INTRODUCTION

L'ordonnancement de projet vise à organiser temporellement la réalisation des différentes tâches d'un projet, en tenant compte des contraintes au niveau des ressources, en respectant les liens de précédence et finalement en tenant compte du temps requis pour chaque tâche.

Le projet est donc un ensemble de tâches qu'il faudra organiser dans le temps. L'ordonnancement permettra d'optimiser l'organisation du projet pour le rendre plus performant en terme de temps et de coûts et permettra aussi de pouvoir évaluer les ressources nécessaires à l'exécution de ce projet. Dans un deuxième temps l'ordonnancement a pour but d'organiser les activités en lien avec l'extérieur telles que la maintenance, les activités en sous traitance et les approvisionnements avec les différents fournisseurs.

Les tâches sont définies par leur durée respective, leurs besoins en ressources (humaines, matières premières, matériels ...) et par leurs relations de précédence – c'est à dire les relations temporelles qu'une activité possède avec les activités qui lui sont directement successeurs ou prédécesseurs.

Malheureusement, et quelle que soit la nature du projet auquel on a affaire, il y a souvent de l'incertitude. Les sources d'incertitudes sont nombreuses et variées. Les ressources peuvent venir à manquer, le matériel peut être livré hors délais, et les tâches peuvent prendre plus ou moins de temps que prévu.

Ainsi, lors de la définition de la durée des tâches, le gestionnaire est régulièrement confronté à des éléments d'incertitude car il n'est pas sûr de l'avenir par exemple, ou à des sources d'imperfection de l'information. Conséquemment, les mesures des durées des tâches ne sont pas précises et on travaille alors avec des estimations. Le gestionnaire de projet devra

alors opter pour une modélisation qui permettra de représenter cette incertitude et il y aura toujours un compromis à faire entre compréhensibilité et représentation de la réalité.

L'origine de l'imperfection de l'information peut avoir plusieurs sources et on considère généralement (Bouchon-Meunier, 1995) les 3 sources suivantes:

- L'incertitude: l'événement s'est-il produit, ou va-t-il se produire ?
- L'imprécision: la mesure est réalisée mais on n'est pas sûr de sa validité, ou il n'y a pas de mesure effectuée mais on se base sur une estimation.
- L'incomplétude: il y a des informations qui ne nous sont pas accessibles.

Ces différents types d'imperfection peuvent, bien entendu, être combinés et on peut, par exemple, avoir à faire une estimation (imprécision) alors que nous sommes déjà en manque d'information (incomplétude).

Quelque soit le type d'imperfection de l'information auquel le gestionnaire fera face, il faudra utiliser une modélisation qui est réaliste, en ce sens qu'elle devra reposer sur des informations dont le gestionnaire pourra raisonnablement disposer.

Il existe différents types de modélisations de l'imperfection de l'information et les principales sont les suivantes:

- La théorie des probabilités, qui est la méthode la plus ancienne et la plus communément utilisée. Elle est employée lorsque l'on fait face à une incertitude de type aléatoire. Cette méthode nécessite l'utilisation d'une distribution de probabilité et sa difficulté réside dans la définition de la bonne distribution.
- La théorie des ensembles flous. Toutes les imperfections ne sont pas d'origine aléatoire, la théorie des ensembles flous permet de gérer les incertitudes dues à de l'information incomplète ou imprécise. La logique floue repose sur le concept des ensembles flous. Ici, contrairement à la notion d'ensemble strict, il y a une transition entre deux ensembles, on parle de fonction d'appartenance. Cette

modélisation permet de prendre en compte des descriptions non chiffrées et des mesures imprécises.

- La modélisation par intervalles est utilisée dans le cas ou les données sont imprécises mais que l'information est tellement pauvre que l'on ne peut même pas donner de fonction d'appartenance. On ne peut définir qu'une valeur minimale et une valeur maximale pour le paramètre à estimer, ici la durée de la tâche.

- La théorie des possibilités. Cette modélisation permet de gérer des informations imprécises sur lesquelles il peut également y avoir de l'incertitude. Il faut bien faire la distinction avec la théorie des probabilités: ici on cherche juste à définir dans quelle mesure un évènement est possible et dans quelle mesure il est certain. Sans disposer pour autant de la distribution de probabilité de l'évènement.

- La fonction de croyance, qui est une modélisation plus polyvalente que la théorie des possibilités et celle des probabilités. Ici on affecte une masse de croyance à des éléments d'un ensemble d'événements possibles.

Il y a donc plusieurs modélisations possibles, et il importe alors de trouver celle qui correspond le mieux au type d'imperfection auquel on fait face (Ben Amor et Martel, 2004).

L'ensemble de ces modélisations conduit à un ensemble de méthodes adaptées pour résoudre les problèmes d'ordonnancement sous incertitude. Le chapitre suivant sera donc dédié à un exposé de ces différentes méthodes existantes.

CHAPITRE I

LES DIFFÉRENTES APPROCHES D'ORDONNANCEMENT SOUS INCERTITUDE

Nous pouvons distinguer dans la littérature un certain nombre de méthodes différentes pour gérer l'incertitude dans l'ordonnancement. Elles sont en partie liées aux différentes formes de modélisation de l'imperfection de l'information, également au fait que l'incertitude puisse provenir de différentes sources : de la durée des tâches, de la disponibilité des ressources ou encore du fait qu'une tâche ait besoin d'être exécutée ou non. Les différentes méthodes sont les suivantes (Herroelen et Leus, 2005):

- La méthode réactive
- La méthode PERT
- La méthode stochastique
- La méthode utilisant les nombres flous
- La méthode robuste
- L'analyse de la sensibilité
- La méthode de la chaîne critique
- Le GERT

1.1 La méthode réactive

La méthode réactive n'essaye pas de combattre l'incertitude en créant un plan d'ordonnancement capable d'anticiper d'éventuelles incertitudes, mais revoit et ré optimise l'ordonnancement lors qu'un événement se produit.

Les principaux auteurs sur le sujet sont Sabuncuoglu et Bayiz (2000), Szelke et Kerr (1994) et Vieira et al. (2003).

L'ordonnancement réactif nécessite des règles d'actions pour "réparer" le plan d'ordonnancement. Un exemple type de ces méthodes est la « right shift rule » (règle de mise au plus tard) (Sadeh et al., 1993; Smith, 1994). Cette règle repousse à plus tard les tâches affectées par l'évènement imprévu. Cette méthode ne peut mener qu'a des résultats médiocres, car elle ne revoit pas l'ordonnancement complet des activités.

Une autre approche consiste en un ré ordonnancement complet des activités. Cette technique fait appel à une heuristique spéciale afin que le nouveau plan d'ordonnancement diffère le moins possible du plan original. Il faut donc introduire une stratégie de perturbation minimale. Une telle technique requiert l'utilisation d'algorithmes exacts ou sub-optimaux utilisant des fonctions minimisant la somme des différences des dates de début des activités entre le plan initial et le nouveau plan. (El Sakkout and Wallace, 2000). Alagöz and Azizoglu(2003), Calhoun et al. (2002) proposent un ré ordonnancement des activités en ayant pour but de minimiser le nombre d'activités changées entre le programme initial et le programme final.

1.2 La méthode PERT

PERT est une abréviation anglaise de « Program Evaluation and Review Technique », ce qui signifie programme d'évaluation et d'examen de projet. Dans la méthode PERT les durées des tâches sont le plus souvent représentées par des distributions Bêta.

Une distribution Bêta est définie sur un intervalle fini, donc elle possède une borne inférieure et une borne supérieure. Son mode, c'est-à-dire la valeur maximale de sa distribution peut se situer n'importe où dans cet intervalle.

Ainsi on défini la distribution de la durée des tâches grâce à trois estimations :

- La durée minimale (a)
- La durée la plus probable (m)
- La durée maximale (b)

On obtient les représentations suivantes ;

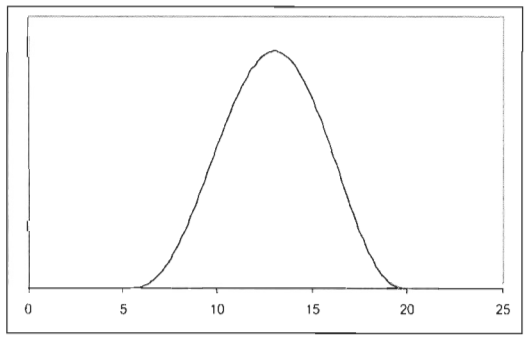

Figure 1.1 Distribution Bêta avec les paramètres a = 5; b = 20; m = 13.

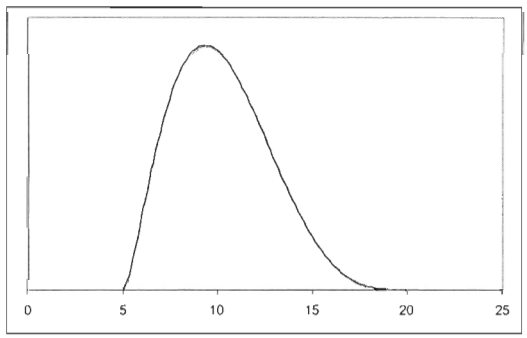

Figure 1.2 Distribution Bêta avec les paramètres a = 5; b = 20; m = 8.

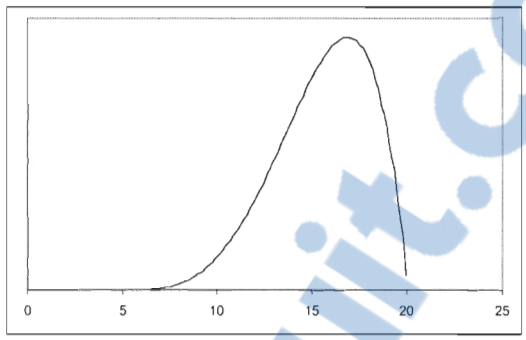

Figure 1.3 Distribution Bêta avec les paramètres a = 5; b = 20; m = 17.

L'analyse de l'ensemble du réseau se fera suivant une loi normale, en effet suivant le théorème central limite, la distribution de la somme de variables indépendantes s'approche de la loi normale à mesure que croit le nombre de variables. Ainsi la durée moyenne du projet sera considérée égale à la somme des durées moyennes des tâches qui composent le chemin critique et la variance égale à la somme des variances de ces mêmes tâches.

1.3 La méthode stochastique

L'ordonnancement stochastique permet de planifier les tâches d'un projet avec des durées incertaines, afin de minimiser la durée du projet attendu sans contrainte de ressources renouvelables et avec des relations de précédence.

La méthode consiste en une génération aléatoire de tous les scénarios possibles en fonction des distributions de probabilités des durées des tâches.

Afin de faire le tri dans l'ensemble des scénarios générés, il faut mettre en place des stratégies d'ordonnancement. L'objectif général consiste à créer une politique d'ordonnancement qui minimise la durée attendue du projet. Fernandez (1995), Fernandez et al. (1996, 1998) et Pet-Edwards et al. (1998) ont modélisé le problème d'optimisation correspondant sous la forme d'une programmation stochastique multi étape.

Radermacher (1985) a défini les « early start policies » (politique du début au plus tôt). Ce concept minimise les solutions interdites, c'est à dire l'exécution de deux tâches sans relation de précédence simultanément alors qu'elles utilisent les mêmes ressources.

Igelmund and Radermacher (1983a,b), ont introduit les politiques pré-sélectives (PRS). Le principe est que pour chaque solution interdite il faut faire en sorte que l'une des activités commence quand l'autre est terminée.

Möhring et Stork (2000) ont introduit une représentation de présélection appelée « waiting conditions » (conditions d'attente). Ces conditions peuvent être modélisées par des conditions de précédences formulées avec des fonctions booléennes OU et ET.

Il faut savoir que les politiques pré-sélectives sont limitées par la capacité de calcul des ordinateurs.

Ainsi Möhring and Stork (2000) ont défini les politiques de pré-sélections linéaires (LIN). Cette méthode vise à planifier dans un premier temps les ensembles d'ordonnancement qui respectent les conditions de précédence originelles.

Au lieu d'utiliser des politiques d'ordonnancement pour trier les différents scénarios générés aléatoirement, on peut faire appel à des méthodes heuristiques. (Pet-Edwards, 1996; Golenko-Ginzburg and Gonik, 1997; Tsai and Gemmil, 1998). Ces techniques sont encore émergentes.

1.4 L'ordonnancement flou

Les partisans de l'ordonnancement flou justifient l'utilisation du nombre flou par le fait que les distributions de probabilités des durées des tâches nécessitent un certain historique.

Comme son nom l'indique cette méthode utilise la théorie des sous-ensembles flous utilisée en intelligence artificielle. C'est à dire des fonctions d'appartenance au lieu de fonctions de distribution.

Les durées des activités sont représentées par un nombre flou, ce qui est très complexe à réaliser. Hapke et al., 1999 a défini une méthode pour définir ce nombre flou.

L'expert n'aura qu'à définir trois intervalles de durée:

- Le premier intervalle pour lequel il est sûr que la durée de l'activité se situera à l'intérieur. On lui affecte une appartenance de 1.
- Un deuxième intervalle où il y a de bonne chance que la durée effective de la tâche en fasse partie. On lui affecte le niveau d'appartenance λ.
- Un dernier intervalle pour lequel il y a peu de chance que la durée de la tache en fasse partie. On lui affecte le niveau d'appartenance ε.

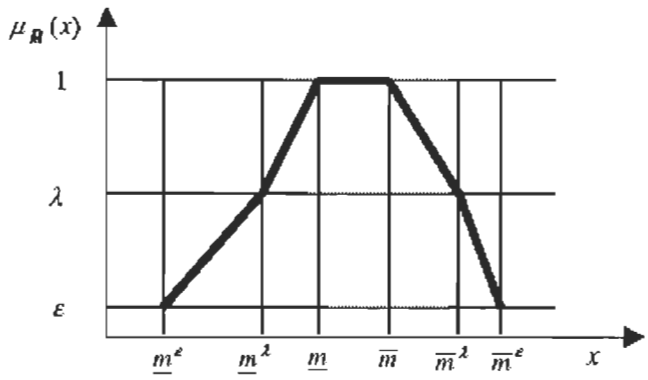

Figure 1.4 Représentation du nombre flou en six points. (Hapke et al. 1999).

L'étude de l'ordonnancement à l'aide de nombre flou a été initiée par Hapke et al. (1994) et Hapke and Slowinski.(1996).

Wang (1999) a développé une approche floue pour planifier les projets de développement de production ayant des durées imprécises. Le projet a une date de début floue, une date de fin floue et des durées de tâches floues, toutes décrites par des représentations floues trapézoïdales.

L'objectif est de déterminer le début de chaque tâche, en tenant compte des dates de début et de fin floues, des durées et des relations de précédence.

Il y a une procédure de recherche de réseaux basée sur la génération de groupes d'activités tels que leurs durées satisfassent les conditions de ressources. Ensuite on génère les dates de début au plus tôt en se basant sur la théorie des possibilités.

Wang (2002) a présenté une procédure de recherche de réseaux qui résolvent le problème d'ordonnancement en ayant pour objectif de minimiser les risques.

1.5 La méthode robuste

La méthode robuste consiste en la création d'un ordonnancement robuste c'est-à-dire qu'il est sensé supporter les imprévus, c'est ce qu'on appelle la tolérance aux fautes. Cette tolérance est générée par des redondances, au niveau des ressources, avec des ressources en stand-by (Ghosh, 1996), au niveau du temps en incluant des tâches "réserves" qui donnent du temps en cas de problème. (Ghosh et Al., 1995).

La pure redondance des ressources est plutôt irréaliste dans un environnement projet, car celles-ci feraient doubler les coûts. Une redondance sur le temps semble plus adéquate, mais dans un environnement multi-projet on est loin d'une étude sur un seul projet.

Gao (1995) a développé la méthode de protection temporelle, basée sur la probabilité non nulle de panne des ressources. La durée des activités est égale à la durée originelle des activités augmentée de la durée probable des pannes sensées se produire durant son exécution. Cette technique est basée sur les statistiques propres aux ressources (mean time failure, mean time to repair) (temps moyen d'apparition des défauts, temps moyen de réparation), ce qui rend cette approche difficilement applicable si la plus part des ressources des tâches sont humaines.

Davenport et al. (2001) proposent une méthode d'amélioration de la protection temporelle, grâce à ses techniques: « time windows slack » (fenêtre de temps creux) et « focused time windows slack » (fenêtre de temps creux limitée). Cette technique n'inclut pas le tampon de temps dans la durée de la tâche. Ainsi le tampon de temps n'est pas utilisé pour une tâche mais pour plusieurs, le tampon de temps est placé aux endroits où il y a le plus de risque.

1.6 L'analyse de la sensibilité

De nombreuses recherches récentes portent sur l'analyse de sensibilité des ordonnancements de machines (Hall et Posner, 2000 a,b). L'analyse de sensibilité cherche à répondre aux questions du type "qu'est ce qui se passe si ...? " ou encore :

- Quelle est la limite si je modifie un paramètre pour que la solution reste optimale?
- Pour un changement de paramètre donné, quel est le nouveau coût optimal?
- Pour un changement de paramètres donnés, quelle est la nouvelle solution optimale?
- Quand est-ce qu'un ordonnancement est optimal?
- Quand est-ce que la valeur de la fonction objective devient optimale?
- Quel type d'analyse de sensibilité est adéquat pour évaluer la robustesse d'un ordonnancement?
- Quel type d'analyse de sensibilité peut être utilisé sans utiliser l'ensemble des détails de la solution?
- Etc...

Ce type d'ordonnancement vise à établir un ordonnancement qui ne craint pas ou très peu les sources d'incertitude.

1.7 La méthode de la chaîne critique

La méthode de la chaîne critique a été développée par Goldratt (1997). C'est une approche 'pratique' de gestion des projets. Cette méthode utilise les mêmes hypothèses que les autres méthodes, c'est-à-dire des tâches avec des durées déterministes, des relations de précédence et des besoins en ressources. Dans cette approche, les ressources ont une disponibilité limitée, ce qui pourrait laisser à penser que la durée du projet sera plus longue qu'avec la méthode du chemin critique.

M. Goldratt propose une approche globale de la gestion des projets reposant sur le constat qu'il est plus important de finir un projet dans les temps que de s'attacher à respecter les dates de fin de tâches intermédiaires.

Cette méthode s'appuie sur plusieurs lois telles que celle de Parkinson qui dit que tout temps prévu pour une tâche sera consommé en entier. Il est donc ainsi proposé de minimiser le temps allouer à chaque tâches. Les durées des tâches sont fixées par rapport aux estimations les plus optimistes et des tampons secours sont placés en fin de projet, servant à protéger non pas la tâche mais le projet dans son ensemble car c'est bien la fin effective du projet qui importe. De plus les chemins critiques seront protégés, en disposant adéquatement des tampons de temps sur les branches. Cette méthode relève de ce que l'on appelle la planification proactive et, même si elle est très récente, certains auteurs ont déjà noté certaines simplifications exagérées dans cette méthode (Herroelen et al.,. 2001).

1.8 La méthode GERT

La méthode GERT (Graphical evaluation and review technique) est une technique qui tient compte des possibilités de l'évolution des enchaînements des tâches, c'est-à-dire que le

réseau peut évoluer. Cette technique permet de gérer la modélisation de tâches qui ne se suivent pas, comme des activités qui peuvent être bouclées (i.e. industrie pharmaceutique), ou permet l'insertion de ramifications conditionnelles, c'est-à-dire que l'enchaînement des activités peut évoluer. On utilise les probabilités pour traiter les liaisons et pour représenter les durées. Ainsi, il est possible que certaines activités ne soient pas exécutées ou même que d'autres le soient plusieurs fois (boucles). Cette méthode a été décrite en 1966 par Pritsker et Happ.

1.9 Conclusion

Le nombre de techniques déployées pour gérer l'incertitude est élevé, pourtant on peut se rendre compte que très peu de ces techniques sont employées. Une étude récente (Pollack, Johnson & Liberatore, 2003), montrait que 70% des utilisateurs de logiciel de planification utilisaient la méthode du chemin critique, et seulement 17% utilisent des logiciels de planification permettant des analyses ou des simulations probabilistes, c'est-à-dire acceptant un indéterminisme des données de base de la planification.

La raison souvent évoquée est que les différentes techniques employées sont trop complexe à mettre en œuvre, ce qui peut dérouter le gestionnaire de projet. De plus les techniques qui utilisent des méthodes probabilistes nécessitent des données statistiques, et il faut donc avoir un certain historique. Les méthodes utilisant les nombres flous ne requièrent pas d'historique, mais nécessitent une certaine connaissance du domaine pour faire les différentes estimations nécessaires. Cette exigence rend difficile leur utilisation pour l'ordonnancement d'un projet innovant pour lequel on ne dispose pas de données statistiques, ni de connaissances accumulées sur d'autres projets permettant d'estimer la durée des tâches.

Compte tenu de ces remarques, il semble intéressant de considérer une modélisation moins 'riche' de l'information concernant la durée des tâches. La modélisation de l'imperfection de l'information choisie dans notre mémoire est celle de l'intervalle de temps et s'applique par exemple à un projet totalement nouveau ou utilisant des technologies encore inutilisées. Ce type de modélisation de l'incertitude est simple et sera facile à exploiter pour

des experts devant définir les bornes de ces intervalles. Les relations de précédence étant fixes et les ressources étant connues et disponibles en quantités suffisantes en tout temps, l'objectif consiste à développer une méthode de planification de projets dont le modèle d'ordonnancement des tâches est basé sur les intervalles de temps des tâches. Le prochain chapitre est consacré à la présentation de cette méthode ainsi qu'à une revue de littérature des différents travaux réalisés à ce jour.

CHAPITRE II

MÉTHODE DES INTERVALLES

Un projet est un ensemble de tâches qui doivent être réalisées dans le respect d'un certain ordre donné par les relations de précédence. Ces relations définissent quelles tâches doivent être complètement terminées avant que celles qui suivent puissent être à leur tour réalisées, on parlera dans se cas de réalisation séquentielle; des tâches pourront être réalisées en même temps, on parlera dans ce cas de réalisation en parallèle. Cet ensemble de relations série et parallèle peut être représenté par un réseau ou les nœuds représentent les tâches et les arcs reliant les différents nœuds représentent les relations de précédence. Cette représentation s'appelle « Project Evaluation and Review Technique » (PERT). Elle a été crée vers la fin des années 50. Quand il n'y a pas d'incertitude quant à la disponibilité des ressources ni d'incertitude concernant la durée des tâches, il est possible d'appliquer la méthode du chemin critique (Critical Path Method CPM). Le but de cette méthode est donc de trouver le chemin critique du réseau, c'est-à-dire l'ensemble des tâches critiques. Une tâche est critique lorsque tout retard pris sur cette tâche retarde d'autant la fin du projet. La méthode CPM permettra donc d'identifier les tâches critiques, mais aussi de calculer les marges des autres tâches, c'est-à-dire la différence entre la date de début au plus tôt de la tâche et la date de début au plus tard de la tâche. La marge représente le temps en plus disponible pour réaliser la tâche, on peut l'utiliser pour commencer la tâche plus tard ou l'utiliser au cas où la tâche pourrait prendre plus de temps. La méthode CPM se déroule en deux temps: premièrement la phase qu'on appellera ascendante, puis deuxièmement la phase descendante. Lors de la phase ascendante on recherche la date de début au plus tôt pour chaque tâche, celle-ci correspond au maximum des dates de début au plus tôt des prédécesseurs auxquelles on ajoute leur durée.

L'opération est effectuée de tâche en tâche dans l'ordre topologique, c'est-à-dire qu'une tâche donnée doit être traitée avant l'un de ses successeurs. Pour pouvoir effectuer ce calcul il faut assigner la date de début au plus tôt de la première tâche à zéro. Puis lors de la phase descendante on calcule la date de début au plus tard, celle-ci correspond au minimum de la date de début au plus tard de l'ensemble des tâches précédentes auxquelles on retire la durée de la tâche en question. Cette opération est effectuée dans l'ordre inverse du tri topologique. Pour pouvoir effectuer ce calcul il faut assigner à la dernière tâche une date de début au plus tard identique à sa date de début au plus tôt. En résumé:

Si d_i représente la durée de la tâche i, t_i représente sa date de début au plus tôt et T_i sa date de début au plus tard, on obtient:

$$t_i = \max_{j \in pred(i)} \left(t_j + d_j \right) \quad \textbf{Équation 2.1}$$

$$T_i = \min_{j \in succ(i)} \left(T_j - d_i \right) \quad \textbf{Équation 2.2}$$

Ainsi la méthode CPM nous permet de définir la date de début au plus tôt, la date de début au plus tard ainsi que la marge de chaque tâche et permet de définir la date de fin au plus tôt du projet.

Malheureusement lorsqu'il y a de l'incertitude sur la durée des tâches, le chemin critique évolue, car en modifiant leur durée, certaines tâches non critiques utilisent toute leur marge devenant ainsi des tâches critiques, ou certaines tâches critiques voyant leur durée diminuer deviennent non critiques. En effet la difficulté de faire de l'ordonnancement sous incertitude est que le chemin critique évolue. Hors les tâches critiques sont les tâches auxquelles il faut apporter le plus d'attention car c'est elles qui peuvent faire prendre du retard au projet. Ainsi la difficulté résidera dans la recherche du chemin critique.

2.1 La méthode des intervalles

Dans cette méthode nous avons choisi de traiter de l'imperfection de l'information concernant la durée des tâches. Les ressources seront considérées sans incertitude, c'est-à-dire disponible tout le temps et en quantité voulue, les relations de précédence elles aussi seront considérées fixes, c'est-à-dire qu'une tâche ne changera pas de prédécesseur(s) ni de successeur(s) au cours de la réalisation du projet.

La modélisation de l'imperfection de l'information choisie pour cette méthode est celle de l'intervalle, modèle utilisé lorsque l'information est incomplète ou imprécise, comme par exemple lors du lancement d'un projet totalement nouveau ou utilisant des technologies encore inutilisées. Ce type de modélisation est simple et sera facile à exploiter pour des experts devant définir les bornes de cet intervalle. Les relations de précédence étant fixes, les ressources étant connues et parfaitement disponibles, il est alors possible la méthode CPM comme mode de représentation du problème. Cependant les durées des tâches étant définies par des intervalles, le chemin critique évolue et il est donc impossible d'appliquer la méthode CPM de façon classique et celle-ci nécessite alors d'être modifiée.

2.1.1 revue de littérature appliquée à la méthode

Ce sont les auteurs travaillant sur l'ordonnancement flou qui ont été les premiers confrontés à ce problème d'adaptation de la méthode du CPM à des situations où les durées des activités sont modélisées sur des intervalles. En effet, lors de la réalisation de la phase descendante de la méthode CPM en nombres flous, Dubois, Fargier et Galvagnon (2003) furent confrontés à des problèmes pour le calcul de la date de début au plus tard et de la marge. Le problème vient du fait que lorsque l'on commence la phase descendante on égalise la date de début au plus tard et la date de début au plus tôt de la dernière tâche. Ils ont donc eu l'idée pour simplifier leur problème de faire les calculs d'abord avec des intervalles plutôt

qu'avec des nombres flous (Chanas, Dubois et Zieliñski, 2002), c'est-à-dire en affectant à chaque tâche une durée minimale et une durée maximale.

La première constatation fut que les différentes valeurs recherchées, c'est-à-dire la date de début au plus tôt, la date de début au plus tard et la marge, pour une tâche donnée, se trouvent dans les configurations extrêmes. Ainsi pour atteindre les valeurs extrêmes des intervalles des différentes valeurs cherchées il faudra assigner aux différentes tâches du réseau: soit leur durée maximale, soit leur durée minimale, ce qui donne un nombre de configuration possible à $2^{\text{nombre de tâches}}$, c'est-à-dire que le problème est « NP-hard ». Le calcul de la date de début au plus tôt a l'aide d'intervalles ne pose pas de problèmes particuliers et ce calcul est de complexité linéaire. Il suffit d'assigner la durée minimale à toutes les tâches précèdent une tâche donnée pour obtenir sa date de début au plus tôt minimale, et inversement les durées maximales pour obtenir la date de début au plus tard maximale. Par contre, calculer les marges des tâches et définir leur criticité est chose bien moins facile, et calculer la marge demeure un problème NP-dur (Chanas et Zelinski, 2003). Récemment des algorithmes polynomiaux capable de trouver des dates de début au plus tôt ont été mis au point (Zelinski, 2005). Ce n'est que récemment qu'un groupe d'auteurs (Chanas, Dubois et Zieliñski, 2002) ont développé des configurations, c'est-à-dire l'affectation des durées des différentes tâches du réseau, permettant de trouver les bornes des intervalles des marges. Les configurations définissant les différentes valeurs pour une tâche donnée sont les suivantes:

- Pour le calcul de la date de début au plus tôt minimale:
  - Assigner la durée de l'ensemble des tâches à leurs valeurs minimales
- Pour le calcul de la date de début au plus tôt maximale:
  - Assigner la durée de l'ensemble des tâches à leurs valeurs maximales.
- Pour le calcul de la date de début au plus tard minimale:
  - Assigner la durée de toutes tâches à leur durée minimale sauf sur un chemin allant de la tâche étudiée jusqu'à la fin du réseau où là la durée assignée sera maximale.

- Pour le calcul de la date de début au plus tard maximale:
    - Assigner la durée des tâches à leurs valeurs minimales sauf sur un chemin allant du début du réseau jusqu'à la fin.
- Calcul de la marge minimale:
    - Assigner la durée de toutes les tâches à leurs valeurs minimales sauf sur un chemin allant du début à la fin du réseau et passant par la tâche étudiée.
- Calcul de la marge maximale
    - Assigner la durée des tâches à leurs valeurs minimales sauf sur un chemin allant du début à la fin du réseau.

L'ensemble de ces résultats permet donc de calculer les différents intervalles des valeurs recherchées avec des algorithmes polynomiaux. En effet, il suffit maintenant d'énumérer les différents chemins possibles afin de trouver les bornes des différentes valeurs cherchées.

Les calculs sont donc possibles quelque soit la taille du réseau. Néanmoins les informations que l'on tire de ces calculs ne sont pas forcément très significatives pour un gestionnaire de projet. En effet les informations fournies par ces calculs ne sont que des intervalles. Ensuite, au niveau de la criticité des tâches, il n'y a que peu d'information: soit elle seront toujours critiques, c'est-à-dire marge minimale égale à la marge maximale égale à zéro, soit elles ne sont jamais critiques, c'est-à-dire marge minimale différente de zéro et finalement possiblement critique, c'est-à-dire parfois critique parfois non suivant les configurations. Le problème principal reste toujours le même: définir la criticité des tâches exactement comme dans une méthode CPM "classique". La notion de criticité est indispensable au gestionnaire de projet pour savoir où porter le plus d'attention et savoir où se trouvent les dangers en cas de prise de retard. Il faut donc essayer de traiter le problème différemment afin d'essayer de fournir un peu plus d'information au gestionnaire de projet.

2.2 Traitement du problème

Pour traiter ce problème, nous avons vu qu'il fallait travailler avec des configurations de durées, c'est-à-dire qu'il faut affecter aux tâches une durée avant d'utiliser la méthode CPM. Ensuite nous avons vu qu'il fallait travailler avec les valeurs extrêmes des intervalles si nous voulions trouver les bornes des valeurs recherchées et finalement qu'il existe des configurations pour lesquelles on est sûr d'obtenir les valeurs cherchées. Nous allons donc utiliser ces différentes constatations dans les trois approches avec lesquelles nous allons traiter le problème, les trois approches choisies sont:

- Une étude des scénarios, c'est-à-dire l'étude de toutes les configurations possibles du problème.
- Une mise en place de la méthode Monte-Carlo appliquée au problème.
- Une mise en place d'un algorithme de fourmis pour le calcul du réseau en utilisant les configurations décrites ci-dessus.

2.2.1 L'étude des scénarios

L'étude de tous les scénarios ne sera pas "gratuite", en effet le nombre de scénarios croît rapidement avec le nombre de tâches du réseau. Cependant le but de l'étude des scénarios a son importance; il s'agit de retirer le maximum d'information possible afin de guider au mieux le gestionnaire de projet. Les informations que nous souhaitons obtenir de cette méthode sont:

- Les bornes de l'intervalle de la date de début au plus tôt de chaque tâche
- Les bornes des intervalles des dates de début au plus tard de chaque tâche
- Les bornes des intervalles marges de chaque tâche

- La criticité relative des différentes tâches
- La répartition des différentes durées possibles du projet.

Ainsi grâce à ces différentes informations le gestionnaire de projet aura plus de facilité à identifier les tâches auxquelles il doit porter plus d'attention (criticité des tâches), puis saura déterminer avec plus de facilité la durée probable du projet grâce à la répartition des durées possibles du projet.

Dans cette approche par scénarios, on s'aperçoit rapidement que le calcul des scénarios atteindra vite ses limites car le nombre de scénarios est égal à $2^{\text{nombre de tâches}}$ ce qui conduit vite à des nombres qui ne sont même pas manipulables par des ordinateurs. En effet, même les ordinateurs sont rapidement dépassés par le nombre de scénarios possibles alors que les temps de calculs croissent exponentiellement (tableau 2.1).

Tableau 2.1 Temps de calcul par la méthode d'étude des scénarios

nombre de tâches	Nombre de scénarios	temps de calcul (ms)	temps de calcul en heures	temps de calcul en années
1	2	1,00E-03	2,79E-10	3,18E-14
2	4	4,01E-03	1,11E-09	1,27E-13
3	8	1,20E-02	3,34E-09	3,82E-13
4	16	3,21E-02	8,92E-09	1,02E-12
5	32	8,03E-02	2,23E-08	2,54E-12
6	64	1,93E-01	5,35E-08	6,11E-12
7	128	4,49E-01	1,25E-07	1,43E-11
8	256	1,03E+00	2,85E-07	3,26E-11
9	512	2,31E+00	6,42E-07	7,33E-11
10	1 024	5,14E+00	1,43E-06	1,63E-10
20	1 048 576	1,05E+04	2,92E-03	3,34E-07
30	1 073 741 824	1,62E+07	4,49E+00	5,12E-04
31	2 147 483 648	3,34E+07	9,28E+00	1,06E-03
40	1,09951E+12	2,21E+10	6,13E+03	6,99E-01
50	1,1259E+15	2,82E+13	7,84E+06	8,95E+02
60	1,15292E+18	3,47E+16	9,64E+09	1,10E+06
70	1,18059E+21	4,15E+19	1,15E+13	1,31E+09
80	1,20893E+24	4,85E+22	1,35E+16	1,54E+12
90	1,23794E+27	5,59E+25	1,55E+19	1,77E+15
100	1,26765E+30	6,36E+28	1,77E+22	2,02E+18

En effet, dès 31 tâches on arrive déjà à plus de 9h de calcul, ce qui limite beaucoup les calculs envisageables, de plus le nombre de scénarios atteint la limite de capacité des ordinateurs. Un ordinateur peut compter avec des entiers sur quatre octets soit 32 bits ce qui correspond à un nombre allant de - 2 147 483 647 à 2 147 483 647. Pour notre étude nous nous limiterons donc à l'étude de réseau n'excédant pas 25 tâches.

Le nombre de scénarios est égal à $2^{\text{nombre de tâches}}$. En effet il y a deux durées possibles par tâches, et on peut alors représenter les différentes configurations par un nombre binaire d'une longueur égale au nombre de tâches. Le zéro représentera le choix de la durée minimale et le 1 représentera le choix de la durée maximale. Ainsi en déroulant la numérotation binaire de zéro jusqu'à $2^{\text{nombre de tâches}}$ on est sûr d'avoir énuméré l'ensemble de tous les scénarios.

L'utilisation de la numérotation binaire semble intéressante pour ne pas oublier de configurations mais pouvoir compter sur un nombre binaire de 30 bits (projet de 30 tâches) n'est pas simple car les modifications de configuration ne sont pas optimales. En effet à chaque fois que l'on passe un nouveau bit, c'est-à-dire lorsque l'on arrive à dénombrer une configuration égale à une puissance de deux, il faut changer toutes les durées des bits inférieurs.

Configuration n° 127: 000.....01111111

Configuration n° 128: 000.....10000000

Les bits de niveau
inférieur changent

Ces changements consomment beaucoup de temps, ainsi pour gagner du temps et pour faciliter le comptage, il faut utiliser un système de comptage différent. L'utilisation du comptage binaire en code Gray semble présenter les avantages recherchés. En effet, lors du comptage en code Gray, il n'y a qu'un seul bit qui change à chaque fois, cette numérotation

utilise l'effet miroir. À chaque fois que l'on arrive à un nombre égal à une puissance de deux, on recopie les nombres précédents précédés d'un 1. (http://fr.wikipedia.org/wiki/Code_Gray)

Tableau 2.2 Comptage en code Gray

rang	code binaire	code Gray
0	0	0
1	1	1
2	10	11
3	11	10
4	100	110

Miroir

Miroir

Le comptage en binaire est donc simple, du moins à la main, mais il va falloir qu'un ordinateur génère lui-même ce code binaire. Il faudra donc utiliser l'algorithme de génération du code Gray. L'algorithme est le suivant:

On passe d'un nombre au suivant en changeant un seul bit:

- Le dernier si le nombre de "1" est pair
- Celui à gauche du "1" le plus à droite sinon

Puis lorsqu'on arrive à une puissance de deux, on change le premier bit.

Nous avons maintenant les capacités de générer les différentes configurations. Il faudra exécuter la méthode CPM pour chaque configuration testée. Les bornes des différentes valeurs seront réévaluées à chaque exécution de la méthode CPM afin de trouver les valeurs extrêmes. À chaque exécution de la méthode nous vérifieront la marge de chaque tâche puis les tâches ayant une marge nulle seront comptées comme critiques. À la fin du calcul nous calculerons le pourcentage de fois où chaque tâche s'est retrouvée critique par rapport au nombre total de configurations. En même temps la durée totale du projet sera notée afin de pouvoir tracer la fonction de répartition des différentes durées possibles du projet.

L'algorithme utilisé par le logiciel peut s'écrire ainsi:

```
Algorithme de calcul des scénarios
Mettre en configuration de base (durée minimale pour toutes les
tâches)
Cpm (exécution de la méthode cpm)
Enregistrer résultats
Compter les tâches critiques
Enregistrer la durée du projet
Rang_configuration = 1
 Répéter
  Incrémenter en code Gray rang_configuration
  Changer la durée à changer (en codage Gray une seule durée change)
  Cpm (exécution de la méthode cpm)
  Si résultats meilleurs: enregistrer résultats
  Compter les tâches critiques
  Enregistrer la durée du projet
 Jusqu'à rang_configuration = 2^nombre de tâches
Fin
```

Figure 2.1 Algorithme de calcul des scénarios (code Gray)

2.2.2 Méthode Monte Carlo

La méthode de Monte Carlo est une méthode qui tire son nom des jeux de hasard pratiqués à Monte Carlo. En effet, la méthode Monte Carlo est une méthode de simulation probabiliste. Lors des simulations, les différents scénarios sont générés de façon aléatoire en respectant les distributions de probabilité des différents éléments fournis. Cette méthode est répandue dans de nombreuses applications et pas seulement l'ordonnancement. Néanmoins parmi les méthodes d'ordonnancement sous incertitude, la méthode Monte Carlo est de loin la plus utilisée.

En ordonnancement sous incertitude avec la méthode Monte Carlo, différents scénarios sont générés de façon aléatoire. Pour cela il faut définir la fonction de distribution des

éléments d'incertitude, tous les éléments incertains se verront affecter une fonction de distribution. Les distributions possibles sont très variées (triangle, discrète, normale,....). Ensuite un tirage aléatoire est fait dans l'intervalle d'incertitude des différentes variables incertaines (coût, durée, ressources,...), le tirage aléatoire respecte la fonction de distribution des variables. À l'issue d'un certain nombre d'itérations, il est possible de faire une étude statistique sur les différents résultats que l'on souhaite connaître lors de la simulation: coûts, durée, date de début au plus tôt, date de début au plus tard, etc. Il est ainsi possible de tracer leurs courbes statistiques respectives. Il faut bien noter que pour deux simulations différentes sur un même problème on obtient deux résultats différents.

Ici la simulation Monte Carlo sera plus simple, les ressources seront considérées parfaitement connues et disponibles, il n'y aura que sur les durées que qu'il y aura de l'incertitude. Les durées étant modélisées par un intervalle les fonctions de distribution associées seront discrètes et de deux points d'égale probabilité de 50 % chacun.

La simulation se déroulera ainsi:

1) Choix pour chaque tâche de façon aléatoire soit de la durée maximale soit de la durée minimale, ce choix sera réalisé de façon équiprobable.

2) Calcul du chemin critique et enregistrement des résultats.

3) Répétition des phases 1 et 2 jusqu'au nombre d'itérations désiré.

L'algorithme peut se traduire ainsi:

```
Algorithme de Monte Carlo
Pour i = 1 à nombre d'itération
 Pour chaque tâche t de 1 à n
  Choisir de façon équiprobable la durée de la tâche
 (minimale/maximale)
 Fin pour
 Cpm (exécution de la méthode cpm)
 Si résultats meilleurs: enregistrer résultats
 Compter les tâches critiques
 Enregistrer la durée du projet
 Fin pour
Fin
```

Figure 2.2 Calcul des scénarios pour la méthode Monte carlo.

2.2.3 Algorithme de fourmi

Cette méthode (Colorni et al. 1994) s'inspire du comportement des colonies de fourmis quand elles vont chercher de la nourriture. Elles sont capables de trouver le plus court chemin entre leur nid et la nourriture. Utilisé pour de nombreuses applications telles que le routage de réseau informatique, cet algorithme pourra être utilisé pour trouver le chemin le plus court de la première à la dernière tâche dans notre problème d'ordonnancement.

2.2.3.1 Fonctionnement

Principe

On se sert ici de la façon dont les fourmis opèrent pour trouver le chemin le plus court entre leur nid et l'endroit où elles vont chercher leur nourriture.

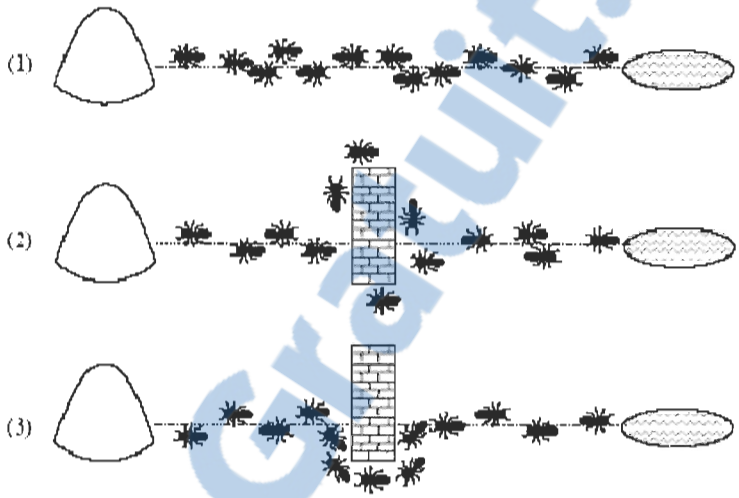

Figure 2.3 Cheminement des fourmis

Schéma 1 On aperçoit que la colonie de fourmis va directement de son nid au tas de nourriture en ligne droite.

Schéma 2 On vient juste de placer un obstacle sur leur chemin, les fourmis vont donc devoir trouver le nouveau chemin le plus court.

Schéma 3 Les fourmis ont fini de trouver le chemin plus court.

Quand elles se déplacent de leur nid vers la nourriture, les fourmis déposent des phéromones qu'elles sont capables de sentir par la suite. Lorsque l'on dépose un obstacle sur leur chemin les fourmis ont le choix de passer soit à gauche soit à droite. On suppose ici qu'il y a autant de chance pour qu'elles choisissent la gauche que la droite, du moins au départ car il n'y a pas de phéromones. Par la suite les fourmis qui seront passées par le chemin le plus

court auront déposées plus de phéromones que les fourmis qui sont passées par le chemin le plus long car elles auront effectué plus de fois le trajet en un même temps, donc au moment de choisir par quel chemin passer (gauche, droite), les fourmis choisiront le chemin qui sent le plus les phéromones, le chemin le plus court. Par analogie, dans notre problème, La nourriture sera remplacée par l'ensemble de toutes les solutions du problème. Le but qui consiste à améliorer la quantité ou la qualité de la nourriture rapportée sera la fonction à optimiser. Les phéromones seront remplacées par une mémoire adaptative.

Application au voyageur de commerce

Le problème du voyageur de commerce est un problème général dans lequel un voyageur doit visiter un ensemble de villes sans en oublier une seule et en passant une seule et unique fois dans chacune d'elles. La difficulté du problème est de définir l'ordre dans lequel le voyageur doit parcourir les villes afin de minimiser la distance parcourue. Initialement utilisé pour résoudre le problème du voyageur de commerce, nous allons voir comment l'algorithme a été implanté. L'analogie est facile à voir ici, comme les fourmis, le voyageur doit effectuer le chemin le plus court. L'algorithme illustré ici est l'algorithme de base « Ant System » (Colorni et al., 1992).

L'algorithme est opéré pour t itérations (*t* de 1 à *tmax (nombre maximal d'itérations)*). À chaque itération les fourmis parcourent le graphe des villes à leur manière. Le parcours de chaque fourmi ce construit ainsi:

1) la fourmi choisit une ville de départ au hasard
2) la fourmi choisit au hasard la ville suivante en fonction de:
    a. la liste des villes non visitées, fourmi k sur ville i: J_i^k
    b. l'inverse de la distance par rapport à la ville choisie, cette quantité est appelée la visibilité, cette grandeur sert à ce qu'une fourmi ne choisisse pas une ville trop éloignée. **Visibilité:** $\eta_{ij} = \frac{1}{d_{ij}}$

c. la quantité de phéromones, paramètre qui évolue au passage des fourmis. **intensité** de la piste: $\tau_{ij}(t)$

La probabilité que la fourmi choisisse la ville est définie ainsi:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{(\tau_{ij}(t))^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l\in j_i^k}(\tau_{il}(t))^{\alpha}.(\eta_{ij})^{\beta}} & si\ j\in j_i^k \\ 0 & si\ j\notin j_i^k\end{cases}$$

Équation 2.3

α et β sont deux paramètres fixant l'importance relative de la visibilité et de l'intensité de la piste.

3) L'étape deux est réitérée jusqu'à atteindre la première ville visitée.
4) On dépose une quantité de phéromones sur l'ensemble des pistes $\Delta\tau_{ij}^{k}(t)$, cette quantité est fonction de la qualité de la solution trouvée par la fourmi:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{q}{L^{k}(t)} & si(i,j)\in T^{k}(t) \\ 0 & si(i,j)\notin T^{k}(t)\end{cases}$$

Équation 2.4

Q est un paramètre fixé, $L^{k}(t)$ représente la longueur du tour effectué par la fourmi k à l'itération t, $T^{k}(t)$ représente le chemin suivi par la fourmi k pendant sont itération t. (notons que cette quantité de phéromones est enregistrée, elle sera déposée une fois que toutes les fourmis de l'itération seront passées).

5) Étape appelée évaporation des pistes, on décrémente la dose de phéromone des pistes, puis on ajoute les quantités de phéromones enregistrée. L'évaporation s'effectue suivant la formule ci-dessous:

$$\tau_{ij}(t+1)=(1-\rho).\tau_{ij}(t)+\Delta\tau_{ij}(t)$$

Équation 2.5

$\tau_{ij}(t)$ Représente la quantité de phéromone sur la liaison de la ville i à la ville j à l'itération t, rhô est le paramètre d'évaporation, $\Delta\tau_{ij}(t)$ représente les quantités de phéromones entre la ville i et j accumulées au cours de l'itération t. L'algorithme AS appliqué au problème du voyageur de commerce peut se représenter ainsi:

(source: Métaheuristiques pour l'optimisation difficile, Dréo, Pétrowski, Siarry et Taillard, 2003)

```
Algorithme de colonie de fourmis de base: le "Ant System".
Pour t = 1, ....., t_max
   Pour (chaque fourmi k = 1,....., m
      Choisir une ville au hasard
      Pour  chaque ville non visité i
        Choisir une ville j, dans la liste J_i^k des villes restantes,
             selon la formule 2.3
      Fin Pour
      Déposer une piste Δτ_ij^k(t) sur le trajet T^k(t) conformément à
        l'équation 2.4
   Fin Pour
   Évaporer les pistes selon l'équation 2.5
Fin Pour
```

Ajustement des paramètres

Les paramètres à ajuster sont:

- Q, pour le dosage des phéromones: il est conseiller de prendre une valeur proche du meilleur chemin possible, il faut donc faire une estimation.

- Le nombre de fourmi: pour mieux exploiter les capacités de cet algorithme, il est conseillé de prendre un nombre de fourmi égale au nombre de ville à parcourir, l'algorithme n'est pas sensible au réglage fin du nombre de fourmi.
- Les paramètres α et β sont à ajuster en fonction du problème, il faudra décider si l'on favorise la visibilité ou l'intensité.

2.2.3.1 Adaptation au problème d'ordonnancement

La formulation du problème d'ordonnancement comme présenté en introduction nous amène à résoudre un problème combinatoire NP-dur, ainsi certains auteurs ont proposé une énumération des différents chemins possibles pour résoudre ce problème. Dans cette partie nous allons résoudre ce problème en appliquant un algorithme de colonie de fourmis.

Nous procéderons en quelque sorte à une énumération des différents chemins, et ce sont les fourmis qui trouveront le meilleur chemin parmi le réseau étudié.

Calcul de la date de début au plus tôt

Pour le calcul de la date de début au plus tôt il n'y a pas besoin de mettre en place une métaheuristique, en effet ce calcul n'est pas NP-dur mais linéaire.

Pour calculer la date de début au plus tôt minimale, il suffit d'affecter à toutes les tâches leur durée minimale et de faire le calcul du chemin critique complet, les dates de début au plus tôt trouvées sont les dates de début au plus tôt minimale pour chaque tâche.

Pour le calcul de la date de début au plus tôt maximale, c'est l'inverse, on affecte la durée maximale à chaque tâche, puis après calcul du chemin critique complet on trouve la date de début au plus tôt maximale pour chaque tâche.

Les algorithmes peuvent se traduire ainsi:

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la date de début au plus tôt minimale
Affecter la durée minimale à toutes les tâches.
Cpm (exécution de la méthode cpm)
Enregistrer résultat (date de début au plus tôt de chaque tâche =
date de début au plus tôt minimale)
Fin
```

Figure 2.4 Algorithme de calcul de la date de début au plus tôt minimale

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la date de début au plus tôt minimale
Affecter la durée maximale à toutes les tâches.
Cpm (exécution de la méthode cpm)
Enregistrer résultat (date de début au plus tôt de chaque tâche =
date de début au plus tôt minimale)
Fin
```

Figure 2.5 Algorithme de calcul de la date de début au plus tôt minimale

Calcul de la date de début au plus tard

Ce calcul est un problème combinatoire NP-dur, si l'on souhaite le traiter en étudiant tous les scénarios. J'ai donc décidé de mettre en place une métaheuristique en tenant compte des hypothèses publiées dans l'article de Dubois, Fargier et Fortin (2005). Les hypothèses sont les suivantes:

- Calcul de la date de début au plus tard minimale

  Dans les conditions suivantes il existe un chemin p qui permet de minimiser la date de début au plus tard.

- Durées de toutes les tâches affectées à la valeur minimale
- Les tâches du chemin p allant de la tâche en question jusqu'à la fin du réseau se voient affecter la durée maximale (tâche étudiée comprise).

➢ Calcul de la date de début au plus tard maximale

Dans les conditions suivantes il existe un chemin p qui permet de maximiser la date de début au plus tard.

- Durées de toutes les tâches affectées à la valeur minimale
- Les tâches du chemin p allant du début du réseau à la fin se voient affecter la durée maximale.

L'objectif est donc de trouver le chemin en question, c'est là que j'utilise l'algorithme des colonies de fourmis.

Calcul de la date de début au plus tard minimale

Pour le calcul de la date de début au plus tard minimale il faut donc trouver le chemin allant de la tâche étudiée jusqu'à la fin du réseau, qui minimise la date de début au plus tôt de la tâche choisie. L'analogie avec le voyageur de commerce est donc envisageable.

L'adaptation se fera comme suit:

L'algorithme est opéré pour t itérations (t de 1 à $tmax$). À chaque itération les fourmis parcourent le graphe des tâches à leur manière. Le parcours de chaque fourmi ce construit ainsi:

1) la fourmi part de la tâche étudiée: i (tâche 0)

2) la fourmi choisi au hasard la tâche suivante (j) en fonction de:
   a. la liste des successeurs: *succ(i)*
   b. la durée de la tâche suivante choisie (j), c'est la **Visibilité:** $D(j)$

c. la quantité de phéromones, paramètre qui évolue au passage des fourmis. C'est l'**intensité** de la liaison. $\tau_{ij}(t)$

La probabilité que la fourmi choisisse la tâche est définie ainsi:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{(\tau_{ij}(t))^{\alpha}.(D(j))^{\beta}}{\sum_{l\in suc(i)}\tau_{il}(t))^{\alpha}.(D(l))^{\beta}} & si\ j\in succ(i)\\ 0 & si\ j\notin succ(i)\end{cases}$$

Équation 2.6

α et β sont deux paramètres fixant l'importance relative de la visibilité et de l'intensité de la liaison.

3) L'étape deux est réitérée jusqu'à atteindre la dernière tâche.

4) On dépose une quantité de phéromones sur l'ensemble des pistes (relation de précédence entre deux tâches) $\Delta\tau_{ij}^{k}(t)$, elle, est fonction de la qualité de la solution trouvée par la fourmi:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q+1}{Lst^{k}(t)+1} & si\ (i,j)\in P^{k}(t)\\ 0 & si\ (i,j)\notin P^{k}(t)\end{cases}$$

Équation 2.7

Le terme $Lst^{k}(t)$ représente la longueur du tour effectué par la fourmi k à l'itération t. Q est un paramètre fixé au départ égale à $Lst^{1}(1)$, la longueur du trajet de la première fourmi. Le terme $P^{k}(t)$ représente le chemin suivi par la fourmi k pendant son itération t (notons que cette quantité est enregistrée, elle est déposée seulement une fois que toutes les fourmis de l'itération sont passées). J'ai rajouté (+ 1) au numérateur et au dénominateur de la formule par rapport à l'algorithme de base, car les premières tâches ont en général une date de

début au plus tard égale à zéro, et un paramètre Q égale à zéro ne permettrait pas de faire de comparaison entre les différentes solutions testées.

5) Étape appelée évaporation des pistes, elle s'effectue à la fin d'une itération quand toutes les fourmis ont terminé. On décrémente la dose de phéromone des liaisons, puis on ajoute les quantités de phéromones enregistrées. L'évaporation s'effectue suivant la formule ci-dessous:

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t)$$

Équation 2.8

Le terme $\tau_{ij}(t)$ représente la quantité de phéromones sur la liaison de la tâche i à la tâche j à l'itération t. ρ est le paramètre d'évaporation et $\Delta\tau_{ij}(t)$ représente les quantités de phéromones entre la ville i et j accumulées au cours de l'itération t.

L'algorithme peut se traduire ainsi:

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la date de début au plus tard minimale
Mettre en configuration de base (durée minimale pour toutes les
tâches sauf celle étudiée)
Cpm (exécution de la méthode cpm)
Q = lst(i)
Enregistrer résultat (date de début au plus tard minimale)
 Pour t = 1…..tmax
  Pour chaque fourmi k = 1, . . . , m
   Mettre en configuration de base
   Tâche actuelle = tâche étudiée
   Répéter
    Choisir une tâche parmi les successeurs de la tâche
    actuelle suivant la formule de probabilité
```

```
    Affecter la durée maximale à la tâche choisie
    Tâche actuelle = tâche choisie
  Jusqu'à tâche choisie = dernière tâche du réseau
  Cpm (exécution de la méthode cpm)
  Déposer phéromones
  Si résultat meilleur enregistrer résultat
 Fin pour
 Évaporer les pistes
Fin pour
```

Figure 2.6 Algorithme de calcul de la date de début au plus tard minimale

Calcul de la date de début au plus tard maximale

Pour le calcul de la date de début au plus tard maximale il faut trouver le chemin allant du début jusqu'à la fin du réseau, qui maximise la date de début au plus tôt de la tâche choisie (cf. calcul de la date de début au plus tard).

L'adaptation de l'algorithme de fourmis se fera comme celle du calcul de la date de début au plus tard minimale, sauf que la fourmi partira de la première tâche du réseau, pour se rendre à la dernière. La quantité de phéromones déposée est également calculée différemment (cf. étape 4 calcul de la date de début au plus tard minimale):

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Lst^{k}(t)}{Q+1} & si\ (i,j)\in P^{k}(t)\\ 0 & si\ (i,j)\notin P^{k}(t)\end{cases}$$

Équation 2.9

On conserve le (+ 1) au dénominateur, car certaines tâches peuvent avoir une date de début au plus tôt égale à zéro.

L'algorithme peut se traduire ainsi:

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la date de début au plus tard maximale
Mettre en configuration de base (durée minimale pour toutes les
tâches)
Cpm (exécution de la méthode cpm)
Q = lst(i)
Enregistrer résultat (date de début au plus tard maxi)
 Pour t = 1…..tmax
  Pour chaque fourmi k = 1, . . . , m
   Mettre en configuration de base
   Tâche actuelle = première tâche
   Répéter
    Choisir une tâche parmi les successeurs de la tâche
    actuelle suivant la formule de probabilité
    Affecter la durée maximale à la tâche choisie
    Tâche actuelle = tâche choisie
   Jusqu'à tâche choisie = dernière tâche du réseau
   Cpm (exécution de la méthode cpm)
   Déposer phéromones
   Si résultat meilleur enregistrer résultat
 Fin pour
 Évaporer les pistes
Fin pour
```

Figure 2.7 Algorithme de calcul de la date de début au plus tard maximale

Calcul de la marge

Ce calcul est également un problème combinatoire NP-dur si on souhaite le traiter en étudiant tous les scénarios. Pour traiter ce problème, nous allons également mettre en place une métaheuristique en tenant compte des hypothèses de Dubois, Fargier et Fortin (2005).

Ces hypothèses sont les suivantes:

- Calcul de la marge minimale

Dans les conditions suivantes il existe un chemin p qui permet de minimiser la marge.

  - Durées de toutes les tâches affectées à la valeur minimale.
  - Les tâches du chemin p allant du début du réseau à la fin du réseau et passant par la tâche étudiée, se voient affecter la durée maximale.

- Calcul de la marge maximale

Dans les conditions suivantes il existe un chemin p qui permet de maximiser la marge.

  - Durées de toutes les tâches affectées à la valeur minimale
  - Les tâches du chemin p allant du début du réseau à la fin se voient affecter la durée maximale.

Calcul de la marge minimale

L'adaptation se fera comme celle du calcul de la date de début au plus tard maximale, c'est-à-dire que la fourmi cherchera un chemin allant du début à la fin et passant par la tâche choisie. La formule de calcul de la dose de phéromones à déposer est également différente, le calcul se fera ainsi:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q+1}{F^{k}(t)+1} & si\,(i,j)\in P^{k}(t)\\ 0 & si\,(i,j)\notin P^{k}(t)\end{cases}$$

Équation 2.10

Le terme $F^{k}(t)$ représente la marge de la tâche étudiée, à l'itération t et pour la fourmi k.

Afin de s'assurer que le chemin p passe bien par la tâche étudiée, la fourmi partira vers le début du réseau dans un premier temps, puis vers la fin du réseau dans un second temps.

L'algorithme peut se traduire ainsi:

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la marge minimale
Mettre en configuration de base (durée minimale pour toutes les
tâches sauf celle étudiée)
Cpm (exécution de la méthode cpm)
Q = lst(i)
Enregistrer résultat (marge minimale)
 Pour t = 1…..tmax
  Pour chaque fourmi k = 1, . . . , m
   Mettre en configuration de base
   Tâche actuelle = tâche étudiée
  Répéter
    Choisir une tâche parmi les prédécesseurs de la tâche
    actuelle suivant la formule de probabilité
    Affecter la durée maximale à la tâche choisie
    Tâche actuelle = tâche choisie
   Jusqu'à tâche choisie = première tâche du réseau
   Tâche actuelle = tâche étudiée
   Répéter
    Choisir une tâche parmi les successeurs de la tâche
    actuelle suivant la formule de probabilité
    Affecter la durée maximale à la tâche choisie
    Tâche actuelle = tâche choisie
   Jusqu'à tâche choisie = dernière tâche du réseau
   Cpm (exécution de la méthode cpm)
   Déposer phéromones
   Si résultat meilleur enregistrer résultat
 Fin pour
 Évaporer les pistes
Fin pour
```

Figure 2.8 Algorithme de calcul de la marge minimale

Calcul de la marge maximale

L'adaptation se fera comme celle du calcul de la date de début au plus tard maximale, c'est-à-dire que la fourmi cherchera un chemin allant du début à la fin du réseau passant ou ne passant pas par la tâche choisie. La formule de calcul de la dose de phéromones à déposer est également différente, le calcul se fera ainsi:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{F^{k}(t)}{Q+1} & si\ (i,j)\in P^{k}(t)\\ 0 & si\ (i,j)\notin P^{k}(t)\end{cases}$$

Équation 2.11

Le terme $F^{k}(t)$ représente la marge de la tâche étudiée, à l'itération t et pour la fourmi k.

L'algorithme peut se traduire ainsi:

```
Algorithme de colonies de fourmis de base: le "Ant System".
Calcul de la marge maximale
Mettre en configuration de base (durée minimale pour toutes les
tâches)
Cpm (exécution de la méthode cpm)
Q = lst(i)
Enregistrer résultat (marge maximale)
 Pour t = 1…..tmax
  Pour chaque fourmi k = 1, . . . , m
   Mettre en configuration de base
   Tâche actuelle = première tâche
   Répéter
```

```
      Choisir une tâche parmi les successeurs de la tâche
      actuelle suivant la formule de probabilité
      Affecter la durée maximale à la tâche choisie
      Tâche actuelle = tâche choisie
    Jusqu'à tâche choisie = dernière tâche du réseau
    Cpm (exécution de la méthode cpm)
    Déposer phéromones
    Si résultat meilleur enregistrer résultat
  Fin pour
  Évaporer les pistes
Fin pour
```

Figure 2.9 Algorithme de calcul de la marge maximale (Ant system)

2.2.4 Conclusion

Nous avons donc mis en évidence trois façons de traiter le problème ainsi formulé. Tel que nous l'avons vu, les trois méthodes nous fourniront des résultats différents en termes de richesse et de précision. Il y aura aussi des différences en termes de temps de calcul et même des limites en termes de taille de réseau, notamment pour la méthode des scénarios. La prochaine partie est dédiée à la comparaison de ces différentes méthodes.

CHAPITRE III

COMPARAISON DES MÉTHODES

Les trois méthodes que nous avons proposé et qui permettent de traiter le problème de planification de projet sous incertitude diffèrent en termes de temps d'exécution et de qualité des résultats. Nous allons donc comparer l'efficacité des différentes méthodes à partir de réseaux de différentes tailles.

Quatre tailles de réseaux différentes seront testées par les différentes méthodes. L'étude des scénarios ne se fera que sur le réseau de petite taille (25 tâches) pour les raisons évoquées précédemment, à savoir l'explosion du temps de traitement pour des réseaux de taille supérieure à une trentaine de tâches.

méthode utilisée / nombre de tâches	Etude des scénarios	simulation Monte Carlo	Algorithme de fourmi
25	oui	Oui	oui
60	non	Oui	oui
90	non	Oui	oui
120	non	Oui	oui

Tableau 3.1 Les configurations testées

L'efficacité des méthodes sera donc testée sur les réseaux du tableau 3.1. La méthode Monte Carlo sera utilisée avec différents réglages, premièrement pour 10 000 itérations, puis 100 000 itérations et finalement pour 1 000 000 itérations et ce pour tous les réseaux. Ils seront répertoriés sous les acronymes MC 10 000, MC 100 000 et MC 1 000 000. La méthode utilisant une métaheuristique de colonie de fourmis sera utilisée avec trois réglages, le premier ne favorisera ni l'intensification, ni la diversification, le second favorisera

l'intensification, le dernier favorisera la diversification. Ils seront répertoriés sous les acronymes F1-1, F2-1, F1-2, les valeurs 1 et 2 correspondent aux coefficients que l'on applique aux variables d'intensification et de diversification soit 1 pour les deux variables, soit 1 pour l'une des variables et 2 pour l'autre variable.

Les réseaux de notre banc d'essais proviennent de la librairie Psplib en ligne sur internet. (http://129.187.106.231/psplib/). Ils sont également détaillés en annexe.

3.1 Résultats

3.1.1 le temps d'exécution

Le temps d'exécution varie suivant la méthode puis suivant la taille du réseau étudié. L'évolution du temps de calculs en fonction de la taille du réseau ne sera pas la même suivant la méthode également.

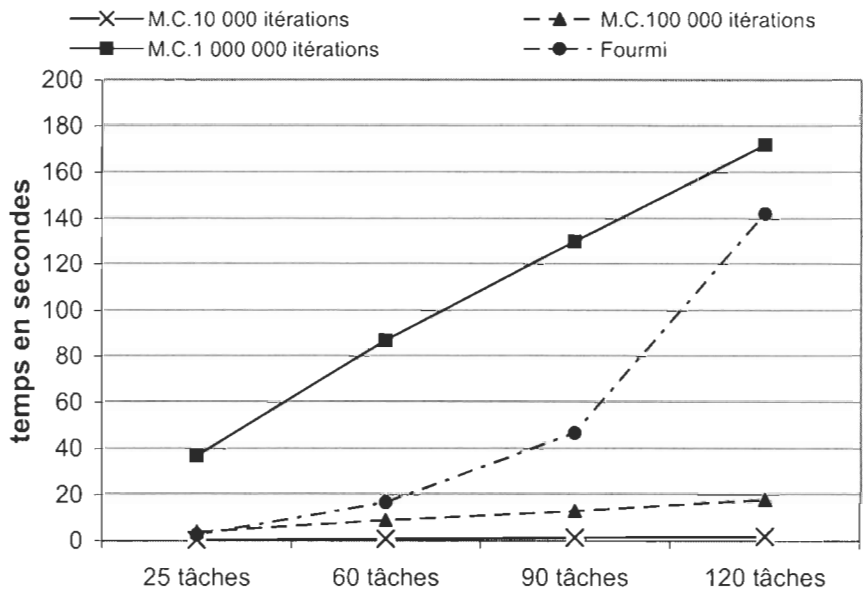

Figure 3.1 Temps de calculs en fonction des méthodes et du nombre de tâches

On constate que le temps d'exécution est croissant en fonction de la taille du réseau étudié. Le temps de calcul évolue de façon linéaire quand il s'agit de la méthode Monte Carlo, la pente de la droite dépend du nombre d'itérations effectuées. Pour ce qui est de la méthode utilisant une métaheuristique de colonie de fourmis, le temps d'exécution évolue parallèlement aux nombres de fourmis ; en effet plus il y a de tâches, plus il y a de fourmi à parcourir le réseau, ce qui explique l'évolution de la courbe.

3.1.2 Précision

Le problème étant NP-dur, lorsque le réseau est de taille importante il devient impossible à calculer de façon exacte. Il y aura donc des différences entre les résultats donnés par chaque méthode. N'ayant pas les résultats exacts pour les réseaux testés de taille supérieure à 25 tâches, la comparaison se fera par rapport au meilleur résultat obtenu par les différentes méthodes.

La précision des différentes méthodes sera comparée par valeur calculée, c'est-à-dire la date de début au plus tôt minimale puis maximale, la marge minimale et maximale. Nous obtenons donc les quatre graphiques suivants. Les valeurs sont exprimées en pourcentage d'erreurs par rapport à la taille du réseau, c'est-à-dire le nombre de valeurs inexactes par rapport au nombre de tâches dans le réseau.

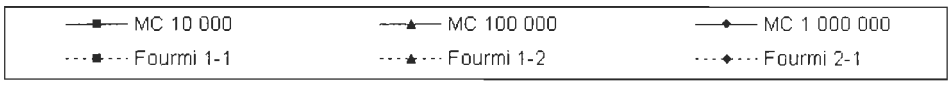

Figure 3.2 Légende des graphiques

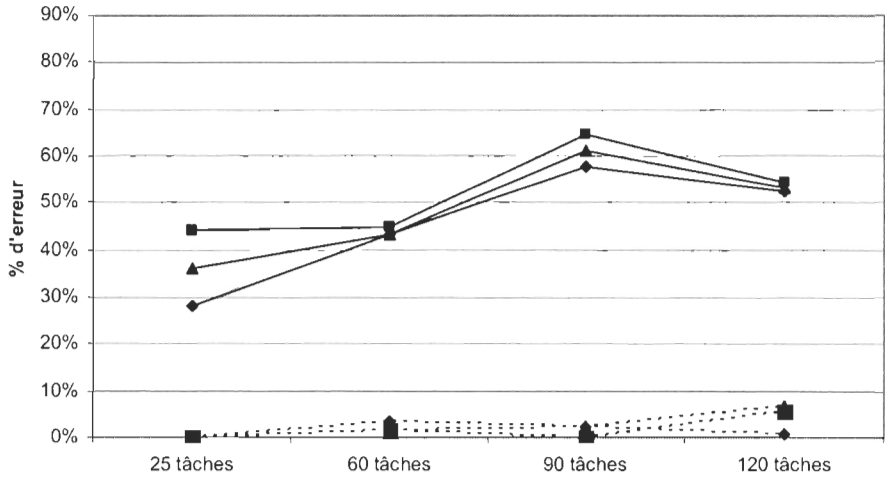

Figure 3.3 Pourcentage d'erreur sur le calcul de la date de début au plus tard minimale

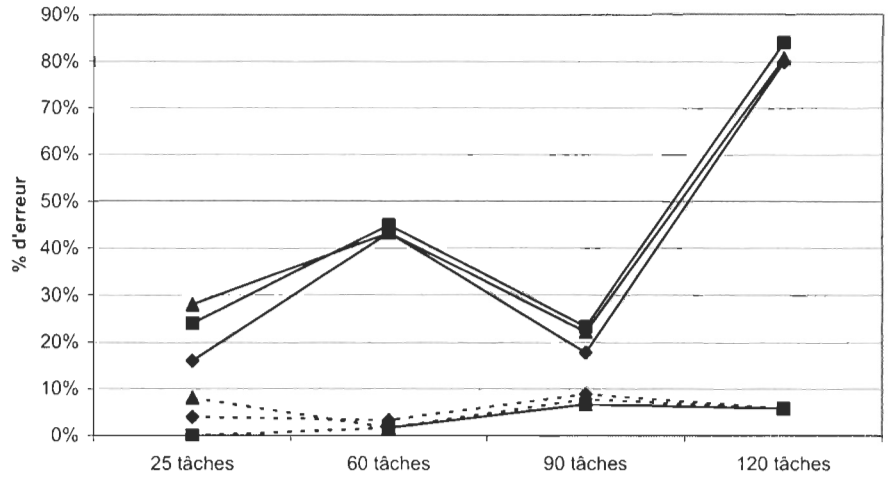

Figure 3.4 Pourcentage d'erreur sur le calcul de la date de début au plus tard maximale

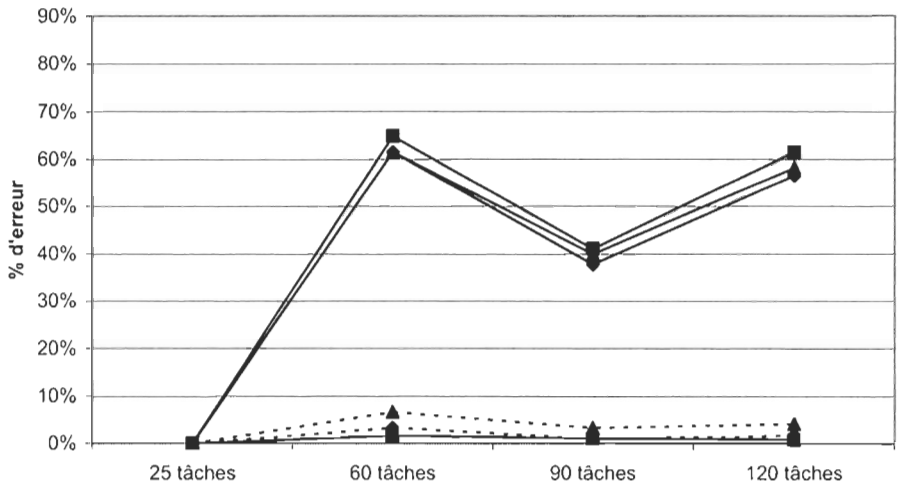

Figure 3.5 Pourcentage d'erreur sur le calcul de la marge minimale

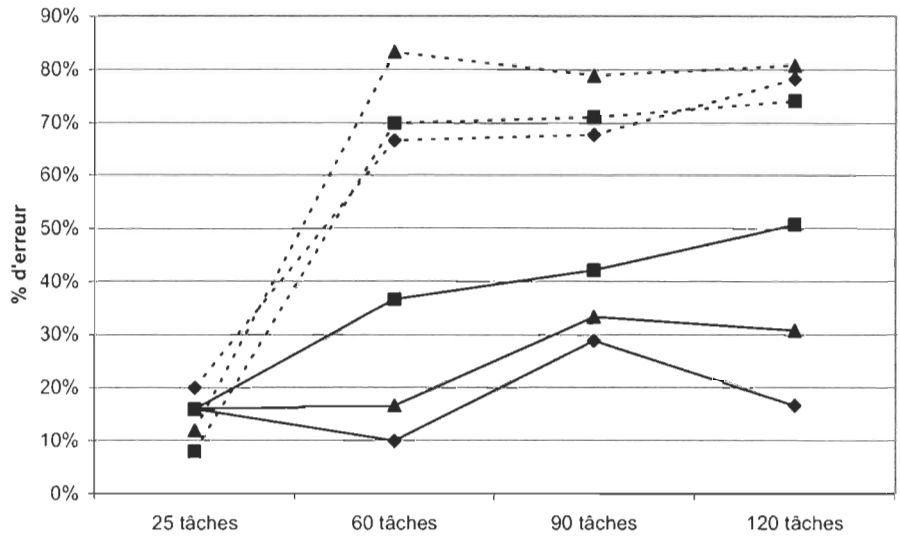

Figure 3.6 Pourcentage d'erreur sur le calcul de la marge maximale

Pour ce qui est des résultats correspondant au calcul de la marge maximale, il s'avère, sans raison apparente, que les résultats obtenus par la méthode "fourmis" sont de moins bonne qualité. Le code de calcul a été vérifié par une tierce personne, et la méthode itérée plusieurs fois sans que la qualité des résultats ait évoluée.

L'évolution du nombre d'itérations de la méthode Monte Carlo a peu d'influence sur la qualité relative des résultats néanmoins la qualité du résultat augmente avec le nombre d'itérations.

Le réglage des paramètres de la métaheuristique de fourmi influe faiblement sur la qualité des résultats obtenus. Il n'y a pas de réglage optimal valable pour tout les cas, mais uniquement un réglage idéal par réseau et par valeur calculée.

N'ayant pas les résultats exacts pour les réseaux de grandes tailles, les résultats seront comparés aux meilleurs résultats obtenus avec les deux méthodes et avec les différents réglages. Les résultats obtenus avec l'algorithme de fourmi sont bons avec un taux d'erreur inférieur à 10%. Par contre pour la méthode Monte Carlo les résultats peuvent atteindre 80 % d'erreur.

On peut également remarquer que la taille du réseau influence la qualité des résultats, mais que la forme même du réseau a également une influence sur la qualité des résultats. Ceci se remarque sur le réseau de 90 tâches qui fournit de meilleurs résultats en pourcentage que le réseau de 60 tâches sur le calcul de début au plus tard maximale et des résultats de moins bonne qualité que le réseau de 120 tâches sur le calcul de la date de début au plus tard minimale.

CONCLUSION

Ce mémoire aborde la problématique de planification sous incertitude dans des situations où l'information disponible est très pauvre. Plus précisément, le problème étudié repose sur une modélisation de la durée des tâches par deux bornes, une borne minimale et une borne maximale. Cette modélisation amène à résoudre un problème nouveau que nous solutionnons par trois méthodes: une méthode basée sur l'énumération de tous les scénarios, la méthode Monte Carlo et finalement l'utilisation d'une métaheuristique basée sur un algorithme d'optimisation de la colonie de fourmis.

La méthode d'énumération des scénarios permet d'obtenir des résultats exacts, mais formulé ainsi le problème est « NP-hard », ce qui limite la taille du réseau étudiable à environ 30 tâches.

La méthode Monte Carlo permet un calcul de réseaux de taille importante mais fournit des résultats de moindre qualité avec un taux d'erreur pouvant aller jusqu'à 80 % en nombre d'erreurs.

La métaheuristique de colonie de fourmis permet d'obtenir un nombre d'erreurs réduit avec un taux d'erreur inférieur à 10 % en nombre. De plus, il est possible d'augmenter le nombre d'itérations afin de réduire encore le nombre d'erreurs. Néanmoins le calcul de la marge maximale génère beaucoup d'erreur. Contrairement au calcul de la date de début au plus tard minimale et de la marge minimale qui se calculent avec des chemins courts, la marge maximale et la date de début au plus tard maximale se calculent sur un chemin allant du début à la fin du réseau, ce qui augmente le nombre de chemins possibles. Cela se voit dans les résultats, avec des meilleurs résultats sur le calcul de la date de début au plus tard minimale et de la marge minimale par rapport au calcul de la date de début au plus tard

maximale et de la marge maximale. Cependant les faibles résultats obtenus sur le calcul de la marge maximale restent sans explication.

Finalement les résultats obtenus par l'utilisation d'une métaheuristique de type colonie de fourmis sont intéressants mais il faut noter que très récemment, et parralèlement à notre recherche, Dubois, Fargier, Fortin et Zieliński (2007) ont proposé une méthode développée qui permettrait de calculer les résultats exacts grâce à une énumération de différents chemins du réseau tout en restant dans un temps polynomial.

L'utilisation d'une métaheuristique de colonie de fourmis est donc, a priori, moins efficace que cette nouvelle méthode mais peut se révéler utile si l'on augmente les contraintes du problème comme par exemple en incluant des contraintes sur les ressources voire même de l'incertitude dans la disponibilité des ressources.

Finalement il serait intéressant d'être plus proche de la réalité. En effet lorsqu'un projet prend du retard le gestionnaire agit afin de palier le retard en effectuant des actions de re-ordonnancement, que ce soit en augmentant les ressources ou en modifiant l'organisation des tâches. Il faudrait pour cela créer des jalons permettant de déclencher une action lors du dépassement de la limite de temps défini. Ce type de simulation a été testé sur la méthode Monte Carlo classique dans l'article de Williams (2004) mais il serait sans nul doute intéressant de tester ce genre de simulation avec la métaheuristique de colonie de fourmis que nous avons développé dans ce mémoire.

RÉFÉRENCES

Alagoz, O., Azizoglu, M., (2003), Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research* 149, 523–532.

Ben Amor, S. et Martel, J.M. (2004), Le choix d'un langage de modélisation des imperfections de l'information en aide à la décision, ASAC 2004, Québec, FSA, Université Laval

Bouchon-Meunier, B.B., (1995), La logique floue et ses applications, Addison-Wesley.

Calhoun, K.M., Deckro, R.F., Moore, J.T., Chrissis, J.W., Van Hove, J.C. (2002), Planning and re-planning in project and production planning. *Omega* 30, 155–170.

Chanas, S., Dubois, D., Zieliński, P., (2002), One the sure Criticality of Task in Activity Networks With Imprecise Durations, *IEE transactions on systems, man, and cybernetics*_part B: *cybernetics*, vol. 32, N°4.

Chanas, S., Zielinski, P. (2003) "On the hardness of evaluating criticality of activities in planar network with duration intervals", *Operation Research Letters*, 31, 53-59.

Colorni, A., Dorigo,M. Maniezzo,V. et Trubian, M. (1994). Ant system for job-shop scheduling, *Belg. J. Oper. Res. Stat. Comput. Sci.*, vol. 34, no.1, pp. 39–53.

Davenport, A.J., Gefflot, C., Beck, J.C., (2001), Slack-based techniques for robust schedules. Paper presented at the Constraints and Uncertainty Workshop, Seventh International Conference on Principles and Practice of Constraint Programming, November 26–December 1, Paphos, Cyprus.

Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2003). Métaheuristiques pour l'optimisation difficile. Éditions Eyrolles.

Dubois, D., Fargier, H. & Galvagnon, V. (2003). On latest starting times and floats in activity networks with ill-know duration. *European Journal of Operational Research*, N° 147, pp. 266-280.

Dubois, D., Fargier, H., Fortin, J., (2005). Computational methods for determining the latest starting times and floats of tasks in interval-valued activity networks. *Journal of Intelligent Manufacturing*, N° 16, pp. 407-421

Dubois, D., Fargier, H., Fortin, J., Zieliński, P. (2007), Criticality analysis of activity--networks under interval uncertainty, Instytut Matematyki i Informatyki PWr., Wrocław, http://www.im.pwr.wroc.pl/~pziel/publications/reports/js.pdf

Dubois, D., Fargier, H., Fortin, J., Zieliński, P., (2005), Interval analysis in scheduling, JProc. Of the 11th Internationl conference on principles and practises of constraint programming (CP' 05), Stiges, Spain, p. 226-240.

El Sakkout, H., Wallace, M., (2000), Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints* 5 (4), 359–388.

Fernandez, A.A., (1995), The optimal solution to the resourceconstrained project scheduling problem with stochastic task durations. Unpublished Doctoral Dissertation. University of Central Florida.

Fernandez, A.A., Armacost, R.L., Pet-Edwards, J., (1996), The role of the non-anticipativity constraint in commercial software for stochastic project scheduling. *Computers and Industrial Engineering* 31, 233–236.

Fernandez, A.A., Armacost, R.L., Pet-Edwards, J., (1998), Understanding simulation solutions to resource-constrained project scheduling problems with stochastic task durations. *Engineering Management Journal* 10, 5–13.

Gao, H. (1995), Building robust schedules using temporal protection—an empirical study of constraint based scheduling under machine failure uncertainty. Master's Thesis. Department of Industrial Engineering, University of Toronto

Ghosh, S. (1996), Guaranteeing fault tolerance through scheduling in real-time systems. Ph.D. thesis. University of Pittsburgh

Ghosh, S., Melhem, R., Mossé, D. (1995), Enhancing Real-Time Schedules to Tolerate Transient Faults, Real-Time Systems Symposium

Goldratt, E.H. (1997). Critical chain, North River Press, Breat Barington, MA.

Golenko-Ginzburg, D. and Gonik, A., (1998). A heuristic for network project scheduling with random activity durations depending on the resource allocation. *International Journal on Production Economics* 55.

Golenko-Ginzburg, D., Gonik, A., (1997), Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics* 48, 29–37.

Hapke, M. and Slowinski, R. (1996), Fuzzy priority heuristics for project scheduling. *Fuzzy Sets and Systems* **83**, pp. 291–299.

Hapke, M., Jaskievicz, A. and Slowinski, R. (1994), Fuzzy project scheduling system for software development. *Fuzzy Sets and Systems* **21**, pp. 101–117

Hapke, M., Jaskievicz, A. and Slowinski, R. (1999), Fuzzy multi-mode resource-constrained project scheduling with multiple objectives. In: Weglarz, J., Editor, 1999. *Project Scheduling—Recent Models, Algorithms and Applications*, Kluwer Academic Publishers, pp. 355–382.

Herroelen, W. and Leus, R. (2001), On the merits and pitfalls of critical chain scheduling, *Journal of Operations Management*, Volume 19-5, pp.559-577.

Herroelen., W. & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, N° 165, pp. 289-306.

Igelmund, G., Radermacher, F.J., (1983a), Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks* 13, 1–28.

Igelmund, G., Radermacher, F.J., (1983b), Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks* 13, 29–48.

Merkle, D, Middendorf, M. & Schmeck, H. (2002). Ant Colony Optimization for Resource-Constrained. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 4, August, pp. 333-346.

Mohring, R.H., Stork, F., (2000), Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research* 52, 501–515.

Pet-Edwards, J., Selim, B., Armacost, R.L., Fernandez, A., (1998), Minimizing risk in stochastic resource-constrained W. Herroelen, R. Leus / *European Journal of Operational Research* 165 (2005) 289–306 305

Radermacher, F.J., (1985), Scheduling of project networks. *Annals of Operations Research* 4, 227–252.

Raz, T., Barnes, R., Dvir, D. (2003), A critical look at critical chain management, *Project Management Journal*, P. 24-32.

Sabuncuoglu, I., Bayiz, M., (2000), Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research* 126, 567–586.

Sadeh, N., Otsuka, S., Schelback, R., (1993), Predictive and reactive scheduling with the microboss production scheduling and control system. In: Proceedings of the IJCAI-93 Workshop on Knowledge-Based Production Planning, Scheduling and Control, pp. 293–306.

Smith, S.S., (1994), Reactive scheduling systems. In: Brown, D.E., Scherer, W.T. (Eds.), Intelligent Scheduling Systems. Kluwer.

Szelke, E., Kerr, R.M., (1994), Knowledge-based reactive scheduling. *Production Planning and Control* 5 (2), 124–145.

Tsai, Y.W. and Gemmil, D.D. (1998), Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research* 111, pp. 129–141

Van de Vonder, S., Demeulemeester, E., Herroelem, W., Leus, R. (2005), The use of buffers in project management: The trade-off between stability and makespan, *Int. J. Production Economics*, N° 87, P. 227-240.

Vieira, G.E., Herrmann, J.W., Lin, E., (2003), Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling* 6, 39–62.

Wang, J.R. (1999), A fuzzy set approach to activity scheduling for product development. *Journal of the Operational Research Society* 50, pp. 1217–1228.

Williams, T. (2004), Why Monte Carlo simulations of project networks can mislead, *Project Management Journal*, pp. 53-61.

Zieliñski, P. (2005). On computing the latest starting times and floats of activities in a network with imprecise durations, cybernetics. *Fuzzy sets and systems*, N°150, pp. 53-76.

APPENDICE A

LES RÉSEAUX

A.1 RÉSEAU DE 25 TÂCHES 55
A.2 RÉSEAU DE 60 TÂCHES 56
A.3 RÉSEAU DE 90 TÂCHES 58
A.4 RÉSEAU DE 120 TÂCHES 61

A.1 Réseau de 25 tâches

numéro de la tache	durée minimale de la tache	durée maximale de la tache	successeurs
0	0	0	1
1	6	7	2;3;4
2	7	9	5;6;11
3	6	7	6
4	2	3	6;7
5	6	8	8
6	7	7	8
7	11	11	9
8	5	6	10
9	4	6	10
10	7	13	12;17
11	17	26	12
12	6	7	13;14;15;16
13	12	19	18
14	25	28	19
15	12	19	20
16	12	18	21
17	12	18	22
18	13	18	23
19	5	9	23
20	13	21	23
21	13	18	24
22	13	18	24
23	5	5	25
24	5	7	25
25	1	2	26
26	0	0	0

A.2 Réseau de 60 tâches

numéro de la tache	durée minimale de la tache	durée maximale de la tache	successeurs
0	0	0	1;2;3
1	8	9	4;9;14
2	1	2	6;13;28
3	10	10	7;11;15
4	6	9	5;21;23
5	5	8	16;37
6	8	14	22
7	9	12	8;19;39
8	1	2	12;34
9	9	15	10;44
10	8	9	25;36;43
11	3	5	20;26
12	6	9	17
13	2	3	17;18;33
14	5	8	24;58
15	1	2	54;57
16	3	4	31;42
17	10	15	27;32
18	9	9	27
19	1	2	26;30
20	3	3	38
21	6	8	30
22	3	4	50
23	3	5	47
24	7	12	39
25	6	11	48;53
26	10	10	29;50
27	9	11	46;60
28	8	13	40;56
29	4	8	55
30	3	5	42
31	3	4	56
32	6	10	38
33	1	1	43
34	9	12	35;38
35	9	14	41
36	1	2	57
37	2	3	49;59
38	4	4	52

39	9	12	52
40	10	19	45
41	8	9	47
42	4	5	50;57
43	3	5	49
44	6	8	52
45	6	9	55
46	7	13	51
47	3	4	54
48	2	3	59
49	10	14	60
50	4	8	59
51	2	2	53
52	1	2	55
53	4	6	54
54	10	14	56
55	8	15	60
56	6	8	58
57	10	10	58
58	3	5	61
59	10	13	61
60	1	2	61
61	0	0	

A.3 Réseau de 90 tâches

numéro de la tache	durée minimale de la tache	durée maximale de la tache	successeurs
0	0	0	1;2;3
1	8	15	10;23
2	10	11	6;7;14
3	1	2	4;5;18
4	2	2	9;29;90
5	10	11	8;21;25
6	8	8	22;26
7	6	8	41
8	9	12	16;24;60
9	1	2	12;51
10	8	11	11;28;44
11	2	3	13;15;17
12	7	8	82
13	2	4	55
14	7	13	19;34;35
15	4	5	37
16	10	17	64
17	9	10	20;46;65
18	9	17	70
19	3	5	27;32
20	7	12	31;56;87
21	2	3	62
22	9	11	33;69
23	6	12	30;46;66
24	3	3	69
25	3	3	36;38;53
26	6	8	48
27	5	7	40;62;76
28	6	12	43
29	8	13	84
30	5	9	81
31	4	6	50;70
32	7	12	51;52;77
33	7	9	72
34	2	3	54
35	8	14	39;42;47
36	4	5	45;60
37	9	10	81
38	5	8	58

39	7	9	62
40	6	9	72;75;85
41	8	16	49
42	8	11	76
43	1	2	57
44	1	1	67
45	8	11	89
46	6	7	53
47	9	15	49
48	1	1	67
49	8	10	86
50	8	12	71
51	10	15	60
52	3	4	80
53	6	10	83
54	3	4	61
55	10	15	59;68;79
56	5	8	78
57	2	3	58;75
58	3	6	84
59	3	5	71;73
60	3	5	73
61	5	8	63
62	9	13	69
63	5	6	74
64	4	4	84
65	8	12	78
66	9	9	85
67	4	5	77
68	2	3	72;82
69	7	8	71
70	2	3	77
71	1	1	80;89
72	5	6	89
73	9	15	78
74	4	4	79
75	2	3	82
76	7	12	85
77	8	12	83
78	8	13	86
79	1	2	81
80	6	10	90
81	10	12	88
82	9	14	87

83	2	4	90
84	7	13	86
85	9	10	87
86	8	15	88
87	1	1	88
88	2	3	91
89	9	15	91
90	3	4	91
91	0	0	

A.4 Réseau de 120 tâches

numéro de la tache	durée minimale de la tache	durée maximale de la tache	successeurs
0	0	0	1;2;3
1	6	8	11;64;74
2	4	6	4;5;9
3	2	2	7;8;12
4	2	4	24;25;98
5	3	4	6;20;37
6	10	16	10
7	2	4	14;22;59
8	7	9	15;58;109
9	4	7	18;52;70
10	6	6	17
11	4	4	13;16;19
12	7	8	32;43;77
13	4	7	31;33;65
14	10	10	21;29;38
15	9	9	27
16	1	1	18;23;50
17	9	12	32;60;72
18	10	17	82
19	5	8	57;75
20	9	14	45;68;105
21	8	14	55;76;96
22	3	5	26;28;34
23	3	4	24
24	7	9	84;100
25	8	10	76
26	10	16	89
27	9	12	36
28	5	6	30
29	2	2	36;41
30	10	11	95
31	6	10	42;47
32	9	17	35;40;44
33	10	18	54
34	6	10	47;63
35	9	17	42
36	10	18	99
37	9	16	46;71;94

38	2	3	39;53
39	8	12	73
40	2	4	115
41	2	2	74
42	2	3	48;76
43	8	15	90
44	2	2	70
45	2	3	56;69
46	1	1	57
47	2	3	77
48	7	9	49;51
49	5	10	109
50	7	10	66;90;97
51	2	3	62
52	9	17	53
53	9	10	87
54	1	2	61;104
55	10	14	58;102
56	10	11	120
57	8	10	86
58	9	16	79;112
59	3	4	70
60	4	8	67;110
61	2	4	66
62	7	12	73
63	10	19	108
64	2	4	92
65	5	6	105
66	5	9	78
67	3	5	92
68	2	2	103
69	6	9	94;97
70	3	3	80
71	7	8	83;85
72	6	12	78;93;114
73	1	1	86;90;97
74	7	12	111
75	5	8	111
76	4	4	85
77	5	5	111
78	6	7	81;112
79	10	12	88
80	1	2	102
81	9	13	101

82	1	2	117
83	8	10	113
84	6	10	107
85	9	12	91
86	2	3	104
87	7	10	89
88	3	4	106
89	7	9	99
90	10	12	101
91	1	1	116
92	5	8	99
93	5	7	117
94	3	5	117
95	1	1	98
96	3	3	106
97	5	10	113
98	5	8	102
99	10	20	118
100	6	7	110
101	5	9	106
102	8	12	104
103	10	11	114
104	10	12	114
105	6	12	108
106	2	3	115
107	2	4	120
108	2	2	118
109	10	17	112
110	1	2	118
111	3	4	115
112	4	5	116
113	5	9	119
114	9	16	119
115	3	4	116
116	1	1	120
117	10	17	119
118	3	3	121
119	8	12	121
120	9	11	121
121	0	0	

APPENDICE B

LES CODES SOURCES

B.1 CODE CALCUL DE LA MÉTHODE CPM 65
B.2 CODE GÉNÉRATION DU CODE GRAY 67
B.3 CODE DE GÉNÉRATION ALÉATOIRE MÉTHODE MONTE CARLO 69
B.4 CODE DE L'ALGORITHME DE FOURMI 70

Les codes ci-dessous sont programmés en langage Basic issu de Visual Basic.

B.1 Code calcul de la méthode CPM

Cette partie du code est utilisée dans les trois logiciels correspondant aux différentes approches de résolution du problème.

```
Private Sub cpm()

Dim i As Integer
Dim Y As Integer
Dim nb_succ As Integer
Dim nb_pred As Integer

nb_cpm = nb_cpm + 1

'date de début au plus tot ========================================
For i = 1 To nombre_de_tache
    t_debut_plus_tot(i) = 0
Next
For i = 2 To nombre_de_tache
    nb_pred = t_reseau(i, 9) 't_reseau est un tableau qui contient
les informations du réseau, la colonne neuf est celle du nombre de
prédecesseur)

     For Y = 1 To nb_pred
        If t_debut_plus_tot(t_reseau(i, 9 + Y) + 1) +
t_duree(t_reseau(i, 9 + Y) + 1) > t_debut_plus_tot(i) Then
            t_debut_plus_tot(i) = t_debut_plus_tot(t_reseau(i, 9 +
Y) + 1) + t_duree(t_reseau(i, 9 + Y) + 1)
        End If
    Next

 Next
```

```
 'date de debut au plus tard=====================================
 t_debut_plus_tard(nombre_de_tache) =
t_debut_plus_tot(nombre_de_tache)

 For i = 1 To nombre_de_tache - 1
    t_debut_plus_tard(i) = 32767
Next
For i = nombre_de_tache - 1 To 1 Step -1
    nb_succ = t_reseau(i, 4)

     For Y = 1 To nb_succ
        If t_debut_plus_tard(t_reseau(i, 4 + Y) + 1) - t_duree(i) <
t_debut_plus_tard(i) Then
            t_debut_plus_tard(i) = t_debut_plus_tard(t_reseau(i, 4 +
Y) + 1) - t_duree(i)
        End If
    Next
 Next
 'la marge ===========================================================
  For i = 1 To nombre_de_tache
  t_marge(i) = t_debut_plus_tard(i) - t_debut_plus_tot(i)
  Next
End Sub
```

B.2 Code génération du code gray

Le code Gray est utilisé dans le calcul des scénarios. Il sert à générer les différentes configuration en ne changeant qu'une seule durée entre chaque configuration.

```
Private Sub code_gray()
Dim rang As Integer
Dim i As Integer
Dim parite As Boolean
'mise à zéro de toutes les variables
redim_tableau_distribution
debut_calcul = Timer
raz

'parité vraie (c'est-à-dire nombre de 1 pair)
parite = True
nb_val = 1
'calcul tte les tache au durée minimale (tous les bits à zéro
configuration 1)
cpm
min_max
'calcul du reste
For rang = 1 To nombre_de_tache - 2 'retrait de la première et la
dernière tâche
Do
    If parite = True Then
        changement (2)'changement du bit de rang 2(on ne compte pas
la première tâche de durée nulle, c'est en fait le premier bit)
        parite = False
    Else
        i = 1
        Do
            i = i + 1
        Loop Until t_verite(i) = True 'recherche du bit à 1 le plus
à droite
```

```
        changement (i + 1)'changement du bit à gauche du 1 le plus à
droite
        parite = True
    End If
    Cpm 'appel de la procedure CPM
    min_max
 Loop Until nb_val = 2 ^ rang
  If rang <> nombre_de_tache - 2 Then
  changement (rang + 1)
 cpm
  parite = Not (parite)

Else
End If

Next
End Sub
```

B.3 Code de génération aléatoire méthode Monte Carlo

Cette partie du code génère les différentes configurations de manière aléatoire.

```
Private Sub code_gray()
nombre_itération = (Text2.Text)'zone d'entrée du nombre d'itération
voulu)
Dim rang As Long
Dim i As Long
Dim y As Integer

'mise à zéro de toutes les variables
redim_tableau_distribution
debut_calcul = Timer
raz

For rang = 1 To nombre_itération
 For i = 2 To nombre_de_tache - 2
  Randomize Timer 'permet un tir vraiment aléatoire
  y = Int(Rnd(Timer) * 2 + 1)'tir aléatoire nombre entre 0 et  2
  If y = 1 Then                          }
   t_duree(i) = t_reseau(i, 2)           }
  Else                                   } application des durées min/max
   t_duree(i) = t_reseau(i, 3)           }
  End If                                 }
 Next
 cpm
Next
End Sub
```

B.4 Code de l'algorithme de fourmi

Sub de calcul des dates de début au plus tard minimales.

```
Private Sub lst_min(num_tache As Integer, nb_gd_succ As Integer)

nb_iteration = CInt(Text9.Text)'zone d'entrée du nombre d'itération
nb_fourmi = nb_gd_succ
Dim i As Integer
Dim j As Integer
Dim f As Integer
Dim derniere_tache As Integer
Dim tache_choisie As Integer
Dim tir_alea As Single
Dim nb_succ As Integer
ReDim t_chemin_fin(nb_gd_succ + 1)
Dim b As Integer
Dim c As Integer
Dim num_succ As Integer
Dim dose_phero As Single
Dim q As Integer
Dim meilleur_res As Integer
Dim probabilite As Single
Dim res_fourmi As Integer
Dim chemin As String
'mettre en configuration de base
For i = 1 To nombre_de_tache
 t_duree(i) = t_reseau(i, 2)'affectation de la durée minimale au
tâches
Next
t_duree(num_tache + 1) = t_reseau(num_tache + 1, 3)'affectation de
la durée maximale à la tâche étudiée.
CPM 'appel de la procédure CPM
```

```
meilleur_res = t_debut_plus_tard(num_tache + 1)
q = t_debut_plus_tard(num_tache + 1)'q paramètre de dosage des
phéromones
'initialiser tableau phéromones
For i = 1 To nombre_de_tache
 For j = 1 To nombre_de_tache
  t_phero_iter(i, j) = 0
  t_phero_somme(i, j) = 0.5
 Next
Next
For i = 1 To nb_iteration
 For f = 1 To nb_fourmi
  t_chemin_fin(1) = num_tache
  derniere_tache = num_tache
  'mettre en configuration de base
   For c = 1 To nombre_de_tache
    t_duree(c) = t_reseau(c, 2)
   Next
  t_duree(num_tache + 1) = t_reseau(num_tache + 1, 3)
  b = 2
  Do
   nb_succ = t_reseau(derniere_tache + 1, 4)
   'choix de la tache
   Do
    Randomize Timer
    num_succ = Int(Rnd(Timer) * nb_succ) + 1
    tir_alea = Rnd(Timer) * 1
    tache_choisie = t_reseau(derniere_tache + 1, 4 + num_succ)
    probabilite = proba_succ(derniere_tache, tache_choisie,
    alpha_lst, beta_lsti)
   Loop Until tir_alea < probabilite
   t_chemin_fin(b) = tache_choisie
   derniere_tache = tache_choisie
   t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 3)
   b = b + 1
```

```
  Loop Until derniere_tache = nombre_de_tache - 1
  CPM
  res_fourmi = t_debut_plus_tard(num_tache + 1)
  c = 1
  chemin = num_tache
  For c = 2 To b - 1
   chemin = chemin & " => " & t_chemin_fin(c)
  Next
  'sauver résultat
  If t_debut_plus_tard(num_tache + 1) < meilleur_res Then
   meilleur_res = t_debut_plus_tard(num_tache + 1)
  End If
  'deposer pheromone
  dose_phero = (q + 1) / (t_debut_plus_tard(num_tache + 1) + 1)

  For c = 1 To b - 2
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) =
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) +
    dose_phero
  Next
 Next
 'evaporer
 For c = 1 To nombre_de_tache
  For b = 1 To nombre_de_tache
   t_phero_somme(c, b) = (1 - rho_lsti) * t_phero_somme(c, b) +
   t_phero_iter(c, b)
  Next
 Next
Next
t_reseau(num_tache + 1, 16) = meilleur_res
End Sub
```

Sub de calcul des dates de début au plus tard maximales

```
Private Sub lst_max(num_tache As Integer, nb_gd_succ As Integer)
nb_iteration = CInt(Text9.Text)
nb_fourmi = nombre_de_tache - 2
Dim i As Integer
Dim j As Integer
Dim f As Integer
Dim derniere_tache As Integer
Dim tache_choisie As Integer
Dim tir_alea As Single
Dim nb_succ As Integer
ReDim t_chemin_fin(nombre_de_tache)
Dim b As Integer
Dim c As Integer
Dim num_succ As Integer
Dim dose_phero As Single
Dim q As Integer
Dim meilleur_res As Integer
Dim probabilite As Single
Dim res_fourmi As Integer
Dim chemin As String
'mettre en configuration de base
For c = 1 To nombre_de_tache
 t_duree(c) = t_reseau(c, 3)
Next
c = 1
Do
 tache_choisie = t_sucesseurs(c)
 t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 2)
 c = c + 1
Loop Until t_sucesseurs(c) = nombre_de_tache + 1

t_duree(num_tache + 1) = t_reseau(num_tache + 1, 2)
```

```
cpm
meilleur_res = t_debut_plus_tard(num_tache + 1)
q = t_debut_plus_tard(num_tache + 1)
'initialiser tableau phero
For i = 1 To nombre_de_tache
 For j = 1 To nombre_de_tache
  t_phero_iter(i, j) = 0
  t_phero_somme(i, j) = 0.5
 Next
Next
For i = 1 To nb_iteration
 For f = 1 To nb_fourmi
  t_chemin_fin(1) = 0
  derniere_tache = 0
  'mettre en configuration de base
  For c = 1 To nombre_de_tache
   t_duree(c) = t_reseau(c, 2)
  Next
  b = 2
  Do
   nb_succ = t_reseau(derniere_tache + 1, 4)
   'choix de la tache
   Do
    Randomize Timer
    num_succ = Int(Rnd(Timer) * nb_succ) + 1
    tir_alea = Rnd(Timer) * 1
    tache_choisie = t_reseau(derniere_tache + 1, 4 + num_succ)
    probabilite = proba_succ(derniere_tache, tache_choisie,
    alpha_lst, beta_lsti)
   Loop Until tir_alea < probabilite
   t_chemin_fin(b) = tache_choisie
   derniere_tache = tache_choisie
   t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 3)
   b = b + 1
  loop Until derniere_tache = nombre_de_tache - 1
```

```
   cpm
   res_fourmi = t_debut_plus_tard(num_tache + 1)
   chemin = "0"
   For c = 2 To b - 1
    chemin = chemin & " => " & t_chemin_fin(c)
   Next
   'sauver résultat
   If t_debut_plus_tard(num_tache + 1) > meilleur_res Then
   meilleur_res = t_debut_plus_tard(num_tache + 1)
   End If
   'deposer pheromone
   dose_phero = (t_debut_plus_tard(num_tache + 1) + 1) / (q + 1)
   For c = 1 To b - 2
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) =
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) +
    dose_phero
   Next
  Next
  'evaporer
  For c = 1 To nombre_de_tache
   For b = 1 To nombre_de_tache
    t_phero_somme(c, b) = (1 - rho_lsti) * t_phero_somme(c, b) +
    t_phero_iter(c, b)
   Next
  Next
Next
t_reseau(num_tache + 1, 17) = meilleur_res
End Sub
```

Sub de calcul des marges minimales.

```
Private Sub float_min(num_tache As Integer, nb_succ As Integer,
nb_pred As Integer)
nb_iteration = CInt(Text4.Text)
nb_fourmi = nombre_de_tache - 2
Dim i As Integer
Dim j As Integer
Dim f As Integer
Dim derniere_tache As Integer
Dim tache_choisie As Integer
Dim tir_alea As Single
Dim nombre_succ As Integer
Dim nombre_pred As Integer
ReDim t_chemin_fin(nb_succ + 1)
ReDim t_chemin_deb(nb_pred + 1)
Dim b As Integer
Dim d As Integer
Dim c As Integer
Dim num_succ As Integer
Dim num_pred As Integer
Dim dose_phero As Single
Dim q As Integer
Dim meilleur_res As Integer
Dim probabilite As Single
Dim res_fourmi As Integer
Dim chemin As String
'mettre en configuration de base
For c = 1 To nombre_de_tache
 t_duree(c) = t_reseau(c, 2)
Next
t_duree(num_tache + 1) = t_reseau(num_tache + 1, 3)
cpm
meilleur_res = t_marge(num_tache + 1)
q = t_marge(num_tache + 1)
```

```
'initialiser tableau phero
For i = 1 To nombre_de_tache
 For j = 1 To nombre_de_tache
  t_phero_iter(i, j) = 0
  t_phero_somme(i, j) = 0.5
 Next
Next
For i = 1 To nb_iteration
 For f = 1 To nb_fourmi
  'mettre en configuration de base
  For c = 1 To nombre_de_tache
   t_duree(c) = t_reseau(c, 2)
  Next
  t_duree(num_tache + 1) = t_reseau(num_tache + 1, 3)
  'effectuer le chemin vers le début
  derniere_tache = num_tache
  t_chemin_deb(1) = num_tache
  d = 2
  Do
   nombre_pred = t_reseau(derniere_tache + 1, 9)
   'choix de la tache
   Do
    Randomize Timer
    num_pred = Int(Rnd(Timer) * nombre_pred) + 1
    tir_alea = Rnd(Timer) * 1
    tache_choisie = t_reseau(derniere_tache + 1, 9 + num_pred)
    probabilite = proba_pred(derniere_tache, tache_choisie,
    alpha_float, beta_float)
   Loop Until tir_alea < probabilite
   t_chemin_deb(d) = tache_choisie
   derniere_tache = tache_choisie
   t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 3)
   d = d + 1
  Loop Until derniere_tache = 0
  'effectuer le chemin vers la fin
```

```
derniere_tache = num_tache
t_chemin_fin(1) = num_tache
b = 2
Do
 nb_succ = t_reseau(derniere_tache + 1, 4)
 'choix de la tache
 Do
  Randomize Timer
  num_succ = Int(Rnd(Timer) * nb_succ) + 1
  tir_alea = Rnd(Timer) * 1
  tache_choisie = t_reseau(derniere_tache + 1, 4 + num_succ)
  probabilite = proba_succ(derniere_tache, tache_choisie,
  alpha_float, beta_float)
 Loop Until tir_alea < probabilite
 t_chemin_fin(b) = tache_choisie
 derniere_tache = tache_choisie
 t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 3)
b = b + 1
Loop Until derniere_tache = nombre_de_tache - 1
cpm
res_fourmi = t_marge(num_tache + 1)
chemin = "0"
For c = d - 2 To 1 Step -1
 chemin = chemin & " => " & t_chemin_deb(c)
Next
For c = 2 To b - 1
 chemin = chemin & " => " & t_chemin_fin(c)
Next
'sauver résultat
If t_marge(num_tache + 1) < meilleur_res Then
 meilleur_res = t_marge(num_tache + 1)
End If
'deposer pheromone
dose_phero = (q + 1) / (t_marge(num_tache + 1) + 1)
```

```
   For c = 1 To d - 2
    t_phero_iter(t_chemin_deb(c) + 1, t_chemin_deb(c + 1) + 1) =
    t_phero_iter(t_chemin_deb(c) + 1, t_chemin_deb(c + 1) + 1) +
    dose_phero
   Next
   For c = 1 To b - 2
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) =
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) +
    dose_phero
   Next
  Next
  'evaporer
  For c = 1 To nombre_de_tache
   For b = 1 To nombre_de_tache
    t_phero_somme(c, b) = (1 - rho_float) * t_phero_somme(c, b) +
    t_phero_iter(c, b)
   Next
  Next
Next
t_reseau(num_tache + 1, 18) = meilleur_res
End Sub
```

Sub de calcul des marges maximales

```
Private Sub float_max(num_tache As Integer)
nb_iteration = CInt(Text4.Text)
nb_fourmi = nombre_de_tache - 2
Dim i As Integer
Dim j As Integer
Dim f As Integer
Dim derniere_tache As Integer
Dim tache_choisie As Integer
Dim tir_alea As Single
Dim nb_succ As Integer
ReDim t_chemin_fin(nombre_de_tache)
Dim b As Integer
Dim c As Integer
Dim num_succ As Integer
Dim dose_phero As Single
Dim q As Integer
Dim meilleur_res As Integer
Dim probabilite As Single
Dim res_fourmi As Integer
Dim chemin As String
'mettre en configuration de base
For c = 1 To nombre_de_tache
 t_duree(c) = t_reseau(c, 2)
Next

cpm
meilleur_res = t_marge(num_tache + 1)
q = t_marge(num_tache + 1)
'initialiser tableau phero
For i = 1 To nombre_de_tache
 For j = 1 To nombre_de_tache
  t_phero_iter(i, j) = 0
  t_phero_somme(i, j) = 0.5
```

```
 Next
Next
For i = 1 To nb_iteration
 For f = 1 To nb_fourmi
  t_chemin_fin(1) = 0
  derniere_tache = 0
  'mettre en configuration de base
  For c = 1 To nombre_de_tache
   t_duree(c) = t_reseau(c, 2)
  Next
  b = 2
  Do
   nb_succ = t_reseau(derniere_tache + 1, 4)
   'choix de la tache
   Do
    Randomize Timer
    num_succ = Int(Rnd(Timer) * nb_succ) + 1
    tir_alea = Rnd(Timer) * 1
    tache_choisie = t_reseau(derniere_tache + 1, 4 + num_succ)
    probabilite = proba_succ(derniere_tache, tache_choisie,
    alpha_float, beta_float)
   Loop Until tir_alea < probabilite
   t_chemin_fin(b) = tache_choisie
   derniere_tache = tache_choisie
   t_duree(tache_choisie + 1) = t_reseau(tache_choisie + 1, 3)
   b = b + 1
  Loop Until derniere_tache = nombre_de_tache - 1
  cpm
  res_fourmi = t_marge(num_tache + 1)
  chemin = "0"
  For c = 2 To b - 1
   chemin = chemin & " => " & t_chemin_fin(c)
  Next
  'sauver résultat
  If t_marge(num_tache + 1) > meilleur_res Then
```

```
    meilleur_res = t_marge(num_tache + 1)
   End If
   'deposer pheromone
   dose_phero = (t_marge(num_tache + 1) + 1) / (q + 1)
   For c = 1 To b - 2
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) =
    t_phero_iter(t_chemin_fin(c) + 1, t_chemin_fin(c + 1) + 1) +
    dose_phero
   Next
  Next
  'evaporer
  For c = 1 To nombre_de_tache
   For b = 1 To nombre_de_tache
    t_phero_somme(c, b) = (1 - rho_float) * t_phero_somme(c, b) +
    t_phero_iter(c, b)
   Next
  Next
Next
t_reseau(num_tache + 1, 19) = meilleur_res
End Sub
```

APPENDICE C

LES RÉSULTATS

C.1 TEMPS DE CALCUL EN SECONDES 83
C.2 ERREURS DE CALCUL 84

C.1 Temps de calcul en secondes

	Méthode			
	Monte-Carlo			Fourmi
	10 000 itérations	100 000 itérations	1 000 000 itérations	
25 tâches	0,50	4,00	37,00	2,69
60 tâches	1,00	9,00	87,00	16,66
90 tâches	1,50	13,00	130,00	46,88
120 tâches	2,00	18,00	172,00	142,30

C.2 Erreurs de calcul

		nb erreur			
		dédut au plus tard min	début au plus tard max	marge min	marge max
25 tâches	mc 10 000	11	6	0	4
	mc 100 000	9	7	0	4
	mc 1 000 000	7	4	0	4
	F 1-1	0	0	0	2
	F 1-2	0	2	0	3
	F 2-1	0	1	0	5
60 tâches	mc 10 000	27	15	39	22
	mc 100 000	26	5	37	10
	mc 1 000 000	26	4	37	6
	F 1-1	1	7	1	42
	F 1-2	1	9	4	50
	F 2-1	2	8	2	40
90 tâches	mc 10 000	58	21	37	38
	mc 100 000	55	20	36	30
	mc 1 000 000	52	16	34	26
	F 1-1	0	6	1	64
	F 1-2	2	7	3	71
	F 2-1	2	8	1	61
120 tâches	mc 10 000	65	101	74	61
	mc 100 000	64	97	70	37
	mc 1 000 000	63	96	68	20
	F 1-1	7	7	1	89
	F 1-2	8	7	5	97
	F 2-1	1	7	2	94