sortants, O($n$).
7. Marquez les arcs (activités) avec des index par leurs nœuds de début et de fin, O($n$). A partir de ce qui précède,

La complexité globale de l'algorithme est **O($n^2$).**

### 4.1 Le cas d'étude « AON »

L'exemple détaillé ci-dessous aidera à illustrer l'algorithme. L'exemple est basé sur un projet de construction de béton coulé. Les données sont fournies dans le tableau 1, et une illustration des données est donnée sur la figure 1 en utilisant la représentation AON.

| **Les activités** | **Sigle** | **Prédécesseurs immédiates** |
|---|---|---|
| A. Obtenir des tiges d'acier et de poutres en acier | Obtenir des tiges | - |
| B. Obtenir des tiges et des poutres en bois | Obtenir des tiges en bois | - |
| C. Obtenir plage de sable | Obtenir plage | - |
| D. Obtenir béton poudre | Obtenir béton | - |
| E. Obtenir réservoir pour mélanger le béton | Obtenir réservoir | - |
| F. Préparation du squelette de coulée comme une grille de barres d'acier et fixés par des fils de soudure | Préparation du squelette | A |
| G. Plier les poutres en acier pour support de châssis | Plier les poutres en acier | A |
| H. Construct a concrete casting frame from wooden beams | Construct a concrete | B |
| I. Connecte (avec un tuyau), un tuyau d'eau dans le réservoir de mélange | Connecte l'eau | E |
| J. Mix béton poudre, sable et eau dans le réservoir | Mix béton | C,D,I |
| K. Couvrir le cadre de moulage avec des panneaux de contreplaqué | Couvrir le cadre | H |
| L. Fixer le squelette interne des tiges d'acier sur le châssis de moulage externe | Fixer le squelette interne | F,H |
| M. Soutenez le cadre avec des poutres en acier | Soutenez le cadre | G,H |
| N. Soutenez les panneaux de contreplaqué avec des poutres en acier | Soutenez les panneaux | G,K |
| O. Jetez le mélange de béton dans le cadre | Jetez le mélange | J,L,M,N |

**Tableau 3.7** Un exemple d'activités et de leurs contraintes de précédence

## 5. Les détails des étapes de l'algorithme [13]

Les étapes de l'algorithme sont détaillées ci-dessous et résumées comme suit : les trois premières étapes sont résumées dans le tableau 2, les étapes quatre et cinq sont résumées dans le tableau 3 et les dernières étapes sont résumées dans le tableau 4. Il est recommandé de suivre l'algorithme à la fin de chaque étape consécutive, en comparant la colonne de l'étape et la colonne précédente de l'étape

**Etape 1** Construire les prédécesseurs immédiats de chaque colonne :

Préparer une colonne avec les prédécesseurs immédiats de chaque activité. Par exemple, voir la colonne 1 dans le tableau 2.

**Étape 2** Identifier les activités avec des contraintes identiques (ces activités vont se transformer en arcs émanant du même nœud :

Aller sur la colonne 2 et trouver et répéter des contraintes de priorité. Pour chaque contrainte répétitive insérer la marque "-" dans la colonne 2, bloquant l'entrée de la contrainte de répétition. Nous pouvons voir dans la colonne 2 que les contraintes récurrentes sont la contrainte nulle (précédentes activités A à E) et de l'activité A (précédentes activités F et G). Par conséquent, les arcs des activités A à E seront crées depuis le nœud 0 (dans le filet AOA) et les arcs des activités F et G seront crées à partir du nœud de l'activité A. final Depuis la colonne 2 doit avoir une liste unique sans répétitions, les occurrences de répétition à des lignes B à E sont bloquées. La même chose est vraie pour la ligne d'activité G dans la colonne 2.

Ouvrez une troisième colonne (colonne 3) et le nombre des contraintes de précédence uniques séquentiellement. Les répétitions obtiennent le même nombre que la première occurrence. Tableau 2 énumère dix (0 à 9) des contraintes de précédence uniques correspondant aux nœuds AOA. Ainsi, nous savons déjà que, avec le nœud final, l'AOA dans l'exemple ne peut pas avoir moins de 10 + 1 = 11 nœuds.

**Étape 3** Identifier nécessairement les arcs fictifs:

Colonne de balayage 2 du tableau 2 (étape 2) et les activités de la liste qui apparaissent plus d'une fois dans cette colonne. Par exemple, la liste de l'activité G car il apparaît comme

un prédécesseur des activités M et N (à noter que les activités M et N ont des contraintes de priorité). En outre, l'activité de la liste H précède les activités K, L et M. Puisque toutes les entrées non-bloquées sont contraintes uniques, chacun d'eux représente un nœud d'AOA. Les activités répétitives (à savoir, G, H) sont des précédents à deux ou plusieurs nœuds d'AOA. Etant donné que l'arc ne peut pointer vers l'un de ces nœuds, il existe un besoin pour des arcs fictifs pointant vers les autres nœuds.

Si l'une des répétitions trouvées dans 3.1 ci-dessus est une activité unique, aucun factif est nécessaire pour cette répétition / contrainte. Le nœud de l'activité prédécesseur unique fin devient le point du successeur de départ. Par exemple, H est le seul prédécesseur immédiat de l'activité K. Ainsi, la flèche de l'activité K peut provenir au niveau du nœud de l'activité H de fin, et il n'y a pas besoin d'un factif dans ce cas. Cependant, comme H est une partie de deux autres relations de priorité, les factifs sont nécessaires.

Répétitions non conformes à 3.2 correspondent à des factifs. Renommez ces factifs avec des lettres. Par exemple, l'activité H est un prédécesseur de l'activité L et M. Cependant, H n'est pas le seul prédécesseur de L ou de M. Ainsi, les deux répétitions (L, M) ne sont pas conformes à 3,2 et, par conséquent, il est nécessaire d' utiliser deux arcs factices. Dans la colonne 3, les deux apparitions de H sont numérotés H1 et H2 de les désigner comme des factifs. Un autre exemple est l'activité G qui précède les activités M et N. Les deux répétitions ne sont pas conformes à 3.2 et sont donc renuméroté G1 et G2 de les désigner comme des mannequins. Les 3 premières étapes de l'algorithme en utilisant l'exemple sont résumées dans le tableau 2. Les activités sont marquées par des lettres et les nœuds sont marqués par des numéros.

| | Etape1 | Etape2 | | Etape3 |
|---|---|---|---|---|
| Activité | Prédécesseurs immédiates | Identifier les même Prédécesseurs | Numération | Identifier les tâches fictives |
| A | - | - | 0 | - |
| B | - | - | 0 | - |
| C | - | - | 0 | - |
| D | - | - | 0 | - |
| E | - | - | 0 | - |
| F | A | A | 1 | A |
| G | A | // | 1 | // |
| H | B | B | 2 | B |
| I | E | E | 3 | E |
| J | C D I | C D I | 4 | C D I |
| K | H | H | 5 | H |
| L | F H | F H | 6 | F **H1** |
| M | G H | G H | 7 | **G1 H2** |
| N | G K | G K | 8 | **G2** K |
| O | J L M N | J L M N | 9 | J L M N |

**Tableau 3.8** Les étapes 1, 2, 3

**Étape 4** Ajouter les lignes et les informations pour les arcs fictifs :

Pour chaque activité fictive (identifiée à l'étape 3), ajoutez une ligne à la fin du tableau 2. Remplissez cette ligne avec le nom de l'activité factice et écrivez l'activité réelle que le prédécesseur (par exemple, ajoutez une nouvelle ligne pour l'activité G1, et de faire G son prédécesseur). Colonne 1 du tableau 3 illustre le résultat de l'étape 4.

**Étape 5** Activités associées avec leurs nœuds finaux :

A ce stade, nous acceptons la colonne de travail est la nouvelle colonne de prédécesseur immédiat. Les étapes 3 et 4 ont été consacrées à la résolution du problème suivant: Chaque fois une activité fait partie de deux ou plusieurs contraintes, un arc ou une flèche (qui pointe vers un seul nœud) ne suffit pas pour sa représentation. Ainsi, les arcs factifs supplémentaires sont nécessaires pour modéliser cette activité.

Ainsi, après les étapes 3 et 4, chaque activité a au plus un non-fictive nœud suivant (s'il y avait d'abord eu d'autres nœuds, ils ont été assignés arcs fictifs). De l'étape 2, nous savons que le nœud AOA à partir de chaque activité (tableau 2, colonne 3). Étant donné que chaque activité a ses propres prédécesseurs immédiats, chaque nœud d'AOA commençant aussi servir le nœud de ses prédécesseurs immédiats de fin. Donc, pour chaque activité, nous ajoutons deux nouvelles colonnes: une colonne successeurs (colonne 3, tableau 3), et un successeur commence colonne de nœud (colonne 4, tableau 3). Le nœud successeurs de départ est aussi le nœud de l'arc de l'activité de fin.

Trouver les successeurs (tableau 3, colonne 3)

Pour chaque activité, analyser la colonne 1 dans le tableau 3 et trouver où l'activité apparaît comme un prédécesseur immédiat. Cette activité est illustrée par des flèches dans le tableau 3.

Si une activité manque encore un successeur, il est soit parce que son nœud final est suivi par des fictives seulement, ou il est le nœud final du projet. Pour ces entrées de nœud final manquant ajouter de nouveaux numéros de nœuds. Par exemple, étant donné que l'activité G est suivie par des fictives seulement, il a besoin d'un nœud artificiel. Par conséquent, le nœud 10 est ajouté (voir tableau 3, colonne 4). En outre, le nœud 11 est ajouté pour le nœud de fin du projet (voir tableau 3, colonne 4).

Trouver le nœud de départ successeur (qui est le nœud activité de fin - tableau 3, colonne 4)

Le nœud de départ de toute activité non-fictive (activités du successeur inclus) est déjà dans la colonne 2 du tableau 3 (prise à partir du tableau 2, colonne 3). Une fois un successeur a été trouvé, son nœud de départ doit être le nœud de l'arc prédécesseur final. Regarder dans la colonne 2 du tableau 3 à la ligne du successeur donne le nœud de départ du successeur. Ce nœud de départ est le nœud de l'activité prédécesseur, qui est l'activité de la ligne courante fin. Ainsi, nous allons successivement sur toutes les lignes et trouver leurs nœuds d'extrémité correspondants de cette façon.

| | Etape4 | Etape5 | | |
|---|---|---|---|---|
| Activité | Pred Immédiates | Nœud de départ AOA | Successeurs | Nœud de fin AOA |
| A | - | 0 | F G | 1 |
| B | - | 0 | H | 2 |
| C | - | 0 | J | 4 |
| D | - | 0 | J | 4 |
| E | - | 0 | I | 3 |
| F | A | 1 | L | 6 |
| G | // | 1 | G1 G2 | 10 |
| H | B | 2 | K H1 H2 | 5 |
| I | E | 3 | J | 4 |
| J | C D I | 4 | O | 9 |
| K | H | 5 | H | 2 |
| L | F **H1** | 6 | O | 9 |
| M | **G1 H2** | 7 | O | 9 |
| N | **G2** K | 8 | O | 9 |
| O | J L M N | 9 | End | 11 |
| **G1** | G | | M | 7 |
| **G2** | G | | N | 8 |
| **H1** | H | | L | 6 |
| **H2** | H | | N | 7 |

**Tableau 3.9** Les étapes 4,5

**Étape 6** Arcs fictifs associés avec leurs nœuds de démarrage :

Chaque arc fictif représente la fin de son activité non-fictive associée. Ainsi, l'arc fictif devrait suivre le nœud de l'arc réel de son activité de fin. Tous les nœuds d'extrémité sont identifiés à l'étape 5. Par conséquent, l'étape 5 est nécessaire avant l'étape 6 pour permettre la désignation d'un nœud d'origine à chaque arc factice. Pour récapituler, le nœud de départ de chaque fictif est le nœud de son activité réelle correspondante de fin. Par exemple, l'activité fictive G1 commence où l'activité G se termine. Puisque l'activité G se termine au nœud 10, l'arc de l'activité G1 commence au nœud 10. Le bas de la colonne 2 du tableau 4 montre les résultats de l'étape 6.

**Étape 7**: Vérifier la nécessité d'ajouter des nœuds et des arcs fictifs pour compenser l'effet d'un nœud de départ d'un seul projet, et un seul nœud de fin du projet. Mettre à jour la table avec les nouveaux noms d'activité :

Il est une convention d'avoir un nœud de départ pour le début d'un projet. Par conséquent, quand un réseau de projet commence par plusieurs arcs parallèles à partir du nœud de départ, certains de ces arcs peuvent se terminer au même nœud d'extrémité, créant une situation où les différents arcs ont les mêmes nœuds de début et de fin. Pour notations classiques tels que Aij il n'y a aucun moyen de différencier entre ces arcs, et en plus ce n'est pas un format acceptable pour certaines formulations de LP (…….). Par conséquent, nœud fictif et l'arc sont ajoutés pour de tels cas (jusqu'à ce qu'il n'y a pas deux arcs avec le même début et nœuds d'extrémité). Par exemple, les deux activités C et D commencent au nœud 0 et se terminent au nœud 4. Par conséquent, un nœud fictif 12 est ajouté pour l'activité 4 et un nouvel arc factif. Maintenant activité 4 commence au nœud 0, mais se termine au nœud 12.

Ensuite, nous connectons un arc fictif supplémentaire à partir du nœud 12 au nœud 4 (en bas du tableau 4).

La même situation peut survenir à la fin du projet (car il est de coutume d'avoir un nœud final unique par projet). Encore une fois, la solution consiste à ajouter des nœuds et des arcs factifs si nécessaire. Ces situations sont faciles à détecter par balayage des colonnes des nœuds de début et de fin. Si la même paire apparaît plus d'une fois, le nœud fictif et l'arc sont nécessaires pour chaque répétition. A la fin de ce processus chaque arc doit avoir sa combinaison unique de nœud de début / fin.

| | Etape4 | Etape6 | Etape7 | | |
|---|---|---|---|---|---|
| Activité | Pred Immédiates | Nœud de départ AOA | Nœud de fin AOA | Nom des activités AOA | AOA Prédécesseurs immédiates |
| A | - | 0 | 1 | A(0.1) | - |
| B | - | 0 | 2 | A(0.2) | - |
| C | - | 0 | 4 | A(0.4) | - |
| D | - | 0 | 4 | A(0.4) | - |
| E | - | 0 | 3 | A(0.3) | - |
| F | A | 1 | 6 | A(1.6) | A(0.1) |
| G | // | 1 | 10 | A(1.10) | A(0.1) |
| H | B | 2 | 5 | A(2.5) | A(0.2) |
| I | E | 3 | 4 | A(3.4) | A(0.3) |
| J | C D I | 4 | 9 | A(4.9) | A(0.4) A(0.4) A(3.4) |
| K | H | 5 | 2 | A(5.2) | A(2.5) |
| L | F **H1** | 6 | 9 | A(6.9) | A(1.6) A(5.6) |
| M | **G1 H2** | 7 | 9 | A(7.9) | A(10.7) A(5.7) |
| N | **G2** K | 8 | 9 | A(8.9) | A(10.8) A(5.2) |
| O | J L M N | 9 | 11 | A(9.11) | A(4.9) A(6.9) A(6.9) A(8.9) |
| **G1** | G | 10 | 7 | A(10.7) | A(1.10) |
| **G2** | G | 10 | 8 | A(10.8) | A(1.10) |
| **H1** | H | 5 | 6 | A(5.6) | A(2.5) |
| **H2** | H | 5 | 7 | A(5.7) | A(2.5) |
| **D1** | D | 12 | 4 | A(12.4) | A(0.4) |

**Tableau 3.10** Les étapes 4, 6,7

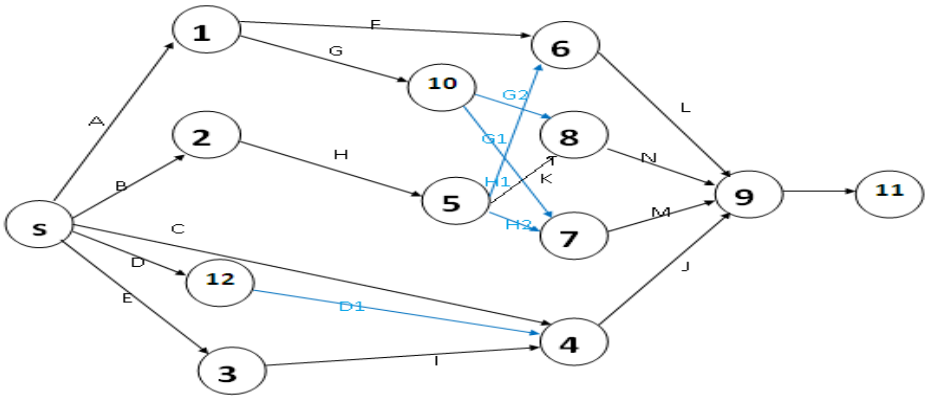



**Figure3.13 Le** graphe AoA final

## 6. Traduire l'AOA Généré à AON [24]

La construction du réseau AON de sa table de priorité (où tous les prédécesseurs immédiats de chaque activité sont énumérés) est une tâche triviale. Ainsi, décrivant les étapes qui génèrent la table de priorité de l'AOA généré, est suffisante pour fournir toutes les informations nécessaires pour générer un réseau AON. Les étapes qui génèrent le tableau de priorité à partir de l'angle d'attaque générée sont les suivantes:

1. Toutes les (non fictives) activités (réelles) sont attribuées des numéros de nœuds AON.

2. Dupliquer le réseau d'AOA avec le nouveau numéro de nœud à l'étape 1. Dans le réseau dupliqué, donner les arcs fictifs le même nombre que leurs prédécesseurs.

3. Pour chaque activité de AOA, trouver et lister le nombre de ses prédécesseurs immédiats (chiffres de l'étape 2). La liste obtenue est la contrainte de priorité de l'activité.

La sortie de l'étape 3 est une table de priorité. Ainsi, ces trois étapes suffisent pour accumuler toutes les informations nécessaires pour générer un réseau AON.

Après s'être assuré que chaque arc a sa propre combinaison unique de nœuds de début / fin, les noms d'arc doivent se référer à cette combinaison. Le nom de chaque activité est remplacée par une lettre et deux indices de marquage de ses nœuds de début et de fin (voir colonne 4 tableau 4).

Le tableau 4 est adapté pour la construction des activités sur les Arcs réseau. Il contient toutes les activités d'AOA (y compris les activités fictives) et les nœuds de début et de fin qui lui sont associés.

## 7. Conclusion

Nous avons présenté une approche pour générer des réseaux AOA: au lieu d'essayer de réduire au minimum le nombre d'arcs factifs (et perdre la différence entre prédécesseurs immédiats et non immédiats) un algorithme de construction du réseau AOA qui construit un réseau, dans lequel les arcs entrants de chaque nœud correspondant aux contraintes de priorité immédiate de l'arc sortant de nœuds. La méthode assure la génération d'un réseau d'AOA unique pour une table de priorité donnée. Alors que l'algorithme ne minimise pas le nombre d'arcs fictifs, il est très efficace: la méthode ajoute des arcs fictifs seulement pour pointer à plus d'une contrainte de priorité. Toute suppression fictive détruit cette logique de priorité immédiate, et tout autre arrangement fictif détruit l'analogie à la table de priorité. La nouvelle méthode présentée ici permet une construction systématique d'un réseau d'AOA unique à partir d'une liste de tâches et contraintes de précédence. Ainsi, il permet une traduction systématique avant et en arrière à partir d'un AON dans un réseau AOA.

# CHAPITRE 04
# L'OPTIMISATION DE L'ALGORITHME DE COHEN

## 1. Introduction

Les réseaux de projet pour la planification et l'ordonnancement sont basés sur deux types de représentation: Activités sur Arcs (AoA) et activités sur les nœuds (AoN). Chaque type de représentation (AoA ou AoN) a des caractéristiques souhaitables qui leurs sont propres. [18]

Par exemple, certains avantages des activités sur Arcs (AoA) réseaux pour la planification du réseau du projet sont : (*K. Agarawal, S. E. Elmaghraby 1996*) chaque arc dans le graphe AoA a son nœud d'origine et de destination, ou avec des variables une matrice d'arc de nœud. (*N. H. Ahuja, S. P. Dozzi 1994*) des réseaux de AoA pourraient être agencés dans une matrice totale uni-modulaire "node-arc" qui assure une solution entière pour les données de nombres entiers. (*Y. Cohen 2003*) AoA est adapté à certaines formulations populaires qui utilisent soustractions appariées des variables de nœud AoA.

Alors que la construction de graphe AoN est relativement facile, les réseaux AoA peuvent contenir des arcs factices et leur construction est beaucoup plus problématique. Toutefois, en raison de leur importance, La différence majeure réside donc dans la construction du graphe Toutefois, en raison de leur importance, La différence majeure réside donc dans la construction du graphe : Le graphe AoA est souvent plus difficile à construire que celui de la méthode AoN car on peut être amené à introduire des arcs fictifs qui ne correspondent à aucune tâche.

Dans la méthode AoA, chaque tâche est donc associée à un arc du graphe, la longueur de l'arc correspondant à la durée de la tâche en question. Les sommets sont utilisés pour traduire les relations de succession temporelle. Ainsi, si la tâche j doit suivre la tâche i, l'extrémité terminale de l'arc représentant la tâche i coïncidera avec l'extrémité initiale de l'arc représentant la tâche j. [18]

## 2. Etat de l'art

Maintenant nous avons un algorithme qui permet une traduction systématique à partir d'un AON vers un réseau AOA, mais ne minimise pas le nombre d'arcs fictifs qui est notre objectif.

Ce qu'on va faire maintenant c'est essayer d'améliorer l'algorithme pour qu'il donne le nombre minimal d'arcs fictifs, donc on a en entrée, l'algorithme de Cohen et Sadeh, ensuite on applique trois nouvelles règles pour réduire le nombre des tâches fictives.

Nous allons maintenant proposer une modification sur la méthode de Cohen et Sadeh par l'ajout de 3 règles. Nommons $G_{e1}$, le graphe PERT obtenu par l'algorithme de Cohen et Sadeh

### 2.1 Les trois règles

**La règle 01 :** Contraction des tâches ayant les mêmes successeurs ou les mêmes prédécesseurs

- Si I= $\{i_1,i_2,\cdots..i_p\}$ $p>1$ est un ensemble de tâches qui ont les mêmes successeurs et maximal pour cette propriété, alors contracter dans $G_{e1}$ les sommets $b_{i1},b_{i2}, \cdots,b_{ip}$ en un sommet unique $b_I$

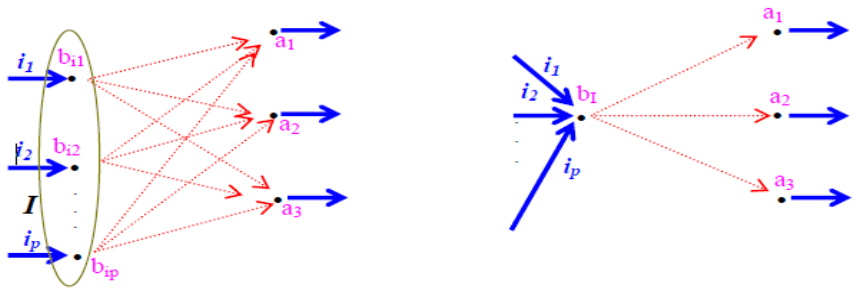



**Figure 4.14** Les sommets ayant les mêmes successeurs contractés en un sommet unique dans Ge1.

Eliminer les arcs rouges multiples et répéter cette règle autant de fois que possible pour obtenir le graphe Ge2.

Noter que dans $G_{e2}$, il n'y a aucun arc rouge qui a $b_i$ comme extrémité initiale.

- Si I= $\{i1,i2,.....ip\}$ $p>1$ est un ensemble de tâches qui ont les mêmes prédécesseurs et maximal pour cette propriété, alors contracter dans Ge2 les sommets $ai1,ai2, ....,aip$ en un sommet unique $aI$.

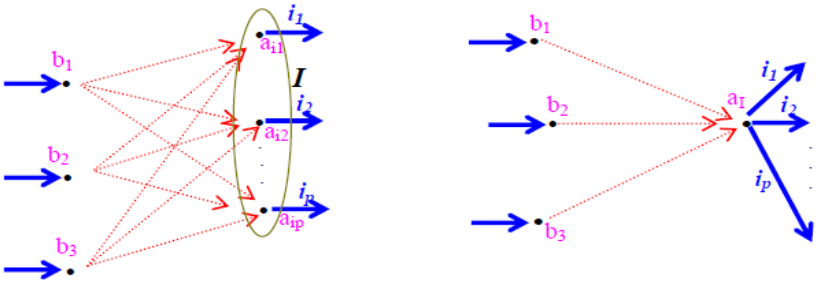



**Figure 4.15** Les sommets ont les mêmes prédécesseurs contractés en un sommet unique dans Ge2.

Eliminer les arcs rouges multiples et répéter cette règle autant de fois que possible pour obtenir le graphe Ge3.

Ge3 n'a pas de sous ensembles de sommets avec les mêmes successeurs ou prédécesseurs. Aucun arc rouge n'a un *ai* comme extrémité initiale ou *bi* comme extrémité terminale.

Les différents chemins de Ge3 sont formés alternativement, d'arcs bleus et rouges. Il n'y a pas un chemin direct de longueur ≥ 2 de même couleur.

**La règle 02 :** Suppression des arcs transmetteurs de type 1 et 2

- On appelle arc transmetteur de type 1 tout arc fictif qui vérifie la condition suivante : le nombre d'arcs incidents intérieurement à $a_i = +1$ ($d^-_{ai} = +1$).
- On appelle arc transmetteur de type 2 tout arc fictif (rouge) qui vérifie la condition suivante : le nombre d'arcs incidents extérieurement à $b_i = +1$ ($d^+_{ai} = +1$).

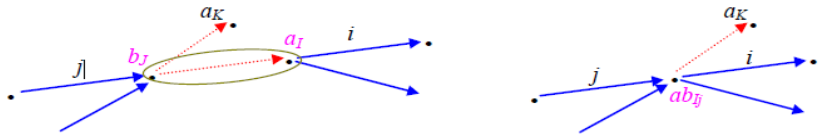



**Figure 4.16** Un arc transmetteur de type 1 entre deux tâches i et j et sa contraction.

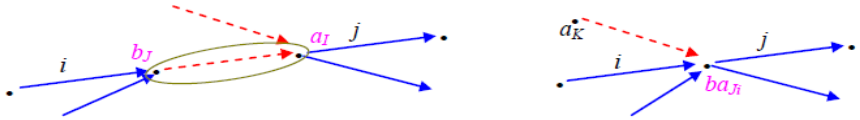



**Figure 4.17** Un arc transmetteur de type 2 entre deux tâches i et j et sa contraction.

**La règle 03 :** Les tâches dont les successeurs/prédécesseurs sont inclus dans l'ensemble des successeurs/prédécesseurs d'une autre tâche. Dans Ge5 cherchons s'il existe une tâche dont ses successeurs sont inclus dans les successeurs d'une autre tâche. Tous les arcs rouges qui sortent de et qui sont inclus dans les successeurs d'une autre tâche seront supprimés et remplacés par un arc fictif.

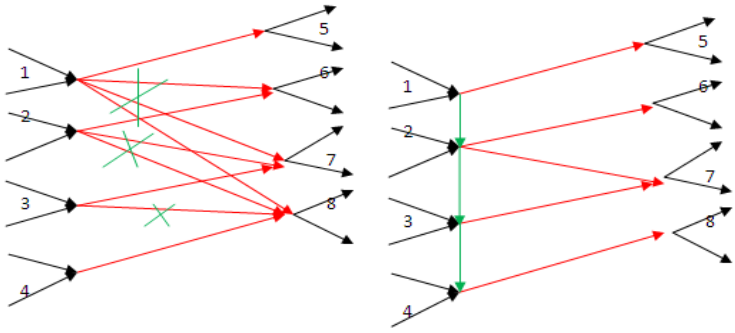



**Figure 4.18** Les tâches dont les successeurs sont incluses dans l'ensemble des successeurs d'une autre tâche dans Ge5 et la réduction des tâches fictives dans Ge6.

S'il existe une tâche dont ses prédécesseurs sont inclus dans les prédécesseurs d'une autre tâche, tous les arcs rouges qui arrivent à et qui sont inclus dans les prédécesseurs d'une autre tâche seront supprimés et remplacés par un arc fictif (rouge).

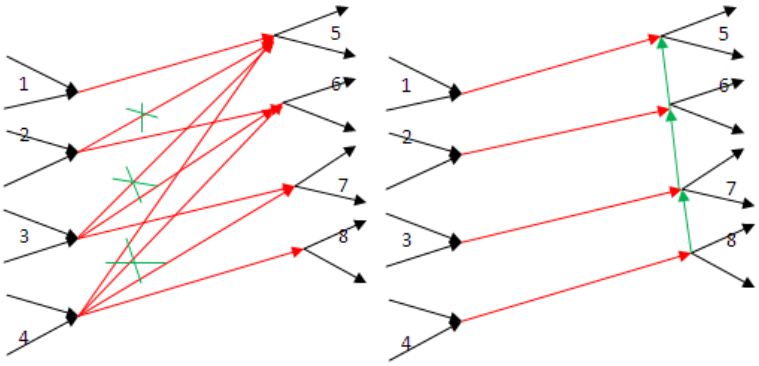



**Figure 4.19** Les tâches dont les prédécesseurs sont incluses dans l'ensemble des prédécesseurs d'une autre tâche dans Ge5 et la réduction des tâches fictives dans Ge6.

### 2.2 Amélioration d'un exemple

Considérons l'exemple suivant :

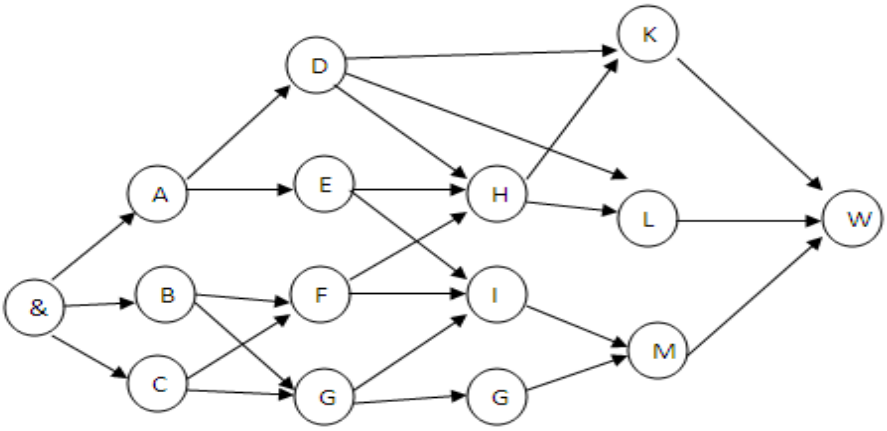



**Figure 4.20** Le graphe AoN initial

Appliquons l'algorithme de Cohen et Sadeh sur ce graphe, ce qui donne le résultat (graphe AOA) suivant :

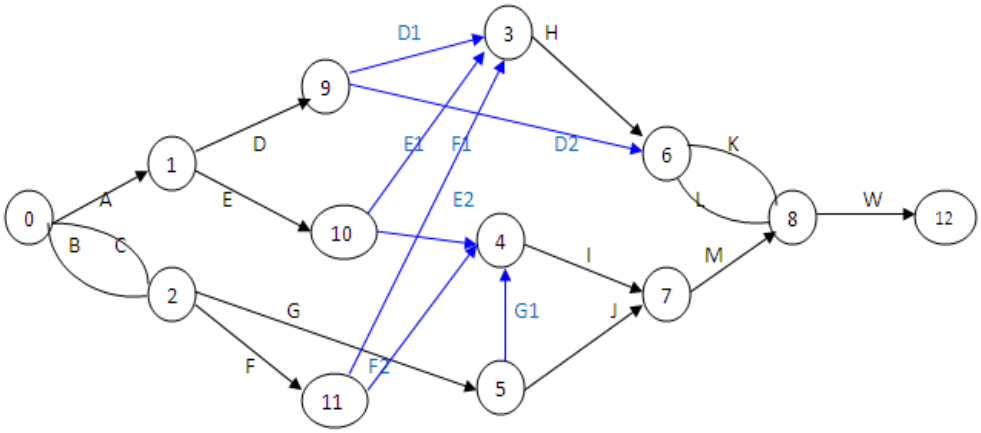



**Figure 4.21** Le graphe AoA en appliquant l'algorithme de Cohen avec 7 tâches fictives

Appliquons maintenant les trois règles sur le graphe Cohen AoA final de l'exemple précédant :

La règle 01 **:** Contraction les tâches ayant les mêmes successeurs ou les mêmes prédécesseurs

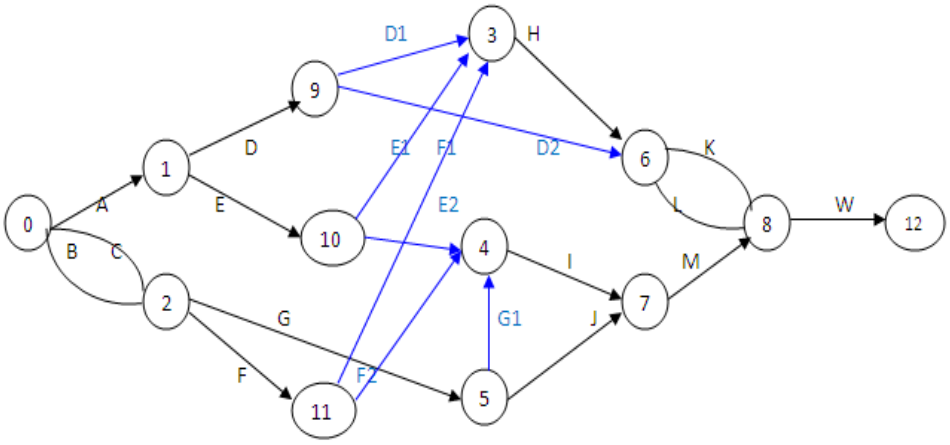



**Figure 4.22** Le graphe Cohen AoA final

La règle 02 : Suppression des arcs transmetteurs

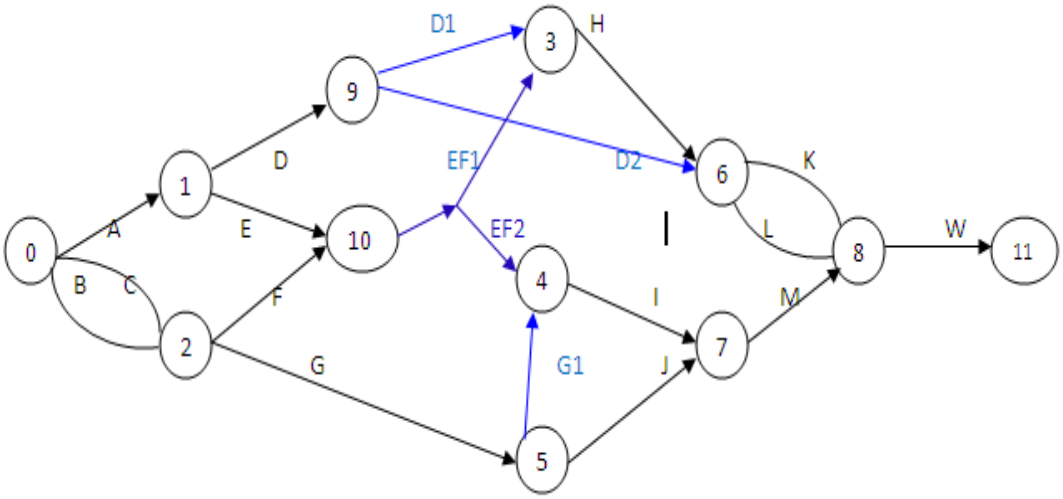



**Figure 4.23** Le graphe après la contraction les tâches ayant les mêmes prédécesseurs

La règle 03 : Les tâches dont les successeurs/prédécesseurs sont inclus dans l'ensemble des successeurs/prédécesseurs d'une autre tâche (Mais on ne peut pas appliquer cette étape car dans ce cas l'exemple courant n'existe pas)

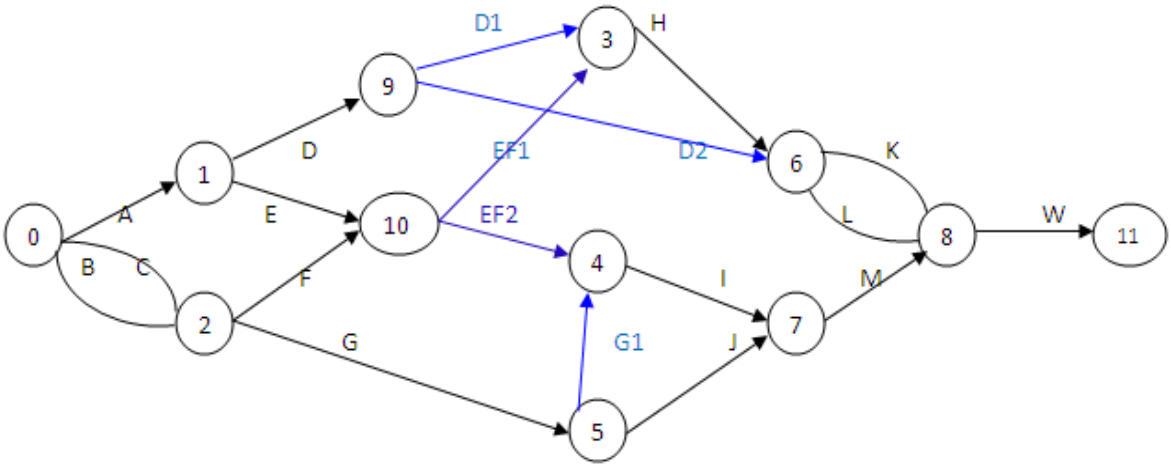



**Figure 4.24** Le graphe AOA final (après la suppression des arcs transmetteurs) avec 5 tâches fictives.

On remarque que le nombre de tâches fictives a encore été minimisé grâce à l'amélioration proposée, pour arriver à cinq tâches fictives au final.

## 3 Conclusion

Ce chapitre présente une amélioration de l’algorithme de Cohen et Sadeh pour générer des réseaux AoA avec un nombre minimum d'arcs fictifs. En effet dans l’algorithme de Cohen et Sadeh en appliquant les sept règles R1, R2, ..., R7 dans l‘ordre séquentiel, l'algorithme permet de transformer un graphe potentiel ou un table de priorité jusqu’à la dernière règle qui donne le graphe AoA PERT, mais il ne minimise pas le nombre de tachés fictives. C'est ce que nous avons travaillé pour l'optimiser en appliquant nos trois règles.

Le nouvel algorithme est très simple à appliquer. Elle donne le AoA minimal avec un certain nombre d'arcs virtuels dans un temps très court.

Les techniques utilisées dans les trois règles de l'algorithme peuvent être exploitées dans d'autres domaines par des spécialistes de la théorie des graphes. Les résultats expérimentaux sont très positifs par rapport à l'algorithme de Cohen et Sadeh, même lorsque les réseaux sont d'une très grande taille.

# CHAPITRE 05

# IMPLEMENTATION

## 1. Introduction

La partie théorique a porté sur l'étude des principes de base de l'ordonnancement et de la gestion de projet ainsi que sur l'étude de l'algorithme de Cohen et Sadeh. Ce dernier fait une traduction systématique à partir d'un AON vers un réseau AOA sans minimiser les arcs fictifs.

Dans ce chapitre, nous aborderons la phase de l'implémentation (partie pratique). Commençons par présenter les outils et environnement de développement que nous allons utiliser. Ensuite, nous allons présenter l'application et ses fonctionnalités et afficher enfin le résultat numérique de l'algorithme avant et après notre amélioration.

## 2. Implémentation de l'algorithme Cohen et Sadeh

**DEBUT**

Nb : nombre de sommets ;

Num : numéro de chaque tâche ;

**POUR** chaque sommet i de x **FAIRE**

Constitué numéro de tout i de x ;

**FIN POUR ;**

**Régle1 : Construction de graphe Fv1**

**POUR** i allant de 1 jusqu'à nb **FAIRE**

**SI** prédécesseur[i]='' **ALORS**

Prédécesseur[i]= aucun ;

**SINON**

Prédécesseur[i]=j ;

**FIN SI**

**FIN POUR ;**

**Règle 2 : construction de graphe Fv2**

**POUR** i allant de 1 jusqu'à nb **FAIRE**

**SI** prédécesseur[i]=' ' **ALORS**

Num- prédécesseur[i]=0 ;

**FIN SI**

K=1 ;

**POUR** j allant de 1 jusqu'à nb **FAIRE**

**SI** (nb-successeur[i]>1) et (nb- prédécesseur[i]=1) **ALORS**

Num- prédécesseur[i]=Num[i] + k ;

**FIN SI**

K=k+1 ;

**FIN POUR ;**

**Règle 3 et 4 : construction de graphe Fv3**

**POUR** i allant de 1 jusqu'à nb **FAIRE**

**SI** (nb prédécesseur[i]=0) ou ( nb-successeur[i]=1) **ALORS**

**FAIT** rien () ;

**FIN SI ;**

**SI** (nb prédécesseur[i] != 0) et (nb-successeur[i]>1) **ALORS**

Ajouter sommet fictif () ;

**FIN SI ;**

**FIN POUR ;**

**Règle 5 : construction de graphe Fv5**

**POUR** i allant de 1 jusqu'à nb **FAIRE**

**SI** (nb-successeur[i]=0) ou (nb-successeur[i]>=2) **ALORS**

Ajouter Num-successeur[i] ;

**FIN SI**

**SI** (nb-successeur[i]=1) **ALORS**

Num-successeur[i]=Num[i] ;

**FIN SI ;**

**FIN POUR ;**

**Règle 6 : construction de graphe Fv6**

h=1 ;

**POUR** i allant de 1 jusqu'à nb **FAIRE**

**SI** Num[i] =Num[i+h] **ALORS**

- insérer sommet fictivf ();
- insérer Num-prédécesseur et Num-successeur des sommets fictifs ();

**FIN SI**

h=h+1 ;

**FIN POUR ;**

**Règle 7 : construction de graphe Fe7 partir de graphe Fv**

**POUR** chaque sommet i de X **FAIRE**

-on définit deux sommets dans graphe à la fin ;

- ai : sommet de début de tâche (prédécesseur[i]) ;
- bi : sommet de fin de tâche (successeur[i]) ;

-on constitue des arcs (ai,bi) pour tout i de X ;

**FIN POUR ;**

Nb=0 ;

**POUR** i allant de 1 à nb **FAIRE**

**SI** ij et ij+1 en mêmes successeurs **ALORS**

Nb=nb+ij ;

Nb=nb+ij+1 ;

**FIN SI ;**

-Contracter les sommets bi1, bi2,….., bij en mêmes successeurs en un seul sommet bi ;

**FIN POUR ;**

**POUR** i allant de 1 à nb **FAIRE**

**SI** ij et ij+1en mêmes prédécesseurs **ALORS**

nb = nb+ij ;

nb = nb+ij+1 ;

**FIN SI ;**

-Contracter les sommets ai1, ai2, ……..aij en mêmes prédécesseurs en un seul sommet ai ;

**FIN POUR ;**

**FIN.**

## 3. Les outils et l'environnement de développement

### 3.1 Le langage de programmation "C#/Sharp"

Le langage C# (C Sharp) est un langage objet créé spécialement pour le Framework Microsoft .NET. L'équipe qui a créé ce langage a été dirigée par Anders Hejlsberg, un informaticien danois qui avait également été à l'origine de la conception du langage Delphi pour la société Borland (évolution objet du langage Pascal). Le Framework .NET est un environnement d'exécution (CLR Common Language Runtime) ainsi qu'une bibliothèque de classes (plus de 2000 classes). L'environnement d'exécution (CLR) de .NET est une machine

virtuelle comparable à celle de Java. Le Runtime fournit des services aux programmes qui s'exécutent sous son contrôle : chargement/exécution, isolation des programmes, vérification des types, conversion d'un code intermédiaire (IL) vers un code natif, accès aux métadonnées (informations sur le code contenu dans les assemblages .NET), vérification des accès mémoire (évite les accès en dehors de la zone allouée au programme), gestion de la mémoire (Garbage Collector), gestion des exceptions, adaptation aux caractéristiques nationales (langue, représentation des nombres), compatibilité avec les DLL et modules COM qui sont en code natif (code non managé). Les classes .NET peuvent être utilisées par tous les langages prenant en charge l'architecture .NET. Les langages .NET doivent satisfaire certaines spécifications : utiliser les mêmes types CTS (Common Type System), les compilateurs doivent générer un même code intermédiaire appelé MSIL (Microsoft Intermediate Language). Le MSIL (contenu dans un fichier .exe) est pris en charge par le Runtime .NET qui le fait tout d'abord compiler par le JIT compiler (Just In Time Compiler). La compilation en code natif a lieu seulement au moment de l'utilisation du programme .NET. Définir un langage .NET revient à fournir un compilateur qui peut générer du langage MSIL. Les spécifications .NET sont publiques (Common Language Spécifications) et n'importe quel éditeur de logiciel peut donc concevoir un langage/un compilateur .NET. Plusieurs compilateurs sont actuellement disponibles : C++.NET (version Managée de C++), VB.NET, C#, Delphi, J# Lors du développement d'applications .NET, la compilation du code source produit du langage MSIL contenu dans un fichier .exe. Lors de la demande d'exécution du fichier .exe, le système d'exploitation reconnaît que l'application n'est pas en code natif. Le langage IL est alors pris en charge par le moteur d'exécution du Framework .NET qui en assure la compilation et le contrôle de l'exécution. Un des points importants étant que le Runtime .NET gère la récupération de mémoire allouée dans le tas (garbage collector), à l'instar de la machine virtuelle Java. "En .NET, les langages ne sont guères plus que des interfaces syntaxiques vers les bibliothèques de classes." Formation à C#, Tom Archer, Microsoft Press. On ne peut rien faire sans utiliser des types/classes fournis par le Framework .NET puisque le code MSIL s'appuie également sur les types CTS. L'avantage de cette architecture est que le Framework .NET facilite l'interopérabilité (projets constitués de sources en différents langages). Avant cela, faire cohabiter différents langages pour un même exécutable imposait des contraintes de choix de types “communs” aux langages ainsi que de respecter des conventions d'appel de fonctions pour chacun des langages utilisés. [02]

### 3.2 L'environnement Microsoft Visual Studio

Microsoft Visual Studio est un environnement de développement intégré (IDE) de Microsoft. Il peut être utilisé pour développer des applications d'interface utilisateur console et graphique avec les applications Windows Forms, les sites Web, les applications Web et les services Web dans le code natif avec le code géré pour toutes les plates-formes prises en charge par Microsoft Windows, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework et Microsoft Silverlight.

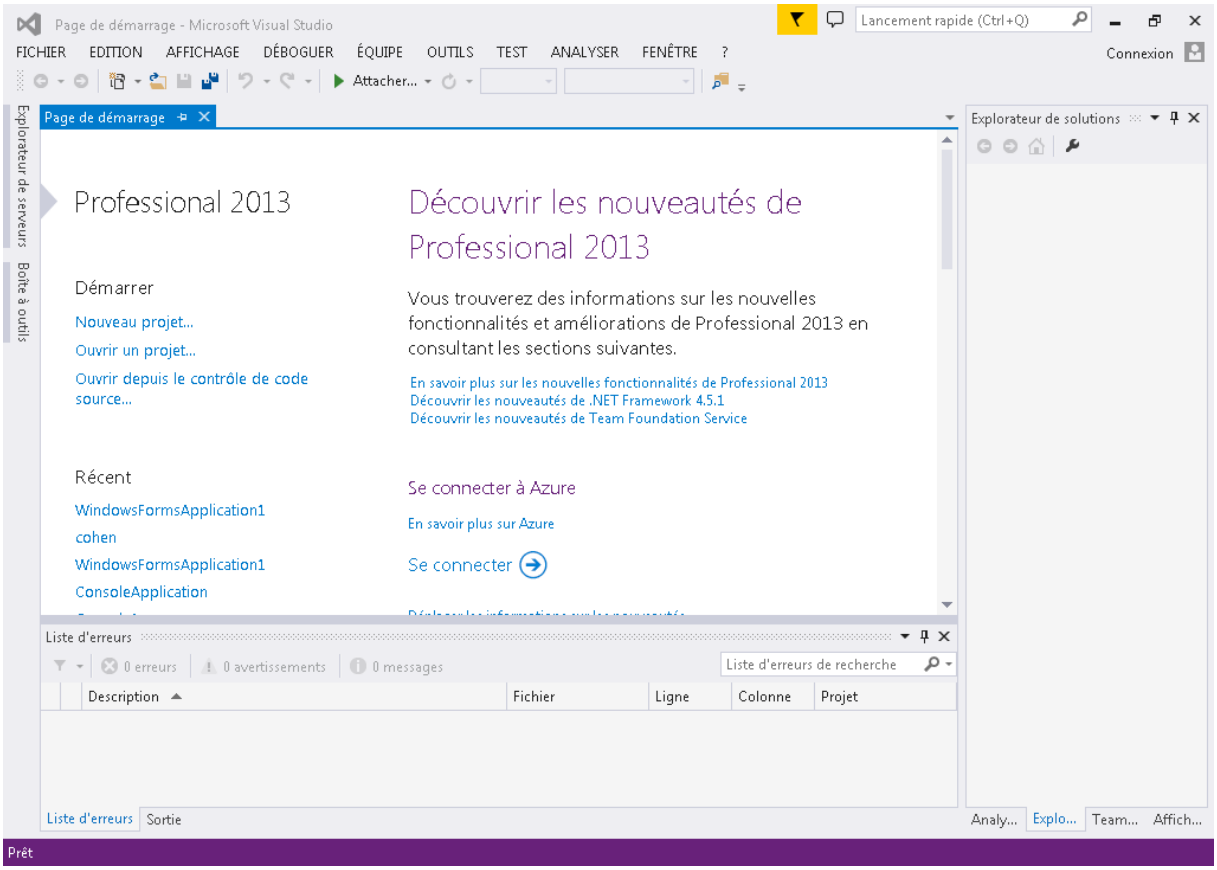



**Figure 5.25** Visual Studio Professional 2013

### 3.3 Les versions Antérieures [26]

- C# 1, Visual Studio .NET 2002 : Première version
- C# 1.1, Visual Studio .NET 2003 : `#line` pragma et commentaires de documents xml
- C# 2, Visual Studio .NET 2005 : Méthodes anonymes, génériques, types Nullable, iterateurs/rendement, classes `static`, variance co/contra pour les délégués
- C# 3, Visual Studio .NET 2008 : Initialiseurs d'objet et de collection, expressions lambda, méthodes d'extension, types anonymes, propriétés automatiques, Language

Integrated Query (LINQ), types anonymes, inférence de type de variable var locale, LINQ

- C# 4, Visual Studio .NET 2010 : Dynamic, arguments nommés, paramètres facultatifs, variance co/contra générique
- C# 5, Visual Studio .NET 2012 : Async / await, attributs des informations sur l'appelant
- Visual Studio .NET 2013 : Correctifs de bogues, améliorations des performances et aperçus des technologies de la plateforme des compilateurs .NET (« Roslyn »)
- C# 6, Visual Studio .NET 2015 : Version actuelle.

### 3.4 Pourquoi utiliser le langage C# ? [26]

C# est un langage élégant, simple, de type sécurisé et orienté objet qui permet aux programmeurs en entreprise de créer une vaste gamme d'applications.
Le langage C# permet également de créer des composants de niveau système durables grâce aux fonctionnalités suivantes :

- Prise en charge totale de plate-forme/COM pour l'intégration du code existant.
- Robustesse grâce au gerbage collection et à la sécurité de type.
- Sécurité assurée par des mécanismes d'approbation de code intrinsèque.
- Prise en charge intégrale des concepts de métadonnées extensibles.

Il est également possible d'inter opérer avec d'autres langages, sur plusieurs plates-formes, en reprenant les données héritées des premiers langages Basic, grâce aux fonctionnalités suivantes :

- Prise en charge intégrale de l'interopérabilité via les services COM+ 1.0 et .NET Framework avec un accès strict basé sur les bibliothèques.
- Prise en charge de XML pour l'interaction des composants Web.
- Version permettant de faciliter l'administration et le déploiement.

### 3.5 Conception architecturale

La conception architecturale a pour but de décomposer le logiciel en composants plus simples. On précisé les interfaces et les fonctions de chaque composant. On obtient une description de l'architecture de logiciel.

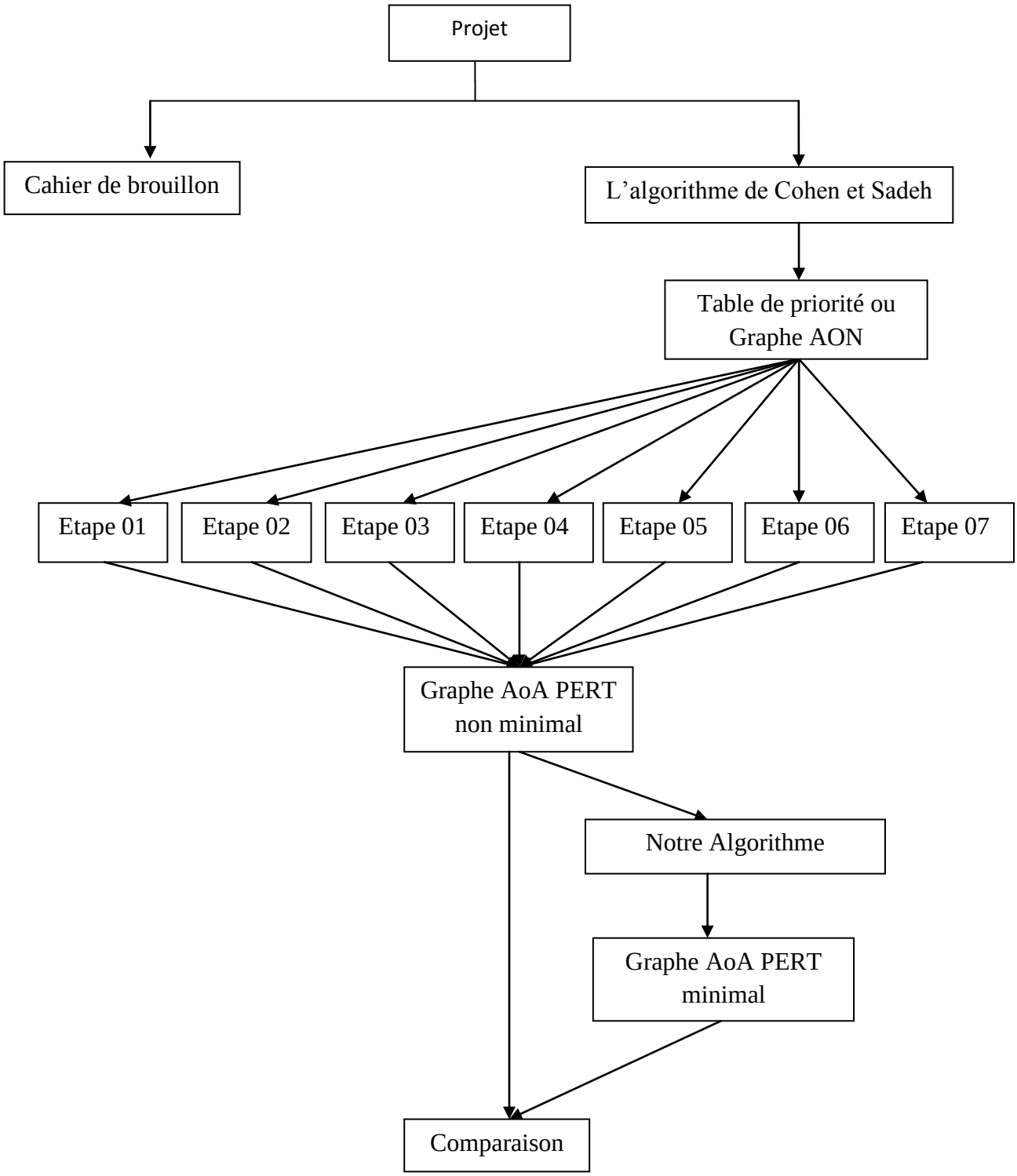



**Figure 5.26** Architecture de projet

## 4. Présentation les interfaces de l'application

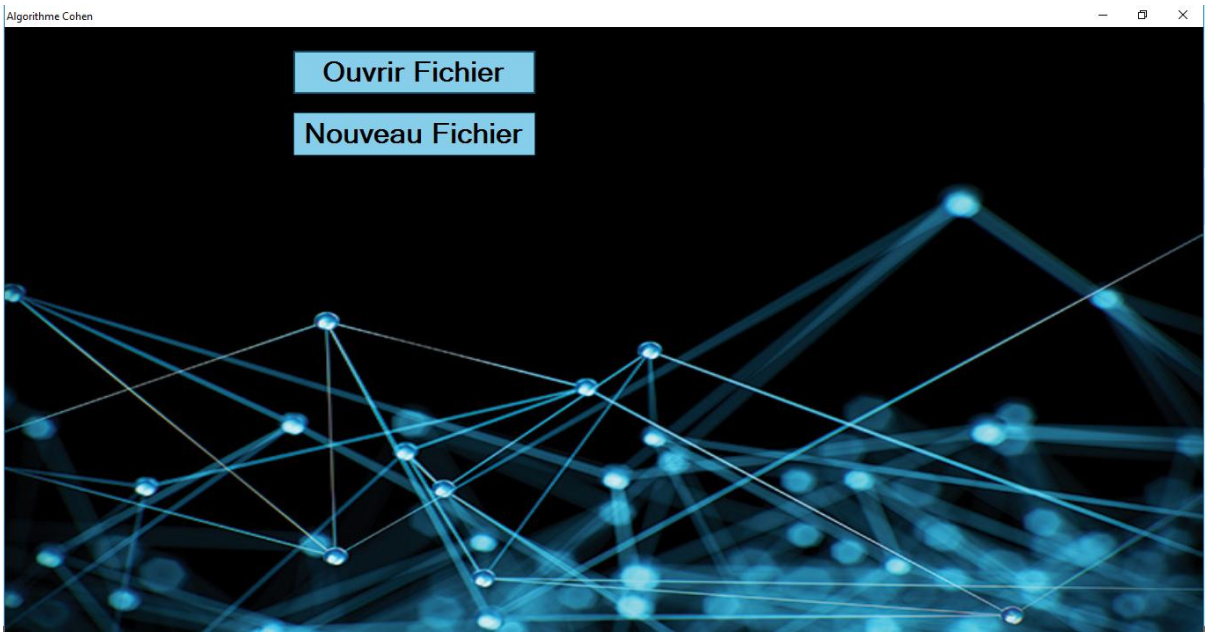



**Figure 5.27** Fenêtre de démarrage (N°1)

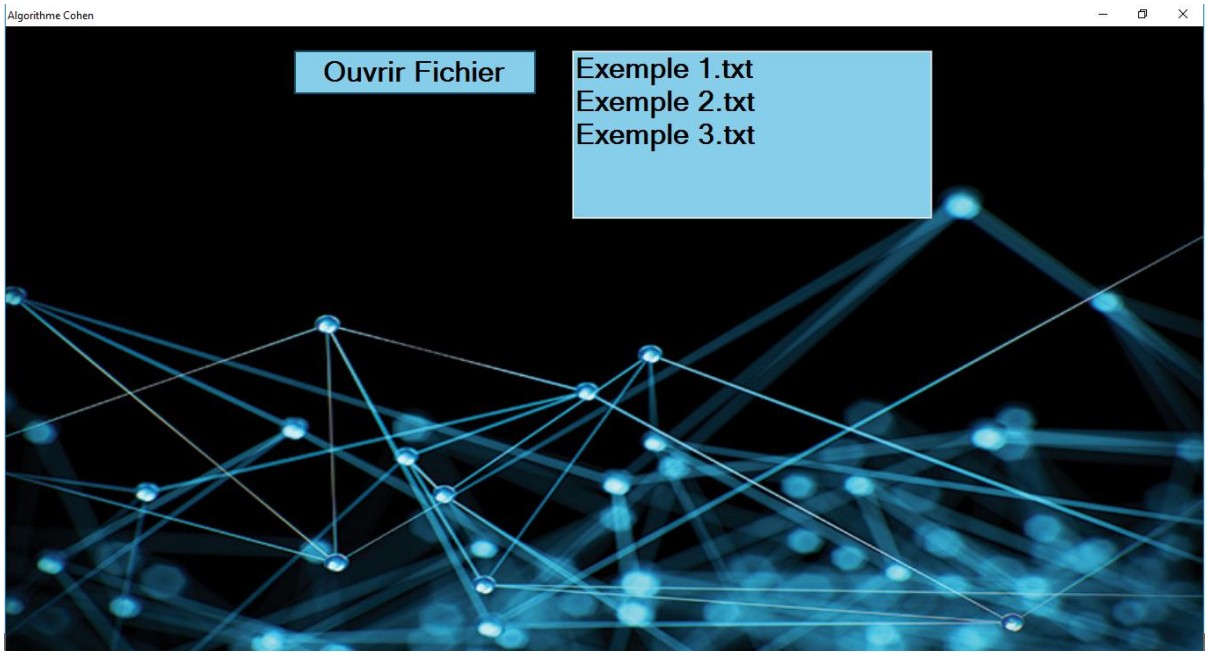
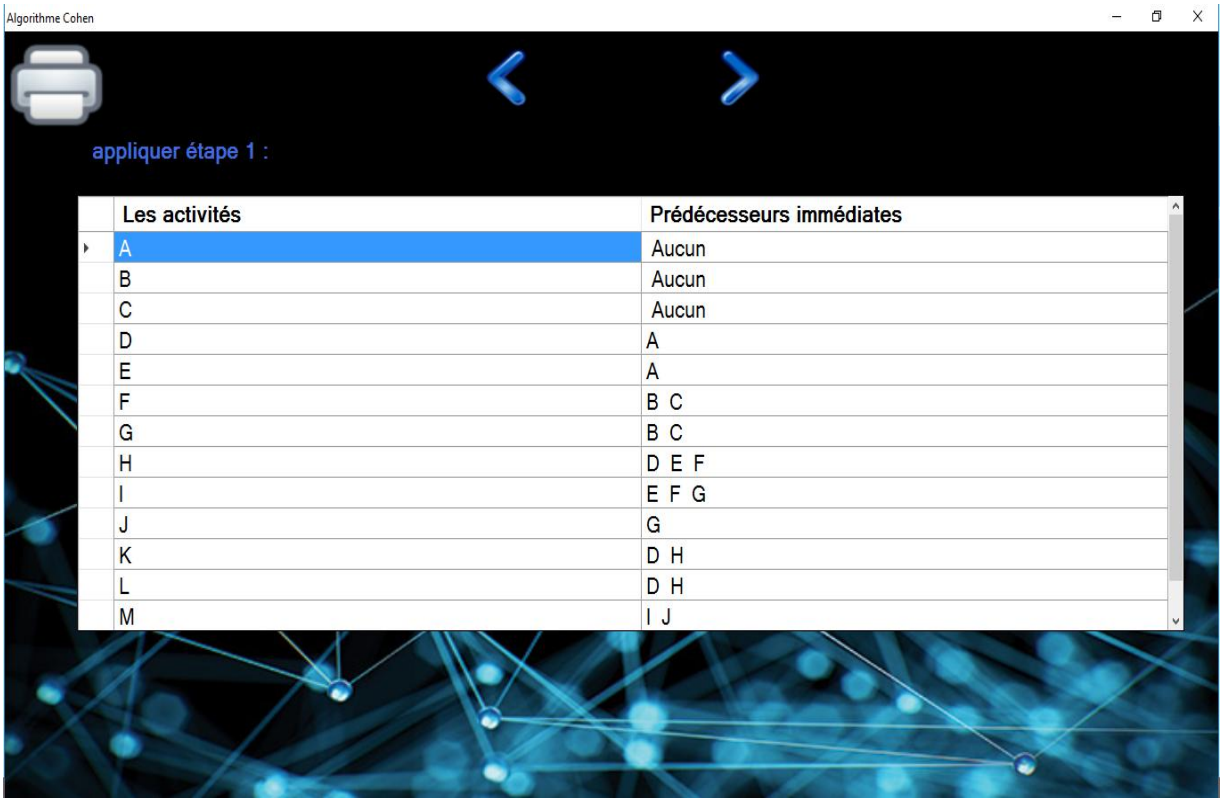
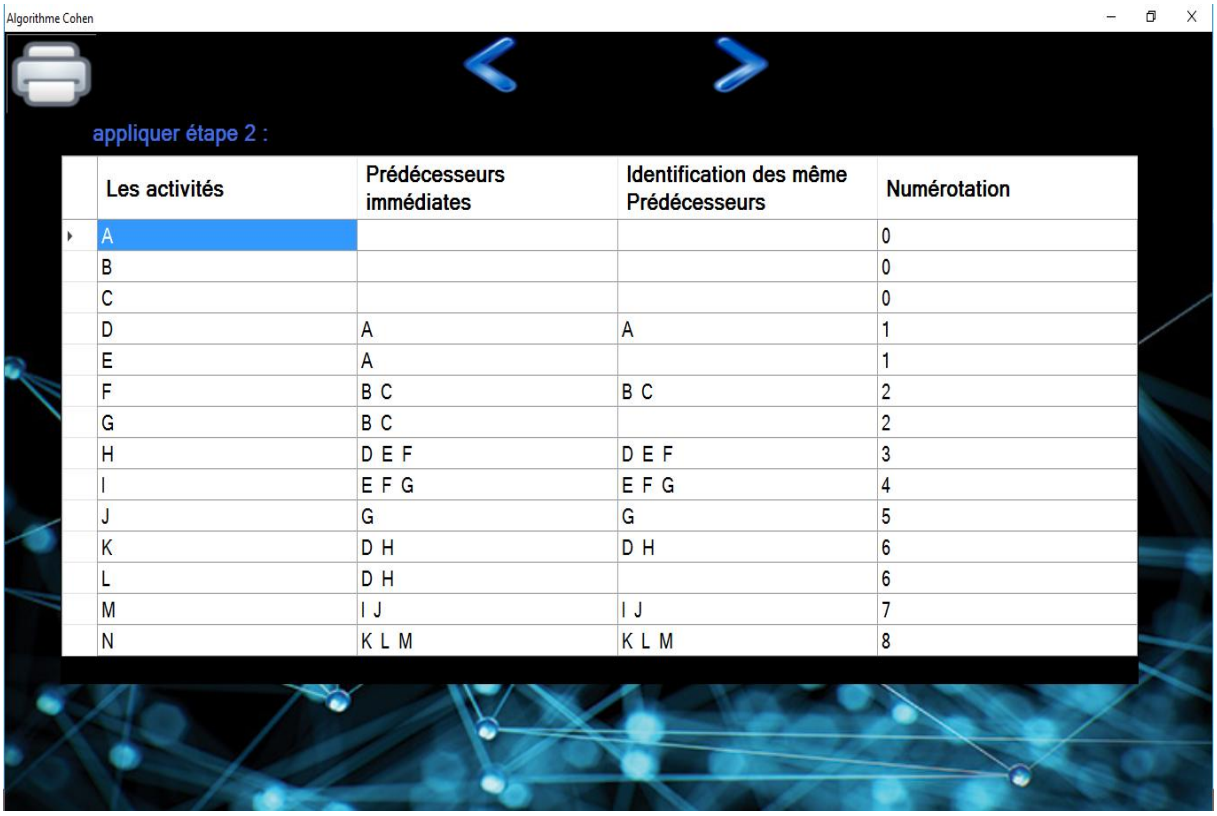



**Figure 5.28** Choisissez un exemple par l'ouverture d'un fichier



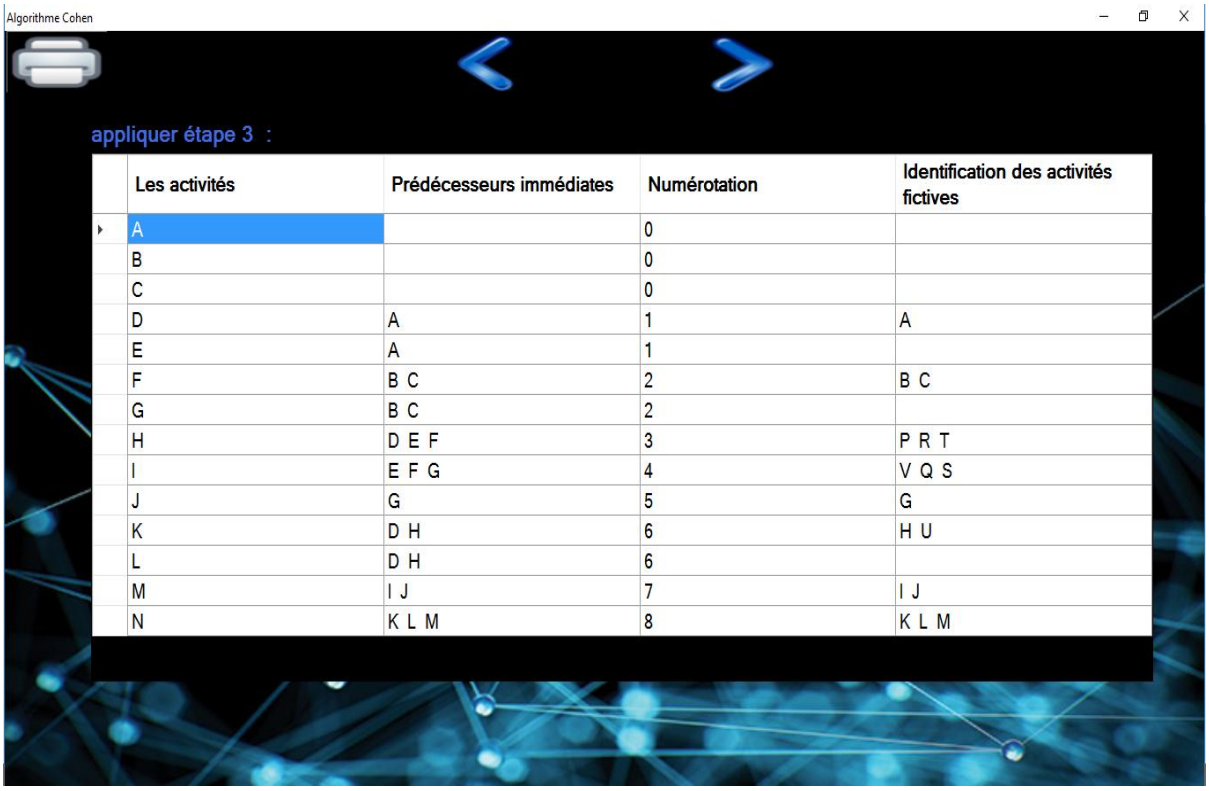

| Les activités | Prédécesseurs immédiates |
|---|---|
| A | Aucun |
| B | Aucun |
| C | Aucun |
| D | A |
| E | A |
| F | B C |
| G | B C |
| H | D E F |
| I | E F G |
| J | G |
| K | D H |
| L | D H |
| M | I J |

**Figure 5.29** L'application de l'étape 01 de l'algo. de Cohen et al.



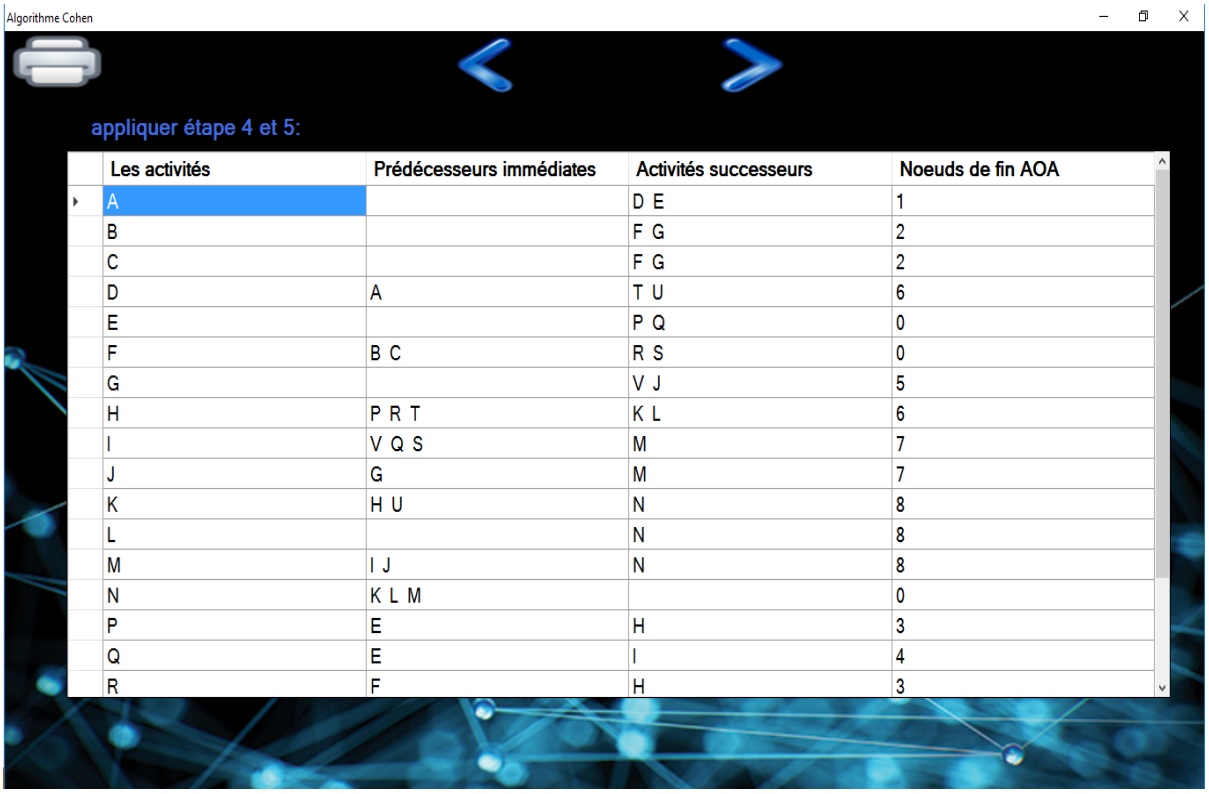

| Les activités | Prédécesseurs immédiates | Identification des même Prédécesseurs | Numérotation |
|---|---|---|---|
| A | | | 0 |
| B | | | 0 |
| C | | | 0 |
| D | A | A | 1 |
| E | A | | 1 |
| F | B C | B C | 2 |
| G | B C | | 2 |
| H | D E F | D E F | 3 |
| I | E F G | E F G | 4 |
| J | G | G | 5 |
| K | D H | D H | 6 |
| L | D H | | 6 |
| M | I J | I J | 7 |
| N | K L M | K L M | 8 |

**Figure 5.30** L'application de l'étape 02 de l'algo. de Cohen et al.



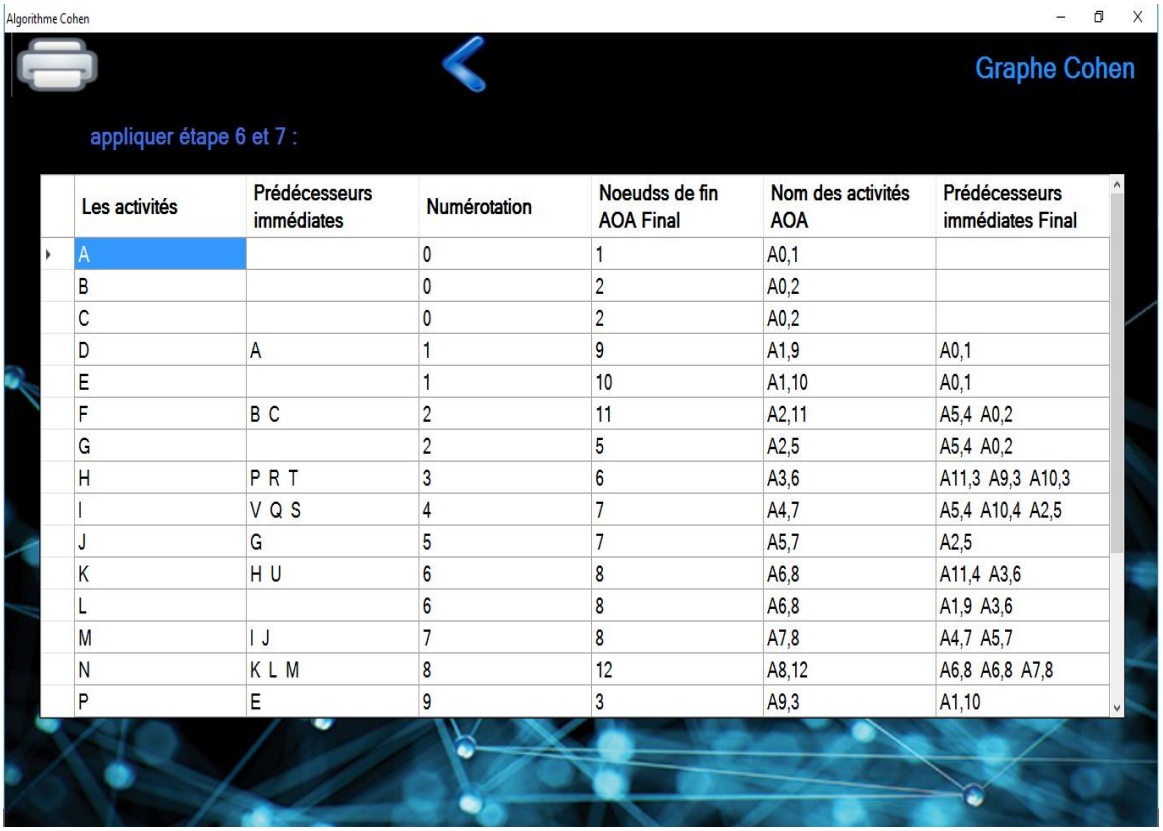

| Les activités | Prédécesseurs immédiates | Numérotation | Identification des activités fictives |
|---|---|---|---|
| A | | 0 | |
| B | | 0 | |
| C | | 0 | |
| D | A | 1 | A |
| E | A | 1 | |
| F | B C | 2 | B C |
| G | B C | 2 | |
| H | D E F | 3 | P R T |
| I | E F G | 4 | V Q S |
| J | G | 5 | G |
| K | D H | 6 | H U |
| L | D H | 6 | |
| M | I J | 7 | I J |
| N | K L M | 8 | K L M |

**Figure 5.31** L'application de l'étape 03 de l'algo. de Cohen et al.




| Les activités | Prédécesseurs immédiates | Activités successeurs | Noeuds de fin AOA |
|---|---|---|---|
| A | | D E | 1 |
| B | | F G | 2 |
| C | | F G | 2 |
| D | A | T U | 6 |
| E | | P Q | 0 |
| F | B C | R S | 0 |
| G | | V J | 5 |
| H | P R T | K L | 6 |
| I | V Q S | M | 7 |
| J | G | M | 7 |
| K | H U | N | 8 |
| L | | N | 8 |
| M | I J | N | 8 |
| N | K L M | | 0 |
| P | E | H | 3 |
| Q | E | I | 4 |
| R | F | H | 3 |

**Figure 5.32** L'application de l'étape 04 et 05 de l'algo. de Cohen et al.




| Les activités | Prédécesseurs immédiates | Numérotation | Noeudss de fin AOA Final | Nom des activités AOA | Prédécesseurs immédiates Final |
|---|---|---|---|---|---|
| A | | 0 | 1 | A0,1 | |
| B | | 0 | 2 | A0,2 | |
| C | | 0 | 2 | A0,2 | |
| D | A | 1 | 9 | A1,9 | A0,1 |
| E | | 1 | 10 | A1,10 | A0,1 |
| F | B C | 2 | 11 | A2,11 | A5,4 A0,2 |
| G | | 2 | 5 | A2,5 | A5,4 A0,2 |
| H | P R T | 3 | 6 | A3,6 | A11,3 A9,3 A10,3 |
| I | V Q S | 4 | 7 | A4,7 | A5,4 A10,4 A2,5 |
| J | G | 5 | 7 | A5,7 | A2,5 |
| K | H U | 6 | 8 | A6,8 | A11,4 A3,6 |
| L | | 6 | 8 | A6,8 | A1,9 A3,6 |
| M | I J | 7 | 8 | A7,8 | A4,7 A5,7 |
| N | K L M | 8 | 12 | A8,12 | A6,8 A6,8 A7,8 |
| P | E | 9 | 3 | A9,3 | A1,10 |

**Figure 5.33** L'application de l'étape 06 et 07 de l'algo. de Cohen et al.

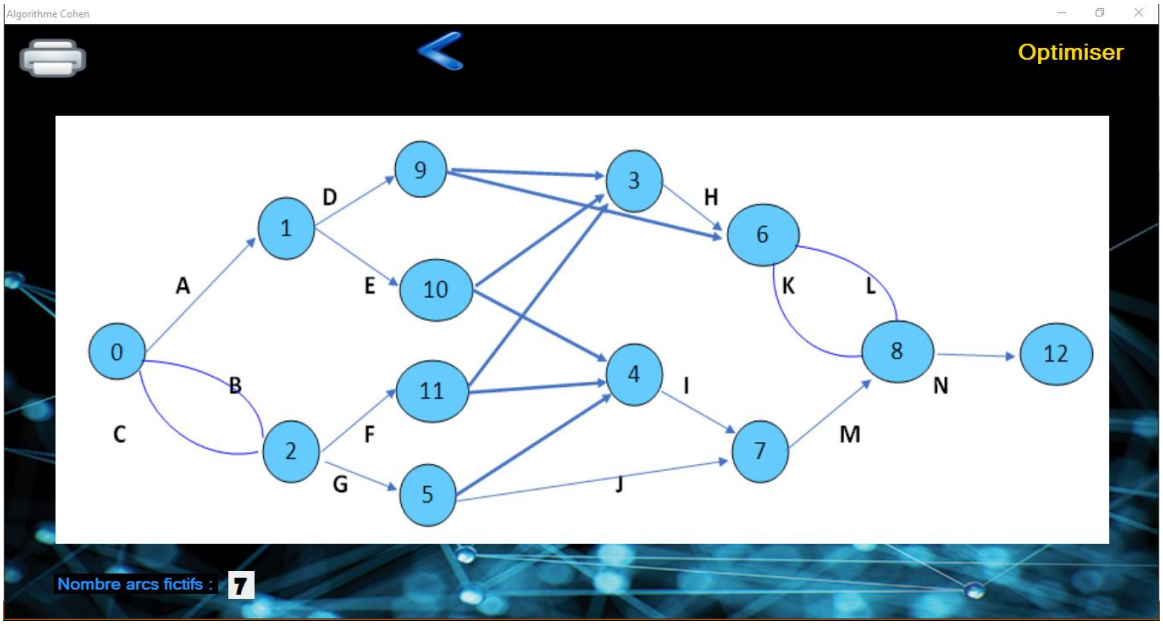



**Figure 5.34** Le graphe de Cohen AoA final (7 tâches fictives) de l'algo. de Cohen et al.

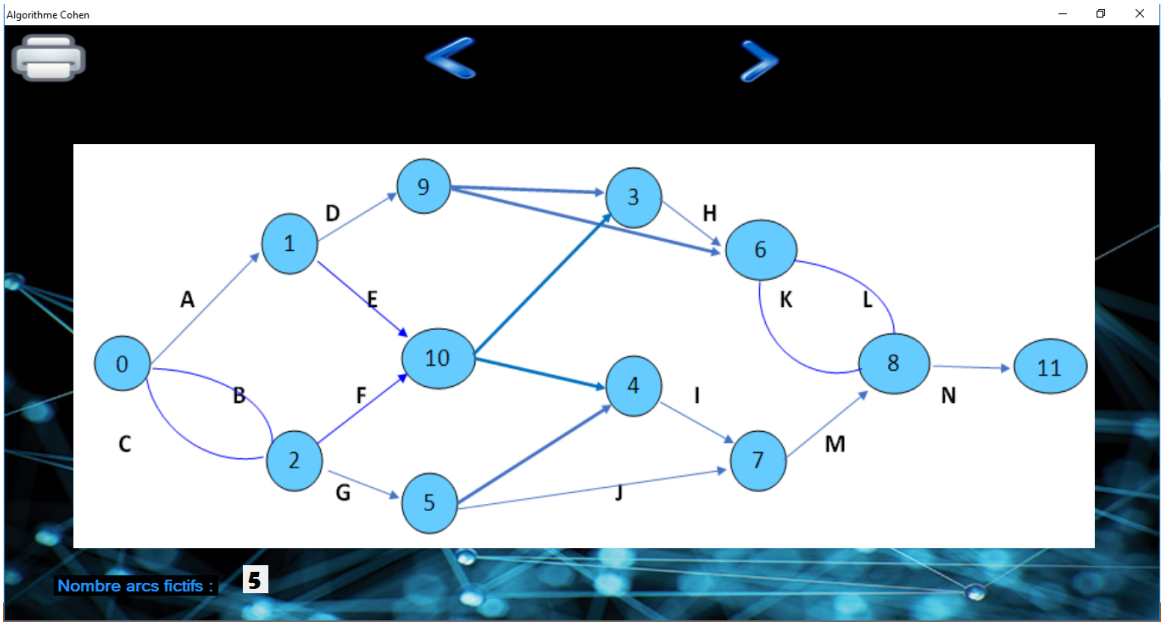
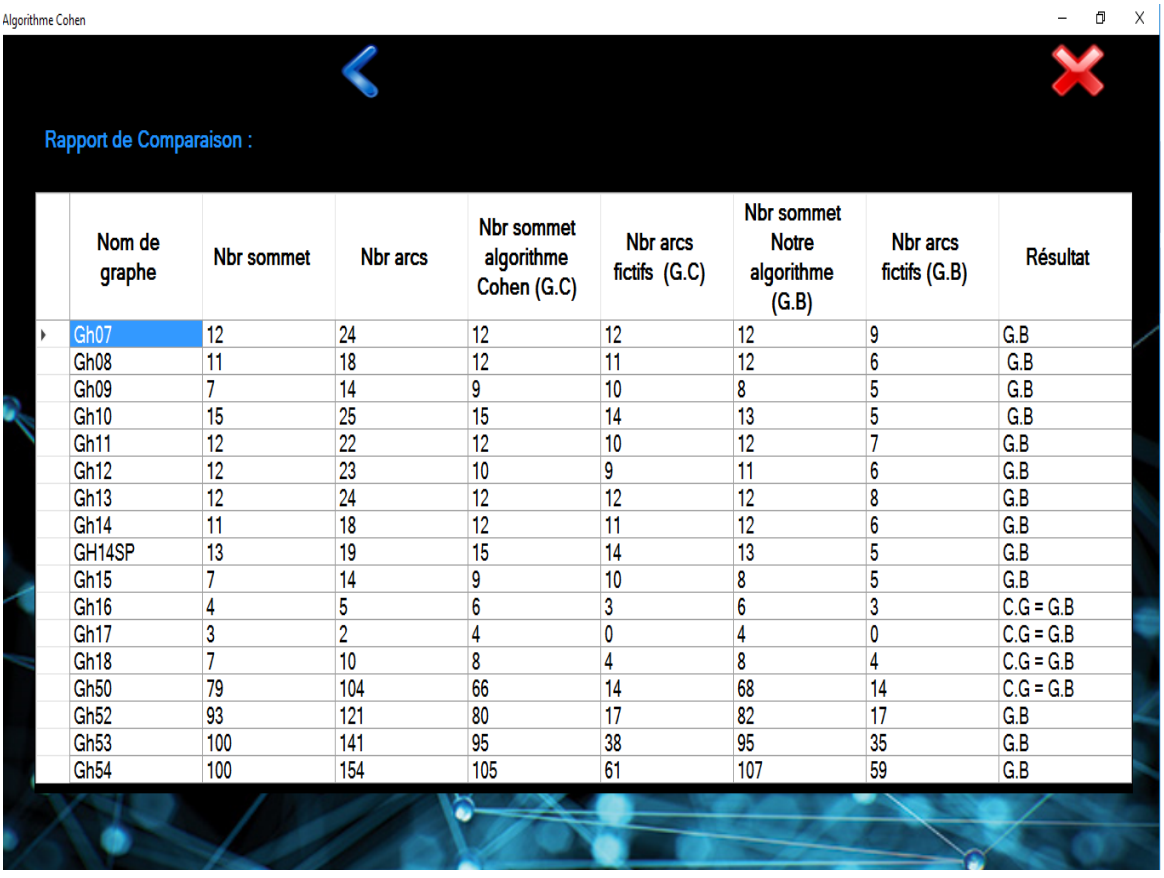



**Figure 5.35** Notre graphe AoA après l'optimisation (5 tâches fictives)




| Nom de graphe | Nbr sommet | Nbr arcs | Nbr sommet algorithme Cohen (G.C) | Nbr arcs fictifs (G.C) | Nbr sommet Notre algorithme (G.B) | Nbr arcs fictifs (G.B) | Résultat |
|---|---|---|---|---|---|---|---|
| Gh07 | 12 | 24 | 12 | 12 | 12 | 9 | G.B |
| Gh08 | 11 | 18 | 12 | 11 | 12 | 6 | G.B |
| Gh09 | 7 | 14 | 9 | 10 | 8 | 5 | G.B |
| Gh10 | 15 | 25 | 15 | 14 | 13 | 5 | G.B |
| Gh11 | 12 | 22 | 12 | 10 | 12 | 7 | G.B |
| Gh12 | 12 | 23 | 10 | 9 | 11 | 6 | G.B |
| Gh13 | 12 | 24 | 12 | 12 | 12 | 8 | G.B |
| Gh14 | 11 | 18 | 12 | 11 | 12 | 6 | G.B |
| GH14SP | 13 | 19 | 15 | 14 | 13 | 5 | G.B |
| Gh15 | 7 | 14 | 9 | 10 | 8 | 5 | G.B |
| Gh16 | 4 | 5 | 6 | 3 | 6 | 3 | C.G = G.B |
| Gh17 | 3 | 2 | 4 | 0 | 4 | 0 | C.G = G.B |
| Gh18 | 7 | 10 | 8 | 4 | 8 | 4 | C.G = G.B |
| Gh50 | 79 | 104 | 66 | 14 | 68 | 14 | C.G = G.B |
| Gh52 | 93 | 121 | 80 | 17 | 82 | 17 | G.B |
| Gh53 | 100 | 141 | 95 | 38 | 95 | 35 | G.B |
| Gh54 | 100 | 154 | 105 | 61 | 107 | 59 | G.B |

**Figure 5.36** Le rapport de comparaison

## 5. Conclusion

Dans ce chapitre nous avons présenté notre étude sur l'algorithme de Cohen et Sadeh, pour améliorer cet algorithme nous avons fait une implémentation avec le langage C# en traitant 100 graphes. Le résultat de la comparaison est affiché. On voit bien que notre algorithme est très efficace dans la minimisation des tâches fictives.

# CONCLUSION GENERALE

Dans ce mémoire, nous avons étudié les méthodes de modélisation de l'ordonnancement de projet par le biais du diagramme de Gantt, la méthode AoN et la méthode AoA,

En guise de conclusion, nous pouvons dire que les praticiens préfèrent utiliser la méthode AoA malgré que le graphe de la méthode AoN soit unique et facile à dessiner alors que les graphes AoA soient infinis, difficiles à dessiner et la majorité des praticiens n'arrivent pas à les dessiner correctement. Cependant, la présence des activités fictives les rend encore plus difficiles à être abordées surtout lorsque leur nombre est très élevé avec des réseaux de tailles très grandes.

Le problème de réduction des tâches fictives dans le graphe AoA nous a ouvert les portes pour chercher une solution radicale au problème appelé dans la littérature « le graphe AoA minimal ». Le contexte est devenu plus clair et plus général tandis que l'objectif devient plus précis mais plus complexe.

L'objectif du travail est donc de faire une étude sur l'algorithme de Cohen et Sadeh qui permet une traduction systématique à partir d'un AON vers un réseau AOA, mais ne minimise pas le nombre d'arcs fictifs.

Cet algorithme permet de réduire un certain nombre d'arcs fictifs mais ne permet pas de le minimiser. Nous avons travaillé sur ce point pour donner le graphe AoA minimal avec le respect total des contraintes dans la table d'ordonnancement.

Cette étude montre la supériorité à notre amélioration de l'algorithme dans la majorité des cas.

Le problème de réduction des tâches est vaste, il a de future perspective comme une autre optimisation de l'algorithme de cohen pour retourné des résultats exacte au minimiser le nombre des arcs fictifs et amélioré son efficacité.

# BIBIOGRAPHIE

## Un ouvrage

**[01]** C.P.Vincent, " Heuristique - Création, intuition, créativité et stratégies d'innovation", BOD- Books on Demande France, 2012.

**[02]** C# / .NET support de cours ISTIA 2012-2013.

**[03]** F. ECOTO, Initiation à la recherche opérationnelle, ELLIPES, Paris, France, 2011.

**[04]** G. FINK, Recherche opérationnelle et réseaux, Lavoisier, Paris, 2002.

**[05]** J. Alliot, Nicolas Durand, «Algorithmes génétiques» March 14, 2005.

**[06]** J.CARLIER & P. CHRETIENNE, Problèmes d'ordonnancement, modélisation, complexité, algorithmes, MASSON, ISBN 2-225-81275-6.

**[07]** R.DIESTEL, Graph theory, Springer-Verlag, Heidelberg Graduate Texts in Mathematics Volume 173. ISBN 978-3-642-14278-9, Springer-Verlag, 2000.

## Un article

**[08]** A. BENHOCINE, Théorie des graphes et applications, Cours de post-graduation, Université de Sétif, 2001.

**[09]** A.Wahiba, Le problème conjoint de l'ordonnancement de la production et de planification de la maintenance, Boumerdes, 2014.

**[10]** R. CASSAGNE, M. GOURGAND, & S. RODIER, Un outil d'aide au dimensionnement, à la planification et à la visualisation d'un programme opératoire. In Gestion et Ingénierie des Systèmes Hospitaliers *(GISEH)* : 807-814. Lausanne, Suisse, 4-6 septembre 2008.

**[11]** R.Christophe et T. Denis, Théorie de la Complexité, Notes de cours, ENSGI – INP Grenoble. 2002.

**[12]** Y. Cohen, 2003. A New Approach for Automated Construction of Activities on Arcs (AOA) Networks, Research report - July 2003-1, Deposited at the library, The Open University of Israel.

**[13]** Yuval Cohen, Arik Sadeh. A New Approach for Constructing and Generating AOA Networks – Journal of computer science (Volume 1, Issue 1, 2007).

**[14]** X. KONG, J. SUN & W. XU, Permutation-based particle swarm algorithm for tasks scheduling in heterogeneous sysWtems with communication delays. International Journal of Computational Intelligence Research, 4(1):61.70, 2008.

## Un mémoire ou une thèse

**[15]** A. LAYEB, Utilisation des Approches d'Optimisation Combinatoire pour La Vérification des Applications Temps Réel. Thèse de Doctorat, Université Mentouri de Constantine 2010.

**[16]** H. HOUARI, Planification et Ordonnancement en temps réel d'un Job shop en utilisant l'Intelligence Artificielle, Thèse de Magister, Tlemcen, 2012.

**[17]** L. Zaourar. Recherche opérationnelle et optimisation pour la conception testable de circuits intégrés complexes. Thèse de doctorat, Grenoble, 2010.

**[18]** N. E. MOUHOUB, Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet, Thèse de doctorat, Université de Sétif, Algérie, 2011

**[19]** R. Pascal, Algorithmes Génétiques Hybrides en Optimisation Combinatoire, Thèse Ph.D, Ecole Normale Supérieure de Lyon, 1999.

**[20]** T. Mehenni, utilisation des métaheuristiques pour résoudre un problème d'ordonnancement sur machine a contrainte de ressource non renouvelable, Thèse de Magister, M'sila, 2006.

## Un site web

**[21]** Bruno bachelet : http://www.bruno.bachelet.net/, consulter le 19/03/2017.

**[22]** Eduscol Education http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/3305/3305-parcours-sin-7-ressource-demarche-projet.doc.

**[23]** Geek Directeur Technique : http://www.geek-directeur-technique.com/2009/07/10/le-triangle-qualite-cout-delai.

**[24]** GPP: http://gpp.oiq.qc.ca/le_cycle_de_vie_d_un_projet.htm.

**[25]** Logistique Conseil : http://www logistiqueconseil.org, consulté le : 19/03/2017.

**[26]** Microsoft **:** https://msdn.microsoft.com/fr-fr/library/hh156499.aspx consulté le 29/04/2017

## ملخص

بعد دراسة نماذج جدولة المشاريع من خلال مخطط غانت وAON و AOA نجد أن مسيري المشاريع يفضلون العمل بطريقة AOAعلى الرغم من صعوبة تحقيقها.في حين ان طريقة AON سهلة و أكثر بساطة . و أعقب عرض هذه الطريقة مظاهر تثبت أنها توفر الحد الأدنى من الرسم البياني AOA من حيث وظائف وهمية .أن مشكلة الحد من الوظائف الوهمية في الرسم البيانيAOA للحد الأدنى " الهدف من عملنا تقديم و تحسين الخوارزمية المشهورة لكوان و تشير النتائج في النهاية إلى أفضلية و تفوق تحسيناتنا بنضرة مضمونة.



**الكلمات المفتاحية:** نماذج جدولة المشاريع , وظائف وهمية , رسم بياني , مخطط غانت ,طريقة AOA و AON ,خوارزمية كوان .



## Abstract

After the study of the scheduling modification of project by bias of Gantt diagram, the AoN method and AoA, are conclude that managers of project prefer work by AoA graph despite its difficult of relation, so the potential of graph offer a lot of simplicity. Presentation of this method has been followed by demonstration prove offered by AoA minimal with terms of factitive task .the problem of reduction of fictitious tacks in AoA graph we have open the door for search radical solution for the problem named in the literature “minimum AoA graph “this work present and optimize an Algorithm of Cohen & Sadeh . Results of this work show the superiority of our optimization in efficacy point of view.



**Key words:** Scheduling modification, Gantt diagram, AoN and AoA method, factitive task, Algorithm of Cohen & Sadeh



## Résumé

Après avoir étudié la modélisation de l‘ordonnancement de projet par le biais du diagramme de Gantt, la méthode AoN et la méthode AoA, on conclue que les managers de projet préfèrent travailler avec le graphe AoA malgré qu‘il est difficile à réaliser, alors que le graphe des potentiels offre plus de simplicité. La présentation de cette méthode a été suivie d‘une démonstration prouvant qu‘elle offre le graphe AoA minimal en termes de tâches fictives. Le problème de réduction des tâches fictives dans le graphe AoA nous a ouvert les portes pour chercher une solution radicale au problème appelé dans la littérature « le graphe AoA minimal ».Ce travail présente et améliorer un algorithme « Cohen et Sadeh ». Les résultats de ce travail montrent la supériorité de notre amélioration du point de vue efficacité.



**Mots clés :** La modélisation de l‘ordonnancement, Taches fictives, Diagramme de Gantt, Méthode AOA, Méthode AON, Algorithme Cohen & Sadeh