Université du Québec
à Rimouski

Proposition d'une méthodologie multicritère pour la résolution du problème d'ordonnancement d'un projet avec prise en compte des compétences et des ressources

Mémoire présenté

dans le cadre du programme de maîtrise en gestion de projet

en vue de l'obtention du grade de maître ès sciences

PAR

© Gabrielle Amyot Lachance

Août 2018

Composition du jury :

Didier Urli, président du jury, Université du Québec à Rimouski

Bruno Urli, directeur de recherche, Université du Québec à Rimouski

Anissa Frini, professeur, membre, Université du Québec à Rimouski

Daniel Leroy, Professeur Des Universités, Université de Tours, France

Dépôt initial le 6 juillet 2018 Dépôt final le 21 août 2018

REMERCIEMENTS

Je souhaite spécialement remercier mon directeur de recherche, monsieur Bruno Urli, pour sa présence, son temps, ses idées, son support et pour l'intérêt qu'il a démontré pour ma recherche, mais aussi pour ma réussite.

Je tiens également à remercier l'Université du Québec à Rimouski ainsi que son personnel, toujours accueillant et disposé à répondre à mes interrogations.

RÉSUMÉ

Cette recherche porte sur le problème d'ordonnancement d'un projet avec contraintes de ressources (RCPSP). Son objectif consiste à étudier deux extensions du problème de base en développant une méthode de résolution du RCPSP à critères multiples qui prend en compte les compétences maîtrisées par les ressources et les compétences requises par chaque activité du projet. Le projet étudié se base sur celui proposé par Montoya (2012), il se compose de quatre activités, de quatre ressources et de trois compétences. La résolution du problème se présente en deux étapes.

D'abord, le logiciel d'optimisation à critères multiples Midaco est utilisé afin d'obtenir des solutions de Pareto optimisant la durée ainsi que le coût du projet. Pour cette recherche, le logiciel Midaco est utilisé avec le logiciel de calcul Matlab, dans lequel le code représentant le problème est créé. À la fin de cette première étape, les meilleurs ordonnancements des activités selon les ressources et les compétences disponibles sont retenus afin de passer à l'étape suivante. Parmi tous les essais réalisés, sept d'entre eux sont retenus, proposant treize solutions optimales.

Ces points de Pareto retenus passent à l'étape suivante, la sélection de la solution de meilleur compromis à l'aide du logiciel d'aide à la décision Prométhée. Cet outil permet de sélectionner la meilleure solution de compromis selon les critères définis par l'utilisateur. Mis à part la durée et le coût du projet, le temps perdu est le troisième critère étudié, il s'agit du temps d'inactivité d'une ressource entre deux activités. Pour effectuer le choix de la solution finale, les trois critères sont pris en considération à poids égaux. D'autres simulations sont effectuées pour des poids différents afin d'observer l'évolution du rangement.

Cette étude contribue à la recherche en proposant une méthode de résolution pour deux extensions du problème d'ordonnancement d'un projet avec contraintes de ressources, les objectifs multiples et les compétences multiples.

Mots clés : RCPSP, objectifs multiples, compétences multiples, gestion de projet, métaheuristique, Midaco, Prométhée, optimisation, points de Pareto

ABSTRACT

This research is about resource-constrained project scheduling problem (RCPSP). The objective is to study two extensions of the basic problem by developing a method to solve the multi-objective RCPSP while considering the skills mastered by the resources and the skills required by each project activity. The studied project is based on the one proposed by Montoya (2012). It consists of four activities, four resources and three skills. The problem is solved in two steps.

First, the multi-criteria optimization software Midaco is used to obtain Pareto points, used to optimize the duration and the cost of the project. For this search, the Midaco software is used with Matlab, in which the code representing the problem is created. At the end of this first step, the best schedules according to available resources and skills are kept in order to take the next step. Among all the tests carried out, seven of them are selected, proposing thirteen optimal solutions.

The second step is to select the best compromise solution by using the Promethee decision support software. This tool allows you to select the best compromise solution according to the criteria defined by the user. Cost, duration and lost time are the three criteria studied. The last one is about the inactivity time of a resource between two activities. To choose the best solution, these three criteria are taken into account with equal weights.

This study contributes to the research by proposing a resolution method for two extensions of the RCPSP, the mutli-objective and the multi-skill optimizations.

Keywords: RCPSP, multi-objective, multi-skill, project management, meta-heuristic, Midaco, Promethee, optimization, Pareto points

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION GÉNÉRALE

La planification d'un projet consiste généralement à coordonner ses différentes tâches dans le temps. Cependant, la gestion des ressources présente aussi une importance non négligeable au sein des organisations. En effet, les entreprises œuvrant en gestion de projet sont continuellement confrontées au problème de planification de projets sous contraintes de ressources (RCPSP). Dans ce type de problème, les activités d'une planification générée pour un projet sont confrontées à des contraintes de précédence ainsi qu'à des contraintes de ressources qui possèdent des compétences particulières et ces activités ne peuvent pas être interrompues lorsqu'elles sont commencées.

Cette recherche prend en considération ces deux extensions du problème de base, le problème à objectifs multiples et à compétences multiples. Son objectif consiste à étudier développer une méthode de résolution du RCPSP à critères multiples en considérant les compétences maîtrisées par les ressources et les compétences requises par chaque activité du projet.

Dans la littérature, certains auteurs traitent du RCPSP à objectifs multiples ou à compétences multiples. Pour l'optimisation à critères multiples, Kreszowska (2009), Leu et Yang (1999) et Cai et Li (2000) proposent des algorithmes génétiques, Viana et se Souza (2000) présentent un algorithme de recuit simulé et *Gomar et al.* (2000) développent un modèle de programmation linéaire.

En ce qui concerne l'optimisation à compétences multiples, Bellenguez et Néron (2005), Bellenguez-Morineau (2008) et Bellenguez-Morineau et Néron (2007) étudient le problème en supposant que certaines ressources sont indisponibles sur certaines périodes. Firat et Hurkens (2012) étudient le problème de charge de travail associée aux projets à compétences multiples. Correia, Lampreia-Lourenço et Saldanha-da-Gama (2012) cherchent à minimiser la durée totale du projet où chaque activité peut nécessiter

plusieurs compétences. Montoya (2012) propose plusieurs méthodes pour résoudre le RCPSP à compétences multiples.

Cai et Li (2000) et Certa *et al.* (2009) font partie des quelques auteurs qui étudient les deux extensions présentées dans un seul problème. Cari et Li (2000), bien qu'ils visent à optimiser plusieurs critères avec un algorithme génétique, considèrent aussi que les activités nécessitent différentes compétences et que les ressources possèdent une ou plusieurs d'entres elles. Certa *et al.* (2009), quant à eux, étudient un contexte à objectifs, projets et compétences multiples. Ils proposent une méthode multiobjectif afin d'allouer différentes ressources possédant différents niveaux de compétences aux activités des projets qui en nécessitent plusieurs.

La méthode proposée dans ce mémoire se compose de deux étapes. D'abord, le logiciel d'optimisation à critères multiples Midaco est utilisé afin d'obtenir des solutions de Pareto optimisant la durée ainsi que le coût du projet. À cette étape, aucune préférence n’est accordée à un critère ou à l’autre. Pour cette recherche, le logiciel Midaco est utilisé avec le logiciel de calcul Matlab, dans lequel le code représentant le problème est créé. À la fin de cette première étape, les meilleurs ordonnancements des activités selon les ressources et les compétences disponibles sont retenus afin de passer à l'étape suivante.

Les points de Pareto retenus sont utilisés à la seconde étape, la sélection de la meilleure solution à l'aide du logiciel d'aide à la décision Prométhée. Cet outil permet de sélectionner la solution de meilleur compromis ainsi que de classer les différentes solutions selon les critères définis par l'utilisateur et de l'importance accordée à chacun d'entre eux. C’est à cette étape que l’utilisateur présente ses préférences, en accordant un poids à chacun des critères présents.

Le prochain chapitre présente une revue de la littérature portant principalement sur le problème d'ordonnancement d'un projet avec contraintes de ressources et ses extensions. Le second chapitre présente le modèle et sa méthode de résolution. Le problème, ses équations, la recherche de solutions Pareto avec Midaco, la sélection d'une solution du meilleur compromis avec Prométhée et un exemple numérique y sont

présentés. Puis, une conclusion fait le point sur la présente recherche et propose des pistes de solutions pour les recherches à venir.

CHAPITRE 1
REVUE DE LITTÉRATURE

La revue de littérature porte essentiellement sur la présentation du RCPSP ainsi que des méthodes développées pour le résoudre. L'objectif de la recherche y est aussi présenté.

1.1 Définition générale du RCPSP

Le RCPSP est l'un des problèmes classiques d'ordonnancement les plus étudiés dans la littérature (Laborie, 2005) et il peut être considéré comme un problème général dans la classe des problèmes d'ordonnancements cumulatifs (Baptiste, le Pape et Nuijten, 1999), où des ressources sont partagées entre différentes activités.

Le problème a été abordé par de nombreux chercheurs, dont Brucker *et al.* (1999), Herroelen, De Reyck et Demeulemeester (1998), Icmeli *et al.* (1993), Kolisch et Padman (2001), Özdamar et Ulusoy (1995) et bien d'autres.

Il porte sur la planification d'un unique projet conçu d'un ensemble d'activités de durées et de besoins en ressources spécifiques, déterminés avant le début du projet. Lors de la réalisation de chaque activité, une quantité constante de ressources est utilisée sur toute la période prévue (Debels *et al.*, 2006). La préemption n'est pas autorisée, donc aucune activité ne peut être interrompue en faveur d'une activité de priorité supérieure. Une activité débutée n'est arrêtée que lorsqu'elle est terminée, elle ne peut être interrompue (Hartmann, 2002). Chaque activité possède un ensemble de prédécesseurs et de successeurs immédiats et les activités sont planifiées selon les relations de précédence. Cela spécifie donc l'ordre dans lequel les activités doivent être exécutées.

Les ressources disponibles, quant à elles, sont connues et elles sont d'une disponibilité limitée. Les ressources sont renouvelables, donc elles sont disponibles en quantité constante tout au long du projet (Mingozzi *et al.*, 1998). En effet, ce type de ressources permet qu'elles soient utilisées à chaque période où elles sont disponibles et à leur pleine capacité (Montoya, 2012). Il peut s'agir, par exemple, des employés, des machines, des équipements, des ressources externes ou tout autre élément étant en mesure d'exécuter une activité requise pour l'achèvement du projet (Krzeszowska, 2009).

Les activités sont reliées entre elles par deux types de contraintes, les contraintes de précédence et les contraintes de ressources. Les contraintes de précédence forcent les activités qui succèdent à débuter seulement lorsque les activités qui les précèdent sont achevées (Kolisch et Hartmann, 2006). Pour qu'une activité puisse débuter, toutes celles qui la précèdent immédiatement doivent être terminées (Briand et Bezanger, 2006). Tel que mentionné, chaque activité nécessite une certaine quantité de ressources constante tout au long de sa durée pour être réalisée, mais ces dernières sont d'une disponibilité limitée (Palpant, Artigues et Michelon, 2004). Afin qu'une activité soit exécutée, toutes les ressources nécessaires doivent être disponibles, il s'agit donc des contraintes de ressources. En effet, les limites définies d'un ensemble de ressources renouvelables sont celles créées par la quantité de ressources disponibles par période de temps pour chacune d'elles (Bouleimen et Lecocq, 2003).

L'objectif du RCPSP est de déterminer le moment où chaque activité doit débuter afin de minimiser la durée totale du projet (Thomas et Salhi, 1998). La planification doit être réalisée de sorte que les contraintes de précédence et de ressources soient satisfaites (Valls, Ballestin et Quintanilla, 2004). Pour qu'une planification soit réalisable, les activités doivent être planifiées afin que la consommation de ressources n'excède pas la capacité de chacune à tout moment du déroulement du projet (Debels *et al.*, 2006).

1.2 Nomenclature du problème général

De manière plus formelle, le RCPSP peut être défini comme un problème d'optimisation combinatoire (Artigues, Demassey et Néron, 2008). Ce type de problème se définit comme un espace de solution X, discret ou pouvant être réduit à un ensemble discret, et un sous-ensemble fini de solutions réalisables $Y \subseteq X$ associé à une fonction objectif $f : Y \longrightarrow \mathbb{R}$. Le problème d'optimisation combinatoire vise à trouver une solution réalisable $y \in Y$ afin de minimiser la fonction objectif f(y) (Hao, Galinier et Habib, 1999). Le RCPSP est donc un problème d'optimisation combinatoire défini par les variables suivantes, ayant chacune des valeurs entières et positives.

Un projet est constitué d'activités définies par l'ensemble $A=\{A_0 , ... , A_N\}$, où les activités A_0 et A_N sont des activités fictives représentant le début et la fin du projet (Debels *et al.*, 2006). Elles ont une durée nulle et ne consomment aucune ressource. L'ensemble des activités réelles du projet est donc identifié par $A=\{A_1 , ... , A_{N-1}\}$.

Le paramètre p est utilisé pour représenter la durée d'une activité, où p_i correspond à la durée de l'activité A_i. Étant donné que les activités A_0 et A_N ont une durée nulle, $p_0 = p_N = 0$ (Artigues, Demassey et Néron, 2008).

Les relations de précédence entre les activités sont données par E, où $(A_i , A_j) \in E$ signifie que l'activité A_i précède l'activité A_j. Chaque activité A_i possède un ensemble de prédécesseurs E_i- et de successeurs E_i-+ (Montoya, 2012). Les relations de précédence peuvent être représentées par un graphique acyclique d'activité sur nœud, noté G = (A,E). Dans ce réseau, les activités sont représentées par des nœuds et les relations de précédence sont représentées par des arcs. Les activités fictives et les activités réelles sont représentées sur la graphique, A_0 précède toutes les activités et A_N les succède (Bellenguez et Néron, 2005).

Les ressources renouvelables composent l'ensemble $W = \{W_0 , ... , W_m\}$ et la disponibilité d'une ressource W_k correspond à B_k. Si $R_k = 1$, il s'agit d'une ressource unitaire, qui peut exécuter uniquement une activité à la fois.

Les demandes des activités pour des ressources spécifiques sont définies par b, où b_{ik} représente la quantité de la ressource W_k utilisée par unité de temps durant l'exécution de l'activité A_i.

Le temps de départ d'une activité est identifié par t_i, donc l'ensemble des temps de départ du projet est défini par $t=\{t_0 , ... , t_N\}$. Le temps de complétion d'une activité A_i est identifié par C_i.

Sachant que chaque activité $i \in A$ a une durée $p_i \geq 0$ et que, pour chaque ressource, la demande $b_{ik} \geq 0$, le problème peut être formulé comme suit (Palpant, Artigues et Michelon, 2004) :

$$Min\ C_N = t_N \tag{1}$$

$$t_j - t_i \geq p_i \quad \forall\ (A_i, A_j) \in E \tag{2}$$

$$\sum_{A_i \in A_t} b_{ik} \leq B_k \quad \forall\ R_k \in R, \forall t \geq 0 \tag{3}$$

où $A_t = \{A_i \in A \mid t_i \leq t < t_i + p_i\}$ représente l'ensemble des activités réelles du projet en cours au temps t (Artigues, Demassey et Néron, 2008).

L'objectif (1) vise à minimiser la durée totale du projet, définie par le temps de départ de l'activité A_N. La contrainte (2) représente les relations de précédence entre les activités et la contrainte (3) représente les contraintes de ressources, de sorte qu'à tout instant, la consommation d'une ressource n'excède pas sa capacité (Möhring et *al.*, 2003).

Tel que montré dans la littérature (Garey et Johnson, 1979; cité par Gonçalves, Mendes et Resende, 2007), le RCPSP, généralisation du problème classique d'ordonnancement, fait partie de la classe des problèmes d'optimisation NP-difficiles, dans le sens fort. Un problème est dit NP-difficile lorsqu'il n'y a pas d'algorithme connu

pour trouver des solutions optimales en un temps polynomial (Lenstra et Kan, 1978; cité par Browning et Yassine, 2010), ce pourquoi plusieurs chercheurs se concentrent sur les solutions optimales heuristiques et métaheuristiques (Browning et Yassine, 2010). En effet, lorsqu'un problème est NP-difficile, il est difficile de trouver une solution optimale dans un délai raisonnable, même pour des problèmes de petite taille (Fleszar et Hindi, 2004).

1.3 Types d'échéanciers

Les échéanciers des différents projets peuvent être classés selon trois types, soit l'échéancier semi-actif, l'échéancier actif et l'échéancier sans retard (Mendes, Gonçalves et Resende, 2009). Ces types d'échéanciers sont basés sur la notion de décalages à gauche local et global.

Considérant un échéancier S et une activité A_i, le décalage à gauche est dit global lorsque l'opérateur $LS(S,A_i,\Delta)$ modifie l'échéancier S en un échéancier identique S', mis à part pour S_i, où $S_i' = S_i - \Delta$, avec $\Delta > 0$. Un décalage à gauche est dit local lorsqu'en plus, tous les échéanciers obtenus par l'opérateur $LS(S,A_i,\rho)$, avec $0 < \rho < \Delta$ sont aussi réalisables (Artigues, Demassey et Néron, 2008). En effet, le décalage à gauche local se définit par l'action de déplacer un bloc d'opération, représentant une activité vers la gauche sur un diagramme de Gantt en préservant la séquence des opérations (Baker, 1974; cité par Sprecher, Kolisch et Drexl, 1995). Donc, le nouvel échéancier avec le décalage local à gauche est obtenu par un ou plusieurs décalages à gauche successifs d'une activité A_i, d'une unité de temps à la fois (Sprecher, Kolisch et Drexl, 1995).

L'échéancier est dit semi-actif s'il ne permet aucun décalage local à gauche. Dans ce type d'échéancier, aucune activité ne peut démarrer plus tôt sans modifier la séquence prévue (Mendes, Gonçalves et Resende, 2009). Un échéancier est donc dit semi-actif si et seulement si, pour chaque activité $A_i \in A$, aucun décalage à gauche d'une unité de temps

n'est possible. Un échéancier semi-actif peut être transformé en un échéancier actif en appliquant une série de décalages à gauche globaux (Sprecher, Kolisch et Drexl, 1995).

Un échéancier actif, quant à lui, ne présente aucun décalage à gauche global possible. Il peut être défini comme un échéancier où aucune activité ne peut débuter plus tôt sans retarder d'autres activités ou sans briser une contrainte de précédence (Valls *et al.*, 2009). Un échéancier est donc actif si et seulement si, pour chaque activité $A_i \in A$, aucun décalage à gauche de $\Delta \in \mathbb{N}^*$ unités de temps n'est possible (Sprecher, Kolisch et Drexl, 1995). Un échéancier actif est toujours optimal, donc l'espace de recherche peut être limité à l'ensemble des échéanciers actifs.

L'échéancier sans retard représente un échéancier réalisable dans lequel aucune ressource n'est gardée inactive alors qu'elle pourrait commencer une activité. Dans ce type d'échéancier, les activités sont planifiées le plus tôt possible, tout en respectant les contraintes de précédence et de ressource (Cho et Kim, 1997). L'échéancier sans retard est donc actif et semi-actif.

1.4 Environnement

Deux types d'environnement sont envisageables, l'environnement statique et l'environnement dynamique. Lorsque l'environnement est statique, l'échéancier optimal est déterminé en caractérisant le temps de départ de chaque activité et en respectant les relations de précédence et les contraintes de ressource.

Lorsque l'environnement est dynamique, des changements du problème initial sont possibles en temps réel, lors de l'exécution du projet. L'échéancier statique initial doit donc être adapté lorsqu'un changement survient (Artigues, Michelon et Reusser, 2003).

1.5 Méthodes de résolution

Les méthodes de résolution du RCPSP telles que les méthodes exactes, les heuristiques et les métaheuristiques sont présentées dans ce qui suit.

1.5.1 Méthodes exactes

Une méthode exacte est utilisée, comme son nom l'indique, pour obtenir une solution exacte au RCPSP (Artigues, Demassey et Néron, 2008). Ce type de méthode permet cependant d'obtenir une solution optimale seulement pour un projet de taille modérée (Carlier et Latapie, 1991). Les chercheurs s'entendent sur le fait que les méthodes exactes sont efficaces pour des projets possédant entre 30 (Valls, Ballestin et Quintanilla 2004) et 60 activités (Valls *et al.*, 2008). Malgré cet inconvénient, ces méthodes peuvent tout de même être utilisées pour des problèmes de plus grande taille afin d'obtenir une solution approchée de la solution exacte dans un temps raisonnable (Carlier et Latapie, 1991).

Les méthodes exactes se divisent en deux grandes catégories qui sont les méthodes arborescentes et les méthodes basées sur la programmation mathématique.

1.5.1.1 Méthodes arborescentes

Les méthodes arborescentes fonctionnent avec la création d'un arbre de recherche et visent à explorer l'ensemble des solutions réalisables. L'arbre de recherche est divisé en sous-ensembles de plus en plus petits afin d'isoler la solution optimale (Carlier et Latapie, 1991). Lorsque chaque nœud est créé, deux évènements sont possibles : une solution réalisable peut être déduite ou l'espace de recherche correspondant au nœud à l'étude est divisé en sous-ensembles. Cette seconde action se nomme le branchement (Artigues, Demassey et Néron, 2008).

Les auteurs ont proposé différents schémas de branchement, les premiers sont de type chronologique, où chaque nœud de l'arbre est associé à une solution partielle, soit un échéancier partiel de l'étape de branchement, auquel au moins une activité est ajoutée afin d'avancer dans la planification du projet. Les schémas de branchement de type chronologique ont pour avantage d'avoir un échéancier partiel fixé du début du temps de départ du projet jusqu'au temps courant du nœud à l'étude. Il est facile de vérifier, pour chaque nœud, si la solution partielle correspond à un ensemble semi-actif afin de réduire l'espace de recherche (Artigues, Demassey et Néron, 2008).

Une première méthode introduite et élaborée par Patterson *et al.* (1989; cité par Sprecher, 2000) considère toutes les activités pertinentes et, à partir de l'échéancier vide, représentant l'activité de départ du projet, des nœuds sont créés selon les activités éligibles, soit les activités dont tous les prédécesseurs sont déjà planifiés. L'activité ajoutée à l'échéancier partiel est planifiée le plus tôt possible, respectant toujours les contraintes de précédence et de ressources. Donc, à partir de l'échéancier vide, un nœud est créé pour chaque activité sans prédécesseur. Ensuite, pour chaque nœud, d'autres sont créés selon le nombre d'activités éligibles et, pour chacun d'entre eux, la planification partielle est créée en ajoutant l'activité (Sprecher, 2000). Cette méthode permet d'identifier les noeuds pertinents et de se diriger vers la solution optimale.

Baptiste *et al.* (1999) proposent, quant à eux, de construire l'échéancier partiel en ajoutant, à chaque nœud, une activité à la fois. Pour chaque nœud, l'ensemble des activités éligibles est déterminé, celle possédant le temps de départ le plus tôt est choisie et deux nœuds sont créés. Pour le premier nœud, l'activité choisie A_i est ajoutée à la solution partielle et planifiée le plus tôt possible et, pour le second, il est considéré qu'il est impossible que l'activité choisie soit la première des activités éligibles planifiées et il est imposé qu'au moins une activité dans l'ensemble éligible débute simultanément ou avant A_i. De cette manière, les solutions partielles construites ne sont pas nécessairement semi-actives, alors lorsqu'une activité est ajoutée à une solution, il est essentiel d'effectuer cette vérification (Artigues, Demassey et Néron, 2008).

Des auteurs présentent des méthodes arborescentes où plusieurs activités sont ajoutées aux solutions partielles. Christophides *et al.* (1987; cité par Bouleimen et Lecocq, 2003) proposent une méthode où, pour un nœud quelconque étudié au temps t correspondant à la fin au plus tôt d'une activité planifiée à un niveau inférieur, les activités en cours et les activités éligibles au temps donné sont identifiées. Un nœud est créé, la totalité des activités éligibles est ajoutée à l'échéancier partiel et, si aucun conflit de ressource ne survient, le temps t est incrémenté. Cependant, si des conflits se présentent, des nœuds sont créés pour chacune des combinaisons possibles entre les activités en cours et les activités éligibles ainsi que pour les combinaisons entre les activités éligibles elles-mêmes (Artigues, Demassey et Néron, 2008). Stinson *et al.* (1978) proposent d'abord d'énumérer les sous-ensembles pouvant être ajoutés aux solutions partielles sans créer de conflit de ressource et Deumelemeester et Herroelen (1997) présentent des améliorations et des corrections à la méthode.

Dans le même ordre d'idées, Mingozzi *et al.* (1998) n'ajoutent pas une ou plusieurs activités à une solution partielle, mais ils proposent d'ajouter des blocs d'activités aux nœuds de l'arbre de recherche. Chacun d'entre eux représente un ensemble d'activités pouvant être planifiées en même temps sans violer aucune contrainte et dont la durée correspond au temps de complétion d'une activité du bloc, les autres pouvant toujours être en progrès. Le temps de départ de chaque nœud correspond au temps de fin du bloc précédent et il y possible d'y retrouver l'ensemble d'activités complétées, l'ensemble d'activités en cours et l'ensemble des blocs candidats, créés à partir des différentes activités éligibles et en progrès. Les blocs candidats peuvent d'ailleurs être créés avant l'exploration de l'arbre de recherche (Artigues, Demassey et Néron, 2008).

Carlier et Latapie (1991) proposent une procédure arborescente basée sur celle des intervalles, développée par Carlier (1987; cité par Carlier et Latapie, 1991). Les temps de départ possibles de l'activité critique sont déterminés, créant ainsi un intervalle d'exécution. Pour chacune d'entre elles, deux nœuds sont créés, divisant ainsi l'intervalle d'exécution en deux. Le premier nœud comprend la première moitié des intervalles d'exécution et le deuxième nœud comprend la seconde moitié.

Les auteurs ont aussi proposé différentes règles de dominance afin de restreindre l'espace de recherche d'échéanciers actifs ou semi-actifs. Par exemple, Demeulemeester et Herroelen (1997) proposent une règle de dominance basée sur la comparaison entre deux nœuds de l'arbre de recherche, soit le nœud actuel et le nœud de l'étape précédente. La règle de l'ensemble coupé utilise l'information enregistrée sur un échéancier partiel précédent afin de vérifier si une solution obtenue par l'échéancier actuel peut être meilleure que celle obtenue précédemment. Si ce n'est pas le cas, le retour en arrière est possible (Brucker *et al.*, 1999). Plusieurs auteurs s'entendent sur le fait qu'il s'agit de l'une des règles de dominance les plus efficaces pour les problèmes d'optimisation classiques à avoir été proposée dans la littérature (Kolisch, 1996a; Kolisch, 1996b).

La règle de dominance du décalage à gauche, introduite par Sprecher (2000), est aussi utilisée. Considérant un échéancier partiel, si une activité planifiée peut être déplacée vers la gauche, que les contraintes de précédence et de ressources sont respectées et que la durée de l'échéancier partiel diminue, celui-ci est dominé (Artigues, Demassey et Néron, 2008).

Brucker *et al.* (1998), Sprecher, Kolisch et Drexl (1995), Balas (1971), Davis et Heidom (1971), Hastings (1972), Talbot et Patterson (1978) et Bell et Park (1990) proposent aussi des méthodes arborescentes pour résoudre le RCPSP.

1.5.1.2 Méthodes basées sur la programmation mathématique

Alvarez-Valdés et Tamarit (1993) proposent une programmation linéaire basée sur la sélection d'une séquence d'activités respectant les contraintes de précédence et, par la suite, sur la sélection du temps de départ de chaque activité afin de respecter les contraintes de ressources. La programmation qu'ils proposent est basée sur l'ensemble interdit minimal. Demassey, Artigues et Michelon (2002) obtiennent aussi des résultats intéressants à partir d'une méthode basée sur ce même concept.

Artigues (2003) présente une autre méthode basée sur la séquence. Les ressources utilisées pour une activité sont libérées à la fin de celle-ci et deviennent disponibles pour une autre activité, le modèle mathématique se base donc sur le flux de ressources entre les activités, soit la quantité d'une ressource transférée de l'activité i à l'activité j. Neumann *et al.* (2003), Artigues et Roubellat (2000) et Schwindt (2005) appliquent cette méthode pour formuler leurs algorithmes.

Plusieurs formulations mathématiques reposent sur l'utilisation des ressources et l'avancement des activités durant le temps. Des auteurs ont conçu des modèles en se basant sur l'indexation du temps d'un projet, où un horizon temporel dans lequel il doit être complété est défini. Pour toute la durée de l'horizon temporel, le temps est incrémenté d'une unité de temps à la fois et, à chaque unité de temps, une variable booléenne est utilisée afin de déterminer si l'activité j débute au temps t. Pritsker, Watters et Wolfe (1969), Christofides *et al.* (1987) et Uetz (2001) ont développé des modèles dans lesquels une contrainte linéaire est ajoutée à chaque période de temps afin d'éviter les conflits de ressources. Fisher (1973) et Möhring *et al.* (2003) étudient aussi ce type de procédures en considérant la relaxation de Lagrange. Mingozzi *et al.* (1998), quant à eux, se basent d'abord sur la résolution des conflits de ressources et ensuite sur l'utilisation de la variable binaire. Brucker et Knust (2000), Baptiste et Demassey et Michelon (2004) et Demassey *et al.* (2004) proposent des méthodes basées sur cette dernière.

Carlier et Néron (2003) proposent aussi un modèle mathématique, basé à la fois sur la séquence et sur le modèle de Mingozzi *et al.* (1998)

1.5.2 Heuristiques

Les heuristiques sont indispensables lors de la résolution de problèmes d'ordonnancement avec de nombreuses variables, comme il est généralement le cas dans la réalité (Kolisch et Hartmann, 2006). En effet, l'heuristique semble être l'une des

meilleures manières d'approcher le RCPSP (Valls *et al.*, 2008) et elle est utilisée afin de générer une bonne solution dans un délai raisonnable (Kumann, Jegan et Raja, 2006).

1.5.2.1 Heuristiques constructives

Les heuristiques les plus simples sont les heuristiques constructives (Palpant, Artigues et Michelon, 2004), elles permettent de construire une solution faisable à un problème d'ordonnancement à partir de zéro par extension progressive d'un échéancier partiel (Kolisch et Hartmann, 2006). Le nombre d'étapes incluses dans l'heuristique constructive correspond généralement au nombre d'activités du projet à l'étude et, à chaque étape, une seule activité est sélectionnée afin d'être programmée parmi une série d'activités candidates (Palpant, Artigues et Michelon, 2004).

1.5.2.1.1 Schéma générateur d'échéanciers

L'ensemble des activités candidates et le processus de sélection des activités peuvent être déterminés par un schéma générateur d'échéanciers. En effet, le schéma générateur d'échéanciers se retrouve au cœur de plusieurs procédures heuristiques pour la résolution du RCPSP (Kolisch et Hartmann, 2006) et, lorsqu'il est combiné avec une liste des priorités, il est possible d'obtenir une solution réalisable se rapprochant de la solution optimale dans un délai acceptable (Briand et Bezanger, 2006). La liste des priorités peut être représentée par une liste d'activités dans un ordre de priorité non croissant ou par un vecteur de valeurs prioritaires, où, par exemple, le second élément est associé à l'activité ayant le second temps de départ le plus tôt (Artigues, Demassey et Néron, 2008). Dans le premier cas, il s'agit d'une représentation par la liste des activités et, dans le second cas, de la représentation par clé aléatoire. Hartmann et Kolisch (2000) ont démontré à l'aide de tests informatiques qu'en général, les méthodes qui utilisent la représentation par liste des activités sont plus performantes que celles aléatoires. Il existe d'autres types de

représentation, mais ce sont les deux types les plus importants (Debels *et al.*, 2006). Le schéma générateur d'échéanciers permet donc de planifier les activités d'un projet étape par étape jusqu'à ce qu'une solution réalisable soit obtenue (Valls *et al.*, 2004). Lorsqu'un temps de départ est assigné à une activité, celui-ci est permanent et ne peut être changé au cours d'une étape suivante de la procédure. Deux types de schémas générateurs d'échéanciers sont considérés, le schéma générateur d'échéanciers en série et le schéma générateur d'échéanciers en parallèle.

1.5.2.1.1.1 Schéma générateur d'échéanciers en série

La méthode sérielle a d'abord été proposée par Kelley (1963). Un schéma générateur d'échéanciers en série est constitué de n itérations et, pour chacune d'entre-elles, une activité est sélectionnée et ajoutée à l'échéancier partiel à son temps de départ le plus tôt, tout en respectant les contraintes de ressources et de précédence (Kolisch et Hartmann, 2006).

À chaque itération, deux ensembles d'activités disjoints y sont associés. Tout d'abord, l'ensemble des activités planifiées, qui sont déjà présentes dans l'échéancier partiel et, deuxièmement, l'ensemble de décision, ou l'ensemble des activités éligibles (Kolisch et Hartmann, 2006), qui contient les activités non planifiées, dont tous les prédécesseurs débutent dans l'ensemble des activités planifiées.

À chaque étape, une activité de l'ensemble de l'échéancier partiel débute à son temps de début le plus tôt. L'activité planifiée est retirée de l'ensemble de décision et est ajoutée à l'ensemble d'activités planifiées. Une activité peut être ajoutée à l'ensemble de décision lorsque tous ses prédécesseurs font partie de l'ensemble des activités planifiées.

Étant donné que les activités sont planifiées une à la fois (Debels *et al.*, 2006), l'algorithme sériel contient le même nombre d'étapes que le nombre d'activités du projet et il se termine lorsque toutes les activités se trouvent dans l'échéancier partiel (Kolish, 1996 b).

La méthode sérielle génère des échéanciers actifs (Kolisch et Hartmann, 2006) et elle est avantageuse puisqu'elle est très simple à mettre en œuvre (Carlier et Latapie, 1991).

1.5.2.1.1.2 Schéma générateur d'échéanciers en parallèle

Deux algorithmes principaux sont associés au schéma générateur d'échéanciers en parallèle (Kolisch, 1996b), celui de Kelley (1963) et celui de Brooks (Bedworth et Bailey, 1982)

Le schéma générateur d'échéanciers en parallèle est, contrairement à la méthode sérielle, orienté sur le temps et, à chaque étape, un ensemble d'activités est planifié et chaque étape est associée à un temps t planifié (Kolisch, 1996b).

À chaque temps t, trois ensembles disjoints sont présents (Kolisch, 1996a). Tout d'abord, l'ensemble des activités planifiées est divisé en deux sous-ensembles, les activités qui étaient planifiées et qui sont complétées au temps planifié t ainsi que les activités qui étaient planifiées, mais qui sont toujours actives au temps t. Puis, tout comme dans la méthode sérielle, il est possible de retrouver l'ensemble de décision, contenant toutes les activités non planifiées qui sont disponibles pour l'être en respectant les contraintes (Kolisch, 1996b).

À chaque étape, l'échéancier partiel est donc conçu des activités incluses dans l'ensemble des activités planifiées et l'ensemble des activités planifiées, mais non complétées. Le temps de planification d'une étape est déterminé par le temps de fin au plus tôt des activités comprises dans l'échéancier partiel de l'étape précédente (Kolisch, 1996b).

Chaque étape se compose de deux sous-étapes. Tout d'abord, le nouveau temps de planification t est déterminé et les activités ayant un temps de fin inférieur ou égal au temps t sont retirées de l'ensemble actif et ajoutées dans l'ensemble complété (Briand et

Bezanger, 2006). À ce moment, des activités peuvent aussi être ajoutées dans l'ensemble de décision. Deuxièmement, une ou des activités de l'ensemble de décision sont sélectionnées à l'aide d'une règle de priorité et sont planifiées au temps t. Ces activités sont retirées de l'ensemble de décision et sont ajoutées à l'ensemble actif. L'algorithme est terminé lorsque toutes les activités du projet se retrouvent dans l'ensemble d'activités actives ou dans l'ensemble d'activités complétées (Kolisch, 1996b).

Les deux méthodes peuvent sembler similaires, cependant, l'échéancier généré par la méthode sérielle est considéré comme un échéancier actif tandis que l'échéancier généré par la méthode en parallèle fait uniquement partie des échéanciers sans retard (Artigues, Demassey et Néron, 2008). Il est connu qu'au moins une solution optimale est active alors que les échéanciers sans retard peuvent parfois exclurent toutes les solutions optimales. C'est donc pour cette raison que, contrairement aux méthodes sérielles, les méthodes en parallèle sont parfois incapables de générer une solution optimale (Kolisch, 1996b).

1.5.2.1.2 Méthodes à passages uniques et multiples

D'un point de vue opérationnel, les schémas générateurs d'échéanciers peuvent être appliqués aux heuristiques à passage unique ainsi qu'aux heuristiques à passages multiples.

Les méthodes à passage unique utilisent un schéma générateur d'échéanciers et une règle de priorité afin d'obtenir une solution réalisable. Ces méthodes ne génèrent qu'une seule planification (Kolisch et Hartmann, 2006).

Gonguet (1969) et Pascoe (1966) utilisent la méthode à passage unique avec un schéma générateur d'échéanciers en série alors que Alvarez-Valdés et Tamarit (1989a; 1989b), Davis et Patterson (1975), Shaffer *et al.* (1965), Elsayed (1982), Thesen (1976), Valls *et al.* (1992), Whitehouse et Brown (1979) et Pascoe (1966) utilisent la méthode à passage unique avec un schéma générateur d'échéanciers en parallèle.

Les méthodes à passages multiples sont généralement basées sur une application itérative du schéma générateur d'échéanciers en série ou en parallèle afin de générer plusieurs planifications. Plusieurs méthodes de combinaison des schémas générateurs d'échéanciers et des règles de priorité sont utilisées, les plus communes sont les méthodes des règles de priorité multiples, les méthodes avant-arrière et les méthodes d'échantillonnage (Kolisch et Hartmann, 2006).

Les méthodes des règles de priorité multiples utilisent des schémas générateurs d'échéanciers à plusieurs reprises et, lors de chaque utilisation, une règle de priorité différente est utilisée. Boctor (1990), Thomas et Salhi (1997; cité par Kolisch et Hartmann, 2006) et Ulusoy et Özdamar (1989) utilisent ces méthodes avec des schémas générateurs d'échéanciers en parallèle.

Les méthodes de planification avant-arrière appliquent des schémas générateurs d'échéanciers afin de planifier un projet de manière itérative en avançant et en reculant sur le réseau de précédence. Palpant *et al.* (2004), Li et Willis (1992;) et Özdamar et Ulusoy (1994; 1996) ont étudié ces méthodes.

Les méthodes d'échantillonnage utilisent généralement un schéma générateur d'échéanciers ainsi qu'une règle de priorité. Différents échéanciers sont obtenus en sélectionnant une règle de priorité de manière aléatoire. Aussi, il ne s'agit pas d'une valeur de priorité qui est associée à chaque activité, mais une probabilité de sélection qui est calculée. Cette probabilité de sélection représente la probabilité qu'une activité de l'ensemble de décision soit sélectionnée. Alvarez-Valdés et Tamarit (1989b), Cooper (1976), Schirmer et Riesenberg (1997), Drexl (1991) et Kolisch (1995) appliquent ces méthodes.

1.5.2.1.3 Règles de priorité

Tel que mentionné à la section 1.6.2.1.1, les heuristiques basées sur des schémas générateurs d'échéanciers utilisent des règles de priorités. Ces dernières sont utilisées afin

de sélectionner les activités à ajouter à l'ensemble de décision (Kolisch et Hartmann, 2006). Les heuristiques basées sur des règles de priorité ont l'avantage d'être robustes, intuitives, faciles à implanter et rapides en termes de temps de calcul (Brucker *et al.*, 1999).

Dans la littérature, il est possible de trouver de nombreuses règles de priorité. Pour les projets uniques, les auteurs utilisent, par exemple, les règles suivantes : premier arrivé, premier servi (FCFS), plus courte activité en premier (SOF), plus longue activité en premier (MOF), sélection au hasard (RAN), date d'échéance de l'activité la plus proche (EDDF), dernier arrivé, premier servi (LCFS), temps minimum de fin le plus tard (MINLFT), quantité de successeurs totaux maximale (MS), quantité de successeurs critiques maximale (MCS), importance la plus élevée, et bien d'autres (Browning et Yassine, 2010)

1.5.2.2 Recherche par voisinage local

Les méthodes de recherche par voisinage local sont des méthodes itératives qui, à partir d'une solution initiale et pour chaque itération, appliquent des modifications locales à la solution actuelle. La solution initiale est généralement calculée à partir d'une méthode heuristique.

Chaque méthode définit un ensemble de solutions voisines, soit les solutions pouvant être atteintes à partir de la solution initiale. Lors du choix de l'ensemble de solutions voisines, il est important de s'assurer qu'il existe une séquence de mouvements permettant d'obtenir une solution. La grosseur de l'ensemble voisin est aussi importante à considérer, plus celui-ci contient de possibilités, plus l'espace de recherche est large à chaque itération, permettant ainsi de trouver des solutions plus intéressantes. Cependant, l'augmentation de la grosseur de l'espace de recherche augmente aussi le temps de calcul (Artigues, Demassey et Néron, 2008).

Boctor (1996), Bouleimen et Lecocq (2003), Lee et Kim (1996), Valls *et al.* (2004), Merkle et Middendorf (2002), Debels *et al.* (2006), Artigues *et al.* (2003), Palpant, Artigues et Michelon (2004), Godard, Laborie et Nuijten (2005) et Kochetov et Stolyar (2003) utilisent les méthodes de recherche par voisinage local dans leurs études.

1.5.3 Métaheuristiques

Les métaheuristiques, tout comme les heuristiques, sont utilisées afin d'obtenir des solutions se rapprochant de l'optimum. Les métaheuristiques sont des heuristiques qui améliorent une solution initiale en exécutant successivement des opérations afin de transformer une ou plusieurs solutions en d'autres (Mendes, Gonçalves et Resende, 2009). Elles sont stochastiques, donc elles sont en mesure de faire face à l'explosion combinatoire de possibilités et elles s'inspirent d'analogies basées sur la physique, la biologie ou l'éthologie. (Dréo *et al.*, 2003) Différentes approches telles que le recuit simulé, la méthode tabou, les métaheuristiques basées sur la population, les algorithmes génétiques et les algorithmes de colonies de fourmis sont proposées dans la littérature. Les paramètres de ces méthodes peuvent être difficiles à régler et le temps de calcul peut être élevé.

1.5.3.1 Recuit simulé

La méthode de recuit simulé, étudiée par des chercheurs tels que Metropolis *et al.* (1953; cité par Boctor, 1996) ainsi que Kirkpatrick *et al.* (1983; cité par Kolisch et Hartmann, 2006), se base sur le processus physique de recuit utilisé par les métallurgistes afin d'obtenir un solide dans un état bien ordonné et à un niveau d'énergie minimal. La technique du recuit consiste à porter le matériau à haute température, puis, contrairement à la technique de la trempe, à abaisser lentement sa température. (Dréo *et al.*, 2005)

L'analogie entre la méthode de recuit simulé et le procédé du recuit se trouve au niveau de la fonction objectif du problème d'optimisation, qui, tout comme la température du matériau, est minimisée à l'aide d'un paramètre de contrôle de l'algorithme pour le recuit simulé et par l'introduction d'une température fictive pour le recuit. (Dréo *et al.*, 2005)

La méthode de recuit simulé se base sur l'algorithme de Metropolis (Métropolis *et al.*, 1953; cité par Boctor, 1996), permettant de décrire le comportement d'un système en équilibre thermodynamique à une température T donnée. À partir d'une configuration initiale donnée, une modification élémentaire, soit une variation d'énergie ΔE, est apportée afin de générer une solution voisine. Si cette dernière est plus performante que la première, soit qu'elle permet de diminuer la fonction objectif du problème, ou l'énergie, elle est acceptée. Le paramètre de contrôle de l'algorithme est alors diminué, une nouvelle solution voisine est générée et ainsi de suite jusqu'à ce que le critère d'arrêt du programme soit atteint. Dans le cas contraire, où la modification provoque une augmentation ΔE de la fonction objectif, cette solution voisine peut être acceptée avec une probabilité $e^{-\frac{\Delta E}{T}}$. Par la suite, en conservant la valeur actuelle du paramètre de contrôle, d'autres modifications élémentaires sont effectuées jusqu'à ce que l'équilibre thermodynamique, dans le cas du recuit, soit atteint. (Dréo *et al.*, 2005)

Étant donné que le paramètre de contrôle est supérieur au début de l'algorithme, les solutions non pertinentes, où la valeur de la fonction objectif augmente, ont une forte probabilité d'être acceptées. Cependant, plus le paramètre diminue, plus la probabilité de sélectionner des solutions voisines non pertinentes diminue aussi.

Boctor (1996) et Bouleimen et Lecocq (2003) utilisent la méthode de recuit simulé avec un schéma générateur d'échéanciers en série avec une représentation par liste d'activités alors que Lee et Kim (1996) et Cho et Kim (1997) utilisent avec un schéma générateur d'échéanciers en parallèle basé sur la représentation par clé aléatoire.

La méthode du recuit simulé peut être difficile à régler et les temps de calcul peuvent devenir très importants. Aussi, lorsque la valeur du paramètre de contrôle est

basse, le taux d'acceptation de l'algorithme devient très faible, rendant ainsi la méthode inefficace. Cependant, cette méthode est intéressante puisqu'elle est offre une souplesse d'utilisation étant donné que de nouvelles contraintes peuvent être ajoutées facilement au programme, puisqu'elle est facile à implémenter et puisqu'elle a donné de très bons résultats pour de nombreux problèmes de grande taille. (Dréo *et al.*, 2005)

1.5.3.2 Méthode tabou

La méthode de recherche tabou est une méthode de recherche locale basée sur l'intelligence artificielle. À l'inverse du recuit simulé, cette méthode dispose d'une mémoire lui permettant de tirer des leçons du passé.

Le principe de fonctionnement de la méthode tabou est semblable à celui du recuit simulé, une configuration initiale est établie et elle est actualisée par des itérations successives. La recherche tabou accepte, elle aussi, des solutions pertinentes ainsi que des solutions non pertinentes (Thomas et Salhi, 1998).

Lors de chaque itération, deux étapes sont nécessaires afin de passer d'une configuration *s* à une configuration *t*. D'abord, l'ensemble de solutions voisines de *s* est généré. Il s'agit de l'ensemble des configurations accessibles à partir de *s* avec un seul mouvement élémentaire. La seconde étape consiste à évaluer la fonction objectif du problème pour chacune des solutions de l'ensemble des configurations accessibles. La configuration *t* choisie, succédant à *s*, est celle présentant la plus faible valeur pour la fonction objectif. Cette configuration est acceptée même si elle est supérieure à la configuration actuelle, de la même manière que pour le recuit simulé. (Dréo *et al.*, 2005) Les itérations continuent jusqu'à ce que les critères d'arrêt établis soient atteints.

Afin d'éviter de tourner en boucle, une mémoire à court terme des mouvements récents appelée liste tabou est utilisée. Cette liste est utilisée pour interdire les mouvements voisins qui pourraient annuler l'effet des différents mouvements

sélectionnés et ainsi conduire à une solution récemment visitée (Kolisch et Hartmann, 2006)

Cet algorithme possède donc une mémoire à court terme étant donné qu'il se souvient des solutions visitées récemment. Deux mécanismes supplémentaires, l'intensification et la diversification, sont souvent utilisés afin d'attribuer une mémoire à long terme à l'algorithme. Ceux-ci gardent en mémoire pour une longue période la fréquence de l'occurrence des événements particuliers. L'intensification permet d'approfondir l'exploration de l'espace de solutions dans des régions prometteuses et la diversification permet d'orienter la recherche d'un optimum vers des régions trop rarement visitées (Dréo *et al.*, 2005).

Glover (1989a; 1989b) fut parmi les premiers à explorer la méthode tabou. Lee et Kim (1996), Baar *et al.* (1998), Thomas et Salhi (1998), Kochetov et Stolyar (2003), Valls *et al.* (2003), Artigues *et al.* (2003) et Debels *et al.* (2006) ont aussi étudié cette méthode.

La méthode tabou a donné d'excellents résultats pour certains problèmes d'optimisation en raison de la simplicité de sa forme de base. Elle est facile à utiliser puisqu'elle comporte peu de paramètres de réglage. Cependant, l'utilisation de mécanismes supplémentaires apporte une importante complexité.

1.5.3.3 Algorithmes génétiques

Les algorithmes génétiques font partie des algorithmes évolutionnaires, apparus à la fin des années 1950 (Fraser 1957). Ces derniers d'inspirent de l'évolution biologique des espèces. Parmi plusieurs approches comme celles de Holland (1962), Fogel *et al.* (1966) et Rechenberg (1965), les algorithmes génétiques constituent assurément l'exemple le plus connu, en raison de la parution du livre de D. E. Goldberg, en 1989 (Goldbeg, 1989).

Contrairement aux méthodes de recherche locale qui considèrent une solution unique, les algorithmes génétiques considèrent un ensemble, ou une population, de solutions et combinent des solutions existantes pour en créer de nouvelles (Kolisch et Hartmann, 2006).

Une population initiale, constituée de N points choisis au hasard, subit des générations successives afin de faire évoluer sa composition. Chaque individu de la population possède un certain degré de performance afin de mesurer sa capacité d'adaptation à l'objectif visé, soit la minimisation de la fonction objectif. Les générations sont effectuées à l'aide de deux mécanismes principaux afin d'améliorer la performance globale des individus. La sélection favorise la reproduction et la survie des individus les plus performants et la reproduction permet le brassage, la recombinaison et les variations des caractères héréditaires des parents, dans le but de former des descendants possédant de nouvelles potentialités (Dréo *et al.*, 2005).

Lors de l'utilisation de l'algorithme génétique, un individu est généralement représenté par une liste d'entiers pour un problème combinatoire, un vecteur de nombres réels pour un problème numérique dans un espace continu ou une chaîne de nombres binaires pour un problème booléen (Dréo *et al.*, 2005).

Afin de passer d'une génération à une autre, quatre étapes sont nécessaires, la phase de sélection, la phase de reproduction, la phase d'évaluation des performances et la phase de remplacement.

La phase de sélection détermine les individus qui participent à la reproduction, ceux-ci deviennent alors des parents et sont disponibles pour la phase de reproduction. Le choix est effectué selon la performance de l'individu.

Lors de la phase de reproduction, des copies des individus sélectionnés sont créées afin d'y appliquer des opérateurs de variation pour générer une progéniture. Chaque solution potentielle est représentée par des paramètres, soit des gènes, qui sont réunis afin de former une chaîne de valeurs, soit un chromosome. Les paramètres représentés par un chromosome représentent l'individu. Deux opérateurs sont

généralement utilisés pour la reproduction, le croisement, qui produit un ou deux descendants en accouplant deux parents et la mutation, qui génère un individu à partir d'un seul parent. Donc, à chaque génération, les individus ayant la plus grande force sont sélectionnés afin d'être reproduits pour créer la suivante tandis que les plus faibles sont supprimés (Mendes, Gonçalves et Resende, 2009).

La performance des nouveaux individus est ensuite évaluée à partir des objectifs fixés pour le problème à l'étude. Pour les problèmes d'optimisation, la force d'une solution se base sur la fonction objectif (Kolisch et Hartmann, 2006).

La phase de remplacement consiste finalement à déterminer les individus de la nouvelle génération. Le choix peut être fait, par exemple, en remplaçant, en nombre égal, les individus les moins performants de la génération précédente par les meilleurs individus produits.

Le critère d'arrêt précisé par l'utilisateur détermine le nombre de générations qui seront produites par l'algorithme.

Leon et Ramamoorthy (1995; cité par Kolisch et Hartmann, 2006), Lee et Kim (1996), Hartmann (2002), Kohlmorgen *et al.* (1998), Mendes, Gonçalves et Resende (2009) et Valls *et al.* (2008) font partie des auteurs dont leurs recherches portent sur les algorithmes génétiques. Merkle et Middendorf (2002), Kochetov et Stolyar (2003), et Valls *et al.* (2003; 2004; 2005) utilisent les approches évolutionnaires.

Les algorithmes génétiques sont intéressants puisqu'ils permettent d'obtenir des solutions variées lorsqu'une fonction comporte plusieurs optimums globaux. Donc, lors de la résolution d'un problème d'optimisation comportant plus d'un objectif, ils permettent de fournir des solutions de compromis (Dréo *et al.*, 2005).

1.5.3.4 Algorithmes de colonies de fourmis

Les algorithmes de colonies de fournis, introduits par Colorni, Dorigo et Maniezzo (1992), simulent la capacité de résolution de problème en collectivité pouvant être observée chez une colonie de fourmis. Chaque fourmi est dotée de facultés très limitées, cependant, collectivement, cette espèce a du succès.

Dorigo *et al.* (1992) s'inspirent du fait que les fourmis sont en mesure de retrouver rapidement le plus court chemin pour se déplacer lorsqu'elles travaillent collectivement afin de proposer un nouvel algorithme pour la résolution du problème du voyageur de commerce. Ce problème consiste à trouver le trajet le plus court reliant un certain nombre de villes, où chaque ville ne peut être visitée qu'une seule fois. Il peut être représenté par un graphique semblable à celui d'un graphique représentant les relations de précédence entre les activités d'un projet, où les nœuds représentent les villes et les arcs représentent les trajets entre ces villes.

Dorigo *et al.* (1996), Gambardella *et al.* (1995), Dorigo *et al.* (1997a; 1997b), Stützle *et al.* (1997), et Stützle *et al.* (2000) proposent des modifications à l'algorithme de base.

Les algorithmes de fourmis sont intéressants puisqu'ils possèdent plusieurs caractéristiques telles que le parallélisme intrinsèque élevé, la flexibilité, qui permet à l'algorithme de s'adapter à des modifications de l'environnement, la robustesse, permettant à l'algorithme de maintenir son activité même si certains éléments sont défaillants, la décentralisation et l'auto-organisation, qui permet à l'algorithme de trouver une solution, inconnue au départ (Dréo *et al.*, 2005).

Les algorithmes de colonies de fourmis peuvent être utilisés pour des problèmes distribués par nature, qui évoluent de manière dynamique ou qui nécessitent une forte tolérance aux pannes. Cependant, la transposition d'un tel algorithme à un problème d'optimisation ne va pas de soi, elle doit être traitée spécifiquement, ce qui peut parfois être ardu (Dréo *et al.*, 2005).

1.6 Extensions du RCPSP

Des auteurs ont étudié des extensions du problème de base : les problèmes à modes multiples, à projets multiples, à critères multiples et à compétences multiples. Ces extensions sont présentées dans cette section.

1.6.1 Modes multiples

Lors de la résolution du RCPSP, les ressources considérées sont renouvelables, soit limitées par unité de temps. Cependant, deux autres types de ressources se distinguent dans la littérature, les ressources non renouvelables, limitées pour le projet entier et les ressources doublement contraintes, limitées par période de temps ainsi que pour la durée totale du projet (Hartmann et Briskorn, 2010; cité par Montoya, 2012). Ces trois types de ressources peuvent être considérés pour le RCPSP à modes multiples. Pour cette extension du problème de base, chaque activité est exécutée selon un mode sélectionné parmi plusieurs. Chaque mode représente la relation entre les besoins en ressources et la durée de chaque activité. Donc, chaque activité exécutée selon le mode sélectionné a une durée déterminée pour laquelle une quantité de ressources renouvelables est utilisée et une quantité de ressources non renouvelables est consommée. L'objectif est de trouver une combinaison de modes afin de minimiser la durée totale du projet.

Hartmann et Drexl (1997) comparent les méthodes arborescentes disponibles pour résoudre le RCPSP à modes multiples, Sprecher, Hartmann et Drexl (1994) présentent une extension de la méthode arborescente de Demeulemeester et Herroelen (1992), Zhu, Bard et Yu (2006) présentent une procédure de branchement et Sprecher (2000) étudie les méthodes exactes pour cette extension. Bouleimen et Lecocq (2003) présentent la version multimode de leur algorithme de recuit simulé et Jozefowska *et al.* (2001) présentent

aussi un algorithme de recuit simulé. Vartouni et Khanli (2014), Hartmann (2002), Lova *et al.* (2000) ainsi que Peteghem et Vanhoucke (2010) présentent des algorithmes génétiques pour le RCPSP à modes multiples.

1.6.2 Projets multiples

Le RCPSP à projets multiples implique, contrairement au problème de base où les activités d'un unique projet sont planifiées, la planification de plusieurs projets simultanément. La planification à projets multiples est plus complexe étant donné que les ressources disponibles sont affectées aux activités de tous les projets, selon les priorités. L'objectif est donc de générer un échéancier des activités pour chaque projet en tenant compte des disponibilités des ressources et en respectant les contraintes de précédence afin de minimiser la durée des projets.

Drexl (1997) utilise une méthode arborescente pour résoudre le problème et Gonçalves, Mendes et Resende (2007), Kim *et al.* (2005), Kumanan, Jose et Raja (2006) ainsi que Damak *et al.* (2009) présentent des algorithmes génétiques. Mendes (2003) présente un algorithme génétique basé sur la représentation aléatoire ainsi qu'un schéma générateur d'échéanciers en parallèle modifié. Pritsker, Watters et Wolfe (1969) utilisent la programmation linéaire, Vercellis (1994) utilise la technique de décomposition de Lagrange et Singh (2013) présente un algorithme hybride qui intègre la priorité des projets. Dalfard et Ranjbar (2012) combinent l'algorithme de recuit simulé avec les règles de priorités et Fendley (1968), Kurtulus et Davis (1982) et Lova *et al.* (2000) basent leurs modèles sur les règles de priorité.

1.6.3 Critères multiples

Le RCPSP est régulièrement utilisé afin d'optimiser un critère, généralement la durée du projet. Cependant, afin d'évaluer la faisabilité des solutions de projets réels,

plusieurs chercheurs orientent leurs recherches sur l'optimisation de deux critères et plus, soit l'optimisation à critères multiples. Ces derniers peuvent être reliés au temps, aux coûts, aux ressources, ou simplement aux éléments visés par la recherche de l'auteur.

Kreszowska (2009) utilise un algorithme génétique afin d'optimiser le coût, le temps et les ressources. Leu et Yang (1999) présentent un algorithme génétique pour minimiser le temps et le coût d'un projet. Cai et Li (2000) proposent un algorithme génétique visant à minimiser le coût des ressources utilisées pour répondre aux demandes de main-d'œuvre de projet, minimiser la variation de la quantité de ressources utilisées durant le projet et maximiser l'utilisation des ressources parmi les solutions proposant un même niveau de coût. Viana et de Souza (2000) présentent des algorithmes de recuit simulé et tabou afin de minimiser trois objectifs, la durée totale du projet, le retard moyen des activités et la somme des violations des disponibilités des ressources. Gomar *et al.* (2000) développent un modèle de programmation linéaire visant à optimiser le processus d'affectation des ressources. Ils considèrent quatre objectifs, la minimisation de la quantité de ressources totales utilisées au cours du projet, la maximisation de l'utilisation des ressources lors de la phase de construction du projet, la minimisation des embauches et des congédiements.

1.6.4 Compétences multiples

Lors de la résolution du RCPSP, chaque activité requiert l'exécution d'une compétence afin d'être complétée. Cependant, pour le RCPSP à compétences multiples, chaque activité nécessite l'exécution d'une ou plusieurs compétences. La quantité de ressources nécessaires pour chaque compétence et pour chaque activité d'un projet est connue et chaque ressource humaine possède un ensemble de compétences lui permettant de participer à plus d'une activité lors de l'exécution du projet. À chaque instant, une ressource peut participer à l'exécution d'une seule activité avec une seule compétence et elle est affectée à une activité pour toute la durée de son exécution.

Certa *et al.* (2009) étudient un contexte à objectifs, projets et compétences multiples. Ils proposent une méthode multiobjectif afin d'allouer différentes ressources possédant différents niveaux de compétences aux activités des projets qui en nécessitent plusieurs. Le modèle qu'ils développent se concentre sur trois aspects fondamentaux des ressources humaines, soit le différent niveau de compétence, le processus d'apprentissage entrepris par chaque ressource et les relations sociales entre les équipes de travail.

Bellenguez et Néron (2005), Bellenguez-Morineau (2008) et Bellenguez-Morineau et Néron (2007) étudient le problème en supposant que certaines ressources sont indisponibles sur certaines périodes. Firat et Hurkens (2012) étudient le problème de charge de travail associée aux projets à compétences multiples. Différents niveaux d'efficacité sont considérés pour chaque compétence et chaque activité nécessite des ressources possédant un niveau particulier d'une compétence donnée. Leur objectif est de déterminer la charge de travail de chaque équipe afin de maximiser le nombre d'activités traitées dans une journée. Correia, Lampreia-Lourenço et Saldanha-da-Gama (2012) cherchent à minimiser la durée totale du projet où chaque activité peut nécessiter plusieurs compétences. Pour chaque compétence, plus d'une ressource peut être nécessaire.

Belenguez et Néron (2005) adaptent deux bornes inférieures utilisées dans les méthodes arborescentes et visent à minimiser la durée du projet. Cai et Li (2000), bien qu'ils visent à optimiser plusieurs critères avec un algorithme génétique, considèrent aussi que les activités nécessitent différentes compétences et que les ressources possèdent une ou plusieurs d'entres elles. Koshijima et Umeda (2001) développent une programmation linéaire afin d'allouer les ressources avec différents niveaux de compétence aux activités dans le but de minimiser les coûts d'exécution d'un projet, ainsi que la minimisation du nombre de changements de ressources entre les équipes. Correia et Saldanha-da-Gama (2014) développent une programmation linéaire afin de minimiser les coûts variables et fixes d'un projet à compétences multiples.

Montoya (2012) propose plusieurs méthodes pour résoudre le RCPSP à compétences multiples, en accordant une importance particulière aux méthodes exactes et

avec le principal objectif d'obtenir des solutions optimales pour des problèmes de petites et moyennes tailles. Afin de représenter le problème sous différentes perspectives, Montoya introduit cinq modèles de programmation linéaire. Le modèle du temps indexé, initialement présenté par Bellenguez-Morineau et Néron (2007), considère une variable de décision binaire et une contrainte linéaire, utilisée afin d'éviter les conflits de ressources. Le modèle du temps indexé avec temps de départ, introduit par Pritsker, Watters et Wolfe (1969), ajoute une variable binaire au modèle précédent afin de représenter le temps de départ des activités. Le modèle du temps indexé avec temps de départ modifié, introduit par Christofides, Alvarez-Valdés et Tamarit (1987; cité par Montoya, 2012), est semblable au modèle précédent, mais les relations de précédence sont représentées de manière décomposée. Une contrainte linéaire est utilisée pour représenter les relations de précédence entre les activités à chaque période de temps. Le modèle de la séquence indexée se concentre sur l'ordre dans lequel chaque ressource réalise chaque activité où elle est affectée. Ce modèle se base sur la formulation proposée par Kesen, Das et Güngör (2010). Le dernier modèle se base sur le flux et les variables de décision principales sont utilisées afin de déterminer si une activité est planifiée avant une autre. Ce modèle se base sur la formulation mathématique du problème des tournées des véhicules, un problème d'optimisation combinatoire (Toth et Vigo, 2002).

1.7 Objectif de la recherche

En résumé, on retrouve, dans la littérature, trois types de méthodes permettant de résoudre le RCPSP. Tout d’abord, les méthodes exactes qui sont utilisées pour des problèmes de taille modérée afin d'obtenir une solution exacte au problème et pour des problèmes de plus grande taille afin d'obtenir une solution approchée de la solution exacte dans un temps raisonnable. Puis, les heuristiques et les métaheuristiques qui sont utilisées afin de générer une bonne solution dans un délai raisonnable.

Des extensions du problème de base telles que le problème à modes multiples, le problème à projets multiples, le problème à critères multiples et le problème à compétences multiples ont été développées.

Tel que mentionné à la section 1.7, certains chercheurs ont développé des modèles permettant de résoudre le problème à compétences multiples alors que d'autres ont développé des modèles permettant de résoudre le problème multiobjectif. Cependant, très peu d'entre eux, comme Certa *et al.* (2009), ont étudié ces deux extensions simultanément, ce pourquoi cette recherche s'oriente en ce sens.

L'objectif de cette recherche consiste donc à proposer une méthode de résolution à critères multiples pour le RCPSP prenant en considération les compétences maîtrisées par les ressources et les compétences requises par chaque activité.

CHAPITRE 2
DESCRIPTION DU MODÈLE ET MÉTHODE DE RÉSOLUTION

Tel que mentionné à la section 1.8, l'objectif de cette recherche consiste à développer un modèle multiobjectif permettant de résoudre le problème d'ordonnancement avec contraintes de ressources à compétences multiples.

Le modèle développé doit d'abord respecter les principes de base du RCPSP présentés à la section 1.1. Le projet est constitué d'activités, de ressources et de compétences, où les ressources maîtrisent une ou plusieurs compétences et chaque activité du projet nécessite une ou plusieurs compétences. Dans ce contexte, les activités peuvent être réalisées simultanément. Ce modèle permet d'abord d'obtenir des solutions Pareto optimales selon deux des critères les plus importants en gestion de projet, le temps et le coût, sans accorder de préférence à l'un ou à l'autre d'entre eux. Par la suite, les solutions optimales sont évaluées à l'aide d'une méthode à critères multiples, où l'utilisateur est en mesure de choisir les critères selon lesquels l'optimisation se base et où il peut accorder une importance à chacun d'entre eux, selon ses préférences.

Donc, le modèle développé affecte les ressources aux activités selon leurs besoins en compétences, en respectant les contraintes de précédence et de ressources, afin de minimiser le temps, le coût ainsi qu'un ou plusieurs autres critères. Pour cette étude, le critère supplémentaire est le temps perdu, soit le nombre d'unités de temps dans la planification obtenue où une ressource est inactive entre deux activités.

Cette section présente la nomenclature utilisée pour le modèle de cette recherche, les équations et inéquations utilisées pour représenter le modèle, la méthode de résolution ainsi qu'un exemple numérique.

2.1 Nomenclature du problème

Le modèle étudié dans cette recherche se base sur l'un des modèles de programmation linéaire présentés dans Montoya (2012), soit le modèle du temps indexé. Ce modèle de départ, proposé par Bellenguez-Morineau et Néron (2005), se base sur la notion du temps indexé considérée par Pritsker, Watters et Wolfe (1969) et est adapté afin d'être utilisé dans un contexte d'objectifs multiples.

Le modèle développé se compose de paramètres et de variables, le tableau 2.1 présente les paramètres utilisés et leur description et le tableau 2.2 présente les variables et leur description.

Tableau 2.1 : Paramètres du modèle à compétences multiples

Paramètres	**Description**
i	Activité, où $i \in A$
m	Ressource, où $m \in W$
k	Compétence, où $k \in S$
es_i	Temps de départ au plus tôt de l'activité A_i
ls_i	Temps de départ au plus tard de l'activité A_i
p_i	Durée de l'activité A_i
T	Horizon temporel du projet
t	Temps donné, où $t \in [0, T]$
b_k^i	Nombre de ressource(s) nécessaire(s) possédant la compétence k pour réaliser la tâche i
MS_m^k	Compétence k maîtrisée par la ressource m, égal à 1 si la ressource m possède la compétence k
$C_{m,k}$	Coût d'utilisation d'une ressource m pour une compétence k pour une unité de temps

Tableau 2.2 : Variables du modèle à compétences multiples

Variables	Description
$x_{i.m}^{t}$	Variable binaire égale à 1 si la ressource W_m débute l'activité A_i au temps t, 0 sinon
$y_{i,m}^{k}$	Variable binaire égale à 1 si la ressource W_m utilise la compétence S_k pour l'activité A_i au temps t, 0 sinon
t_i	Temps de départ de l'activité A_i
$Coût$	Coût total d'utilisation des ressources au cours du projet, déterminé par les compétences k utilisées par les ressources m pour les activités i

2.2 Formulation mathématique du problème

Afin de faciliter la compréhension du modèle, le temps de départ de l'activité A_i est déterminé par :

$$t_i = \frac{\sum_{m \in W} \sum_{t \in [0,T]} \left(x_{i,m}^{t} \cdot t\right)}{\sum_{k \in S} b_i^k} \quad \forall i \in A \tag{2.1}$$

Par conséquent, la formulation mathématique linéaire en nombres entiers à deux critères associée au modèle peut être définie par :

$$Z[TIM]: Min\ C_{max} = t_N \tag{2.2}$$

$$Z[TIM]: Min\ Coût \tag{2.3}$$

Tel que

$$t_i + p_i \leq t_j \quad \forall i \in A, \forall j \in E_i^+ \tag{2.4}$$

$$es_i \leq t_i \leq ls_i \quad \forall i\ A \tag{2.5}$$

$$\sum_{t\in[0,T]} x_{i,m}^t \leq 1 \quad \forall i \in A, \forall m \in W \tag{2.6}$$

$$\sum_{i\in A} \sum_{d\in[t-p_i+1,t]} x_{i,m}^d \leq 1 \quad \forall t \in [0,T]\ \forall m \in W \tag{2.7}$$

$$\sum_{t\in[0,T]} (x_{i,m}^t \cdot t) \leq \frac{\sum_{h\in W}\sum_{t\in[0,T]}(x_{i,h}^t \cdot t)}{\sum_{k\in S} b_i^k} \quad \forall i \in A, \forall m \in W \tag{2.8}$$

$$y_{i,m}^k \leq MS_m^k \quad \forall i \in A, \forall m \in W, \forall k \in S \tag{2.9}$$

$$\sum_{m\in W} y_{i,m}^k = b_i^k \quad \forall i \in A, \forall k \in S \tag{2.10}$$

$$\sum_{t\in[0,T]} x_{i,m}^t = \sum_{k\in S} y_{i,m}^k \quad \forall i \in A, \forall m \in W \tag{2.11}$$

$$Coût = \sum_{k\in S}\sum_{m\in W} \left(y_{i,m}^k \cdot p_i \cdot C_{m,k}\right) \quad \forall i \in A \tag{2.12}$$

$$x_{i,m}^t \in \{0,1\} \quad \forall i \in A, \forall m \in W, \forall t \in [0,T] \tag{2.13}$$

$$y_{i,m}^k \in \{0,1\} \quad \forall i \in A, \forall m \in W, \forall k \in S \tag{2.14}$$

Les objectifs de ce modèle consistent à minimiser la durée totale du projet, définie par le temps de départ de l'activité A_N (2.2) et à minimiser le coût d'utilisation de ses ressources (2.3). La contrainte (2.4) représente les relations de précédence entre les activités, impliquant que le temps de fin d'une activité A_i est égal ou inférieur au temps de départ de son successeur E_i^+. La contrainte (2.5) permet de s'assurer que le temps de départ de chaque activité se trouve dans une fenêtre de temps définie par le début au plus tôt et le début au plus tard de chaque activité. La contrainte (2.6) permet de s'assurer qu'une ressource ne débute une activité A_i qu'une seule fois durant l'exécution du projet. La contrainte (2.7) permet de s'assurer qu'une ressource n'exécute qu'une activité à la fois.

La contrainte (2.8) permet de s'assurer que les ressources assignées à une activité spécifique travaillent simultanément durant toute la durée de celle-ci. La contrainte (2.9) garantit qu'une ressource peut seulement exercer une compétence qu'elle maîtrise. La contrainte (2.10) s'assure que la planification respecte les demandes de compétences pour chacune des activités. La contrainte (2.11) permet de s'assurer qu'une ressource n'utilise qu'une seule compétence pour une activité à laquelle elle est assignée. L'équation (2.12) permet de calculer le coût d'utilisation des ressources pour l'exécution du projet, selon le coût d'utilisation associé à chaque ressource pour une compétence donnée pour une unité de temps et selon la durée de chaque activité. Puis, les contraintes (2.13) et (2.14) représentent les variables de décision binaires.

La formulation mathématique linéaire en nombres entiers de ce modèle permet d'optimiser deux critères, le temps et le coût d'un projet.

2.3 Méthode de résolution

La méthode de résolution se présente en cinq étapes, elles sont présentées dans l'organigramme de la figure 2.1.

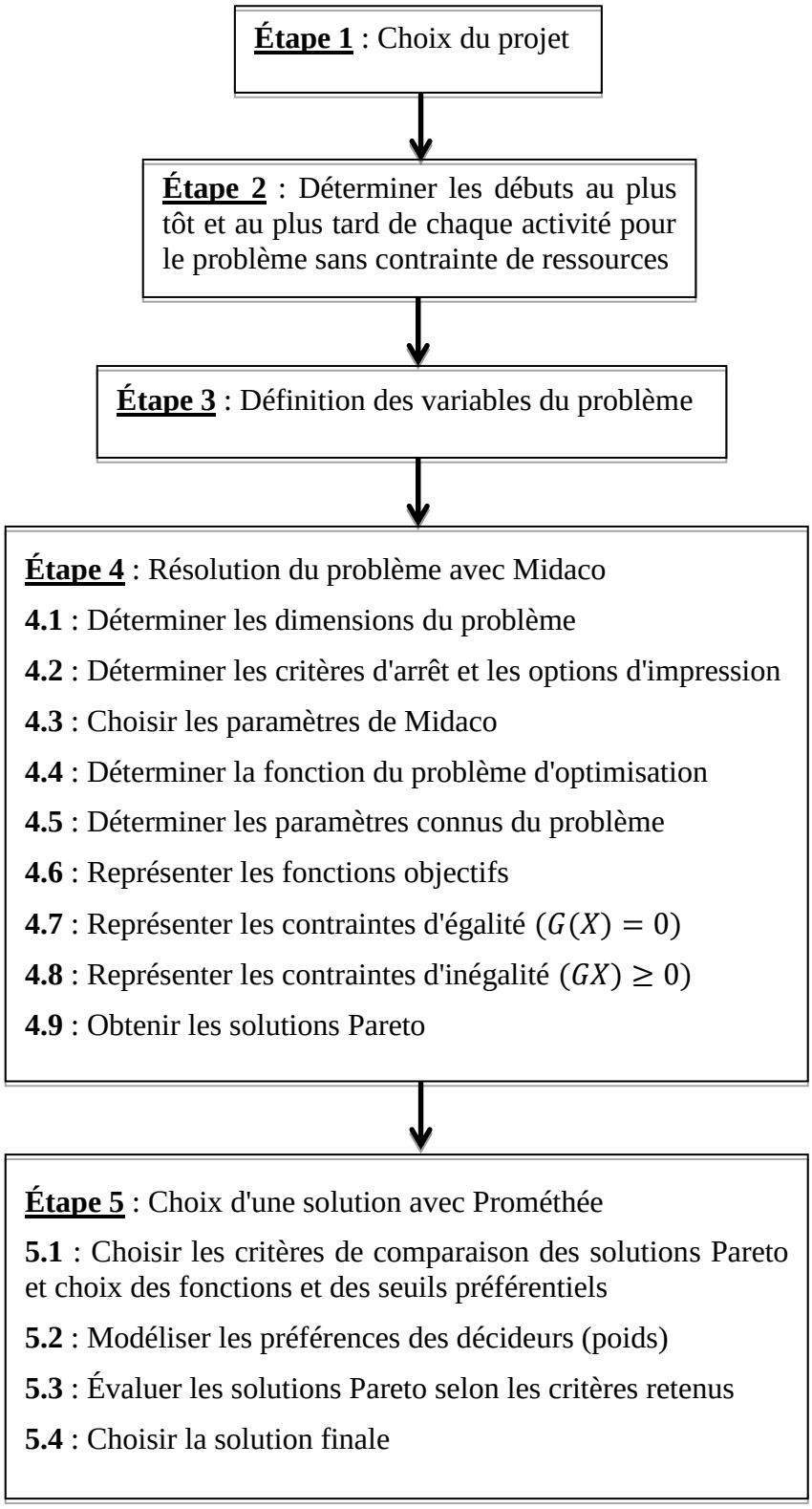

Figure 2.1 : Méthode de résolution du problème

Les logiciels Midaco et Prométhée et leur fonctionnement sont présentés.

2.3.1 Recherche de solutions Pareto avec Midaco

La recherche de solutions Pareto s'effectue avec Midaco, un logiciel permettant de générer un sous-ensemble de solutions satisfaisant toutes les contraintes du projet. Les informations présentées dans les sections 2.3.1.1 sont tirées du manuel de l'utilisateur de Midaco (Midaco-Solver, 2018).

2.3.1.1 Présentation du logiciel d'optimisation Midaco

Midaco est un logiciel numérique haute performance utilisé pour l'optimisation globale de problèmes à objectifs uniques ou multiples. Le logiciel fonctionne avec plusieurs types de variables, les variables continues, les variables entières, mixtes ainsi que continues, entières et mixtes avec contraintes d'égalité et d'inégalité. Les problèmes étudiés peuvent contenir comporter un ou plusieurs objectifs.

Le fonctionnement de Midaco se base sur un algorithme évolutif hybride, composé de la méthode d'optimisation métaheuristique des colonies de fourmis et d'une ligne de recherche de retour arrière. Midaco peut résoudre des problèmes composés de plus de 100 000 variables et peut gérer des milliers de contraintes et des centaines d'objectifs. Ce logiciel fonctionne avec une variété de langages, comme Excel, VBA, Java, C#, R, Matlab, Octave, Python, Julia, C/C++ et Fortran. Dans le cas de cette recherche, le problème est étudié avec Matlab et est à objectifs multiples.

L'algorithme de Midaco considère le problème d'optimisation comme une boîte noire retournant, pour des variables d'entrée X, les valeurs de la fonction objective correspondante F(X) et les valeurs des fonctions de contraintes G(X).

En ce qui concerne l'optimisation à objectifs multiples, le résultat ne présente pas une solution unique qui optimise simultanément chaque objectif, contrairement à l'optimisation à un objectif, où une seule solution existe comme optimum global. La solution obtenue est plutôt représentée par un ensemble de solutions non dominées, aussi appelé solutions optimales de Pareto, formant ainsi une courbe de compromis, aussi appelée frontière de Pareto, entre les objectifs individuels. Cette courbe représente les meilleures solutions obtenues selon tous les objectifs. Par exemple, pour un problème à deux objectifs, une solution obtenant de meilleurs résultats qu'une autre solution pour les deux objectifs sera non dominée, donc optimale. Si deux solutions obtiennent le même résultat pour un critère, mais que l'une est supérieure à l'autre pour le second critère, les deux solutions seront Pareto optimales.

2.3.1.1.1 Présentation du problème d'optimisation avec Midaco

La première étape de la présentation du problème d'optimisation avec Midaco consiste à définir les dimensions du problème, ses limites et le point de départ du vecteur des variables de décision X. Pour les dimensions, le nombre d'objectifs, le nombre total de variables, le nombre de variables entières, le nombre total de contraintes et le nombre de contraintes d'égalité sont pris en considération, dans cet ordre. Les limites inférieure et supérieure de chaque variable sont indiquées et le point de départ de vecteur des variables de décision, généralement égal aux limites inférieures du problème, est déterminé.

La seconde étape consiste à choisir les critères d'arrêt, tels que le nombre maximal d'évaluations de la fonction et le temps limite d'évaluation et les options d'impression, soit la fréquence à laquelle la meilleure solution actuelle est imprimée à l'écran et le choix de créer un fichier de sortie ou non.

Les utilisateurs avancés peuvent alors choisir les paramètres Midaco afin de personnaliser les performances et le comportement de l'algorithme. La valeur par défaut

des 13 paramètres est égale à zéro. Les paramètres modifiés dans cette recherche sont présentés à la section 2.3.1.1.2.

La fonction du problème d'optimisation est ensuite inscrite dans le langage de programmation approprié et elle ne prend en considération que les trois vecteurs mentionnés ci-haut, le vecteur des variables de décision X, le vecteur des fonctions objectif F(X), et le vecteur des contraintes G(X). En ce qui concerne Matlab, elle s'écrit sous la forme suivante : $[f, g] = problem\_function(x)$.

Puis, la ou les fonctions objectifs, les contraintes d'égalité et les contraintes d'inégalité sont ajoutées, dans cet ordre.

2.3.1.1.2 Présentation des paramètres de Midaco

Lors de l'utilisation de Midaco, 13 paramètres peuvent être modifiés par l'utilisateur afin de personnaliser l'algorithme. Ils sont tous présentés et les paramètres modifiés dans cette étude sont présentés avec des détails supplémentaires.

Le paramètre «ACCURACY» définit la tolérance de précision pour la violation de contrainte. Les paramètres «FSTOP», «ALGOSTOP» et «EVALSTOP» activent un critère d'arrêt pour Midaco. Le premier est basé sur la valeur à atteindre pour la fonction objectif, le est basé sur le processus algorithmique du logiciel et le dernier est basé sur le processus algorithmique de Midaco en tenant compte du nombre d'évaluations de la fonction. Le paramètre «FOCUS» force Midaco à concentrer son processus de recherche sur la meilleure solution actuelle pour la rendre plus locale. Ce paramètre peut entraîner une vitesse de convergence plus rapide et peut affiner les solutions. Le paramètre «ORACLE» spécifie la fonction de pénalité. Ce paramètre n'est pertinent que pour les problèmes contraints où l'utilisateur connaît certains éléments de base sur le problème. Le paramètre «PARETOMAX» définit le nombre maximal de solutions non dominées sauvegardées par Midaco. Le paramètre «BALANCE» est généralement utilisé pour les problèmes à objectifs multiples puisqu'il définit la zone de la frontière de Pareto sur

laquelle Midaco devrait concentrer sa recherche principale. Le paramètre «CHARACTER» permet d'activer les paramètres internes de Midaco, soit ceux pour les problèmes de type continu, les problèmes de type combinatoire et les situations où les problèmes sont tous différents.

Le paramètre «SEED» définit la progéniture initiale du générateur de nombres pseudo-aléatoires interne de Midaco. La progéniture est utilisée pour déterminer la séquence des nombres pseudo-aléatoires échantillonnés par le générateur. La valeur de ce paramètre doit être un entier supérieur ou égal à zéro et changer cette valeur permet à l'algorithme de reproduire les passages prometteurs. L'impact de ce paramètre varie proportionnellement avec la complexité du problème. Pour les problèmes difficiles, il est plus prometteur d'exécuter plusieurs petites séries de Midaco avec des progénitures aléatoires différentes plutôt que d'effectuer une seule longue course.

Le paramètre «ANTS» permet de fixer le nombre de fourmis, soit le nombre d'itérations, que Midaco produit en une génération. Ce paramètre doit être utilisé en combinaison avec le paramètre «KERNEL» et l'utilisation de ceux-ci peut être prometteuse pour les problèmes à grande échelle ou avec un temps de calcul élevé.

Le paramètre «KERNEL» permet de fixer le nombre de noyaux dans les fonctions gaussiennes de probabilité de densité multinoyaux de Midaco. Ces fonctions génèrent des échantillons d'itérations, aussi appelés fourmis ou individus et la taille du noyau correspond au nombre de solutions sauvegardées dans les archives de Midaco. Pour un problème convexe, un nombre de noyaux faible entraîne une convergence plus rapide, mais augmente les risques que Midaco reste bloqué dans un optimal local.

Le paramètre «EPSILON» définit la précision du filtre de dominance de Pareto à objectifs multiples utilisé par l'algorithme. Plus la valeur du paramètre est fiable, plus les chances qu'une nouvelle solution soit introduite dans la frontière de Pareto sont élevées. La valeur choisie peut exercer une influence importante sur la quantité de points stockés par Midaco et sur le temps de calcul interne.

2.3.2 Sélection d'une solution du meilleur compromis avec Prométhée

La présentation du logiciel Prométhée de cette section repose fortement sur celle réalisée par Naoum (2010).

Le logiciel Prométhée est une méthode d'agrégation partielle répondant à la problématique de rangement des meilleurs projets. La méthode Prométhée est simple d'utilisation, elle exige peu d'information et elle respecte les principes de l'intransitivité et de l'incomparabilité. Roy (1985) et Maystre *et al*. (1994) l'appellent «approche du surclassement de synthèse acceptant l'incomparabilité» ou «méthodes de surclassement» (Vincke, 1989).

La méthode Prométhée de Brans et Vincke (1985) relève de la problématique de rangement P(γ). Elle consiste à poser le problème en termes de rangement des actions, c'est-à-dire à orienter l'investigation vers la mise en évidence d'un classement défini selon l'ensemble des actions (A). Cette problématique prépare une forme de recommandation visant à indiquer un ordre partiel ou complet portant sur des actions jugées équivalentes.

La méthode Prométhée vise à construire une relation de surclassement valuée en s'appuyant sur la comparaison des actions deux à deux dans le but de les ranger de la meilleure à la moins bonne.

2.3.2.1 Construction de la relation de surclassement valuée

Pour chaque critère (j), un poids w_j proportionnel à son importance relative est disponible. Pour chaque couple (a,b) d'actions de l'ensemble des actions A, le degré de surclassement de l'action a sur l'action b est calculé par $\pi(a,b) = \sum_{j=1}^{k} P_j(a,b)w_j$, où P_j(a,b) est un nombre compris entre 1 et 0. Ce dernier est d'autant plus grand que $g_j(a) - g_j(b)$ est grand et nul si $g_j(a) \leq g_j(b)$. Concrètement, P_j(a,b) est calculé de la manière

suivante : $P_j(a,b) = F_j[d_j(a,b)]$, où $d_j(a,b) = g_j(a) - g_j(b)$. Pour estimer les P_j(a,b), le décideur peut choisir, pour chaque critère, une des six formes de courbes représentées dans la figure 2.2.

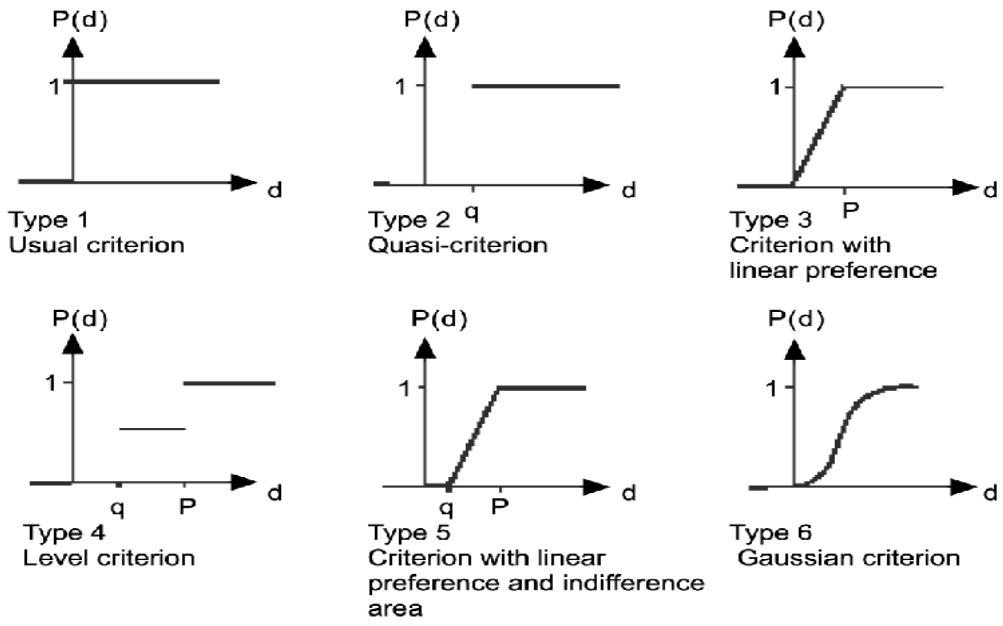

Figure 2.2 : Fonction de préférence de Prométhée (Gul *et al.*, 2017)

En fonction de la manière dont sa préférence croît avec l'écart $g_j(a) - g_j(b)$, le décideur fixe, pour chaque critère, la forme de Pj et le ou les paramètres associés. Les paramètres à estimer ont une interprétation simple puisque ce sont des seuils d'indifférence q et de préférence p. Pour la forme gaussienne, le paramètre à estimer est l'écart-type.

2.3.2.2 Exploitation de la relation de surclassement valuée

Deux préordres totaux peuvent être établis à partir du calcul de la matrice représentant la relation de surclassement. Le premier consiste à ranger les actions dans l'ordre décroissant des nombres $\phi^+(a)$ *tels que* $\phi^+(a) = \sum_{b \in A} \pi\,(a,b)$, soit le flux sortant et le second, dans l'ordre croissant des nombres $\phi^-(a)$ tels que $\phi^-(a) = \sum_{b \in A} \pi(b,a)$, soit le flux rentrant.

Prométhée I établit son rangement en cherchant l'intersection de ces deux préordres totaux afin d'obtenir un préordre partiel. Par définition, les trois critères suivants sont utilisés :

- l'action a surclasse b si $\phi^+(a) > \phi^+(b)$ et $\phi^-(a) < \phi^-(b)$, si $\phi^+(a) > \phi^+(b)$ et $\phi^-(a) = \phi^-(b)$ ou si $\phi^+(a) = \phi^+(b)$ et $\phi^-(a) < \phi^-(b)$;

- l'action a est indifférente à b si $\phi^+(a) = \phi^+(b)$ et $\phi^-(a) = \phi^-(b)$;

- l'action a est incomparable à b si $\phi^+(a) > \phi^+(b)$ et $\phi^-(a) > \phi^-(b)$ ou si $\phi^+(a) < \phi^+(b)$ et $\phi^-(a) < \phi^-(b)$.

Quant à la méthode Prométhée II, elle propose un préordre total des actions basé sur l'ordre décroissant des flux nets $\phi(a)$ tels que $\phi(a) = \phi^+(a) - \phi^-(a)$. Dans ce cas, un rangement complet, soit sans incomparabilité, est généré.

2.3.2.3 Les applications de Prométhée

La méthode Prométhée a été appliquée avec succès dans des disciplines variées dont notamment la gestion hospitalière (D'Avignon et Mareschal, 1989), la localisation d'usines (Mladineo *et al*., 1987) et l'allocation budgétaire (Struys et Pastijn,1988). Les méthodes ont été utilisées comme outil d'aide à la décision multicritère pour sélectionner et classer les projets (Brans, Vincke et Mareschal, 1986) et gérer les différentes

problématiques de sélection de projets (Al-Rash-dan *et al*., 1999; Goumas et Lygerou, 2000). D'autres auteurs ont exploité les méthodes Prométhée dans un contexte stochastique (Marinoni *et al*., 2005) ou dans un contexte où les informations sont incertaines (Wang et Lin, 2003), en les associant avec les méthodes de programmation mathématique (Fernandez-Castro et Jimenez, 2005), en utilisant les nombres flous (Goumas et Lygerou, 2000; Radojevic et Petrovic, 1997; Srinivasa, 1995) et avec intervalles (Geldermann *et al*., 2000; Al-Rashdan *et al*., 1999; Le Teno et Mareschal, 1998). De nombreuses autres applications plus récentes se retrouvent dans la section «ressources» du site internet de Prométhée, comme la figure 2.3 le prouve (Mareschal, 2018).

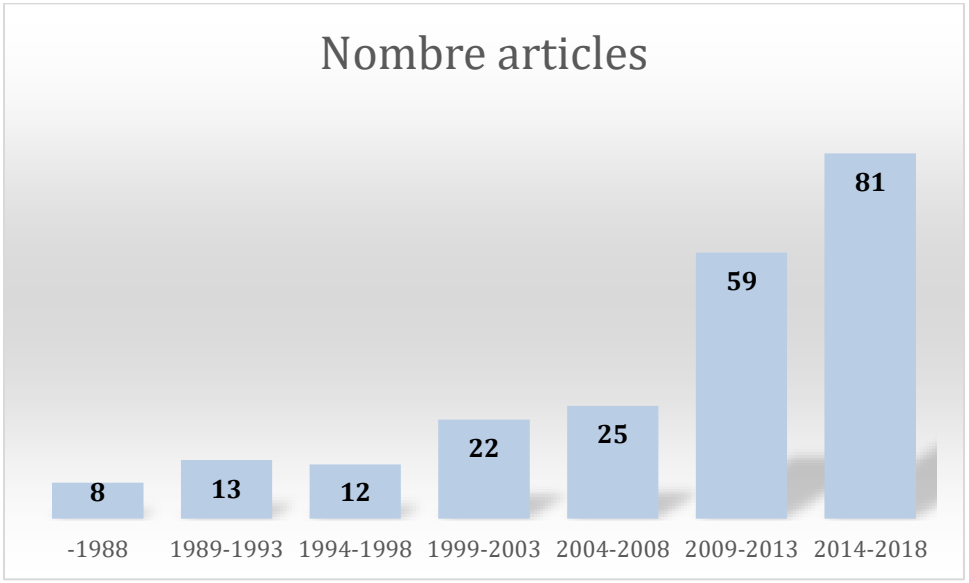

Figure 2.3 : Nombres d'articles scientifiques sur la méthode Promethee (ABI-Inform)

2.4 Illustration de la méthode proposée

Les étapes présentées dans l'organigramme de la figure 2.1 sont reprises avec un exemple numérique. Il est important de noter que l'exemple est utilisé afin d'illustrer la méthode proposée, cependant, il s'agit d'une méthode générale pouvant s'appliquer à

divers types de problèmes pour résolution à critères et à compétences multiples. Les critères étudiés peuvent différer d'un problème à l'autre. Il s'agit d'adapter la méthode à son propre problème.

2.4.1 Choix du projet

Le modèle étudié dans cette recherche se base sur l'un des modèles de programmation linéaire présentés dans Montoya (2012), soit le modèle du temps indexé. Ce dernier, initialement proposé par Bellenguez-Morineau et Néron (2005), se base sur la notion du temps indexé considérée par Pritsker *et al*. (1969) et est adapté afin d'être utilisé dans un contexte de compétences multiples.

Le projet considéré représente un exemple réel de développement de produits logiciels. Ce problème de petite taille est composé de trois activités, quatre ressources et trois compétences et les ressources sont uniquement renouvelables.

Ce problème comporte un ensemble W de M ressources et un ensemble S de K compétences. Chaque ressource W_m ($W_m \in W$) possède un ensemble donné de compétences. La distribution des compétences pour le problème est représentée par le paramètre $MS_{m,k}$, où il est égal à un si la ressource W_m possède la compétence S_k ou à zéro dans le cas contraire. Une ressource ne peut utiliser qu'une seule compétence pour une seule activité à chaque instant t donné. Une quantité totale de $b_{i,k}$ ressources possédant chaque compétence S_k peut être assignée à l'activité A_i. Si la compétence S_k est requise par l'activité A_i, le paramètre $b_{i,k}$ est égal à un, dans le cas contraire, sa valeur est nulle. Puis, toutes les ressources assignées à une activité A_i doivent travailler simultanément durant la durée totale de celle-ci.

Les tableaux qui suivent présentent les caractéristiques du projet. Le tableau 2.3 présente les compétences requises durant le projet, le tableau 2.4 présente les compétences nécessaires pour chacune des activités et le tableau 2.5 présente les compétences maîtrisées par chaque ressource.

Tableau 2.3 : Compétences requises

S_0	Programmeur
S_1	Concepteur
S_2	Webmaster

Tableau 2.4 : Quantité de ressources requises par activité et par compétence ($b_{i,k}$)

	S_0	**S_1**	**S_2**
A_1	-	1	1
A_2	1	-	1
A_3	2	1	-
A_4	-	1	-

Le tableau 2.4 permet de constater que l'activité A_3, par exemple, nécessite deux ressources possédant la compétence S_0 ainsi qu'une ressource possédant la compétence S_1 tandis que l'activité A_4 nécessite uniquement une ressource maîtrisant la compétence S_1.

Tableau 2.5 : Compétences maîtrisées par ressource ($MS_{m,k}$)

	S_0	**S_1**	**S_2**
W_0	-	1	1
W_1	1	-	1
W_2	1	1	-
W_3	1	-	1

Le tableau 2.5 permet de constater que la ressource W_2 possède les compétences S_0 et S_1 et que la compétence S_1 est seulement maîtrisée par les ressources W_0 et W_2.

Puis, la durée de chaque activité en jours et les relations de précédence entre chacune d'entre elles sont aussi connues. Elles sont présentées à la figure 2.4, où la durée de l'activité A_i est inscrite au-dessus de la liaison entre l'activité A_i et A_j.

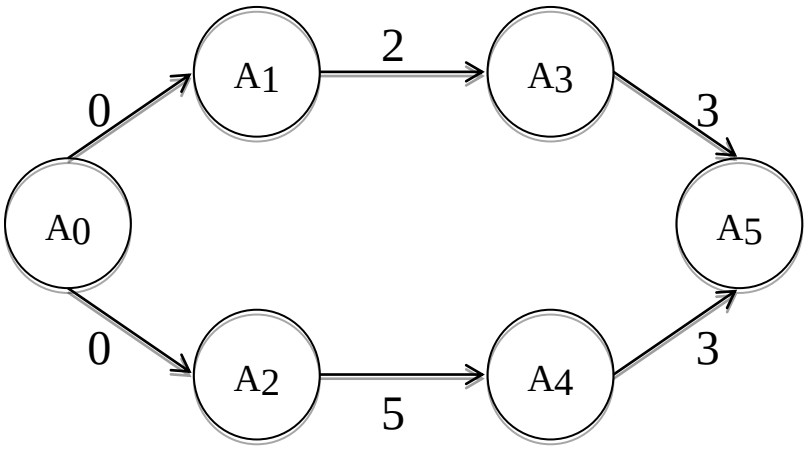

Figure 2.4 : Relations de précédence entre les activités

Les activités A_0 et A_5 présentes à la figure 2.4 sont les activités de début et de fin du projet. Comme il est possible de le constater en observant cette figure, l'activité A_1 précède l'activité A_3 et l'activité A_2 précède l'activité A_4.

2.4.2 Déterminer le début au plus tôt et le début au plus tard de chaque activité

Cette seconde étape est réalisée afin de trouver les solutions de Pareto plus rapidement. Il est à noter que les débuts au plus tôt et au plus tard sont relatifs au problème d'ordonnancement classique sans contrainte de ressources étant donné que les activités peuvent être réalisées simultanément et que l'on désire déterminer des

intervalles les plus grands possible pour les débuts potentiels de chaque activité, et ce afin de ne pas supprimer des ordonnancements possibles.

Les chaînages avant et arrière du diagramme de Gantt sont effectués afin de déterminer l'intervalle de temps dans lequel chaque activité doit être exécutée, ces derniers sont présentés à la figure 2.5.

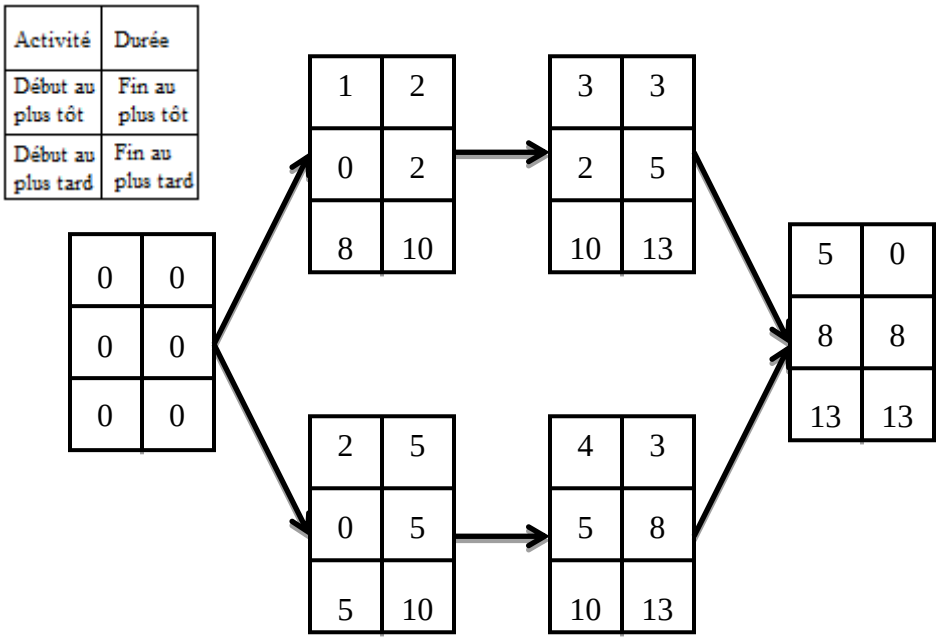

Figure 2.5 : Chaînages avant et arrière du diagramme de Gantt

Le chaînage avant, permettant de déterminer les dates de réalisation au plus tôt des activités, est effectué selon la méthode conventionnelle, en parcourant le réseau activité par activité depuis l'activité de départ jusqu'à l'activité de fin du projet. Dans le cas de ce problème, les dates de réalisation au plus tard de l'activité de fin du projet ne sont pas les mêmes que celles de réalisation au plus tôt du projet étant donné que l'ordonnancement du projet est basé sur la disponibilité des ressources. Les dates de début et de fin au plus tard du projet sont égales à la somme des durées de chaque activité. De cette manière, le pire scénario selon la durée se produit lorsque les quatre activités sont

réalisées l'une après l'autre. Lorsque la date de réalisation au plus tard du projet est déterminée, le chaînage arrière est effectué selon la méthode traditionnelle.

Comme il est possible de le constater en observant la figure 2.5, la première activité doit commencer entre les temps 0 et 8 inclusivement, la seconde activité doit commencer entre les temps 0 et 5 inclusivement, la troisième activité doit commencer entre les temps 2 et 10 inclusivement et la quatrième activité doit débuter entre les temps 5 et 10 inclusivement.

De plus, étant donné que la date de début au plus tard de l'activité cinq, soit l'activité marquant la fin du projet est de 13 jours après le début de celui-ci, l'horizon temporel du projet est fixé à 13.

2.4.3 Définition des variables du problème

Tel que le tableau 2.2 le présente, deux variables binaires sont utilisées, $x_{i,m}^{t}$ afin de déterminer si la ressource W_m débute l'activité A_i au temps t et $y_{i,m}^{k}$ afin de déterminer si la ressource W_m utilise la compétence S_k pour réaliser l'activité A_i.

Comme le problème du temps indexé présenté dans Montoya (2012) le démontre, le nombre de variables pour $x_{i,m}^{t}$ devrait être égal à la multiplication du nombre d'activités, du nombre de ressources et de l'horizon temporel alors que le nombre de variables pour $y_{i,m}^{k}$ devrait être égal la multiplication du nombre d'activités, du nombre de ressources et du nombre de compétences. En théorie, il devrait donc y avoir, avec un horizon temporel fixé à 13, 208 variables pour représenter la variable binaire x et 48 variables pour représenter la variable binaire y.

Afin de faciliter l'implantation du modèle, seulement les variables se trouvant dans les intervalles de temps entre le début au plus tôt et le début au plus tard de chaque activité sont prises en considération, diminuant la quantité de variables utilisées pour représenter la variable binaire x à 120. La quantité de variables utilisées pour représenter

la variable binaire y demeure à 48 étant donné qu'elle ne prend pas le temps en considération.

Le temps de départ de chaque activité est aussi considéré et il est représenté par les variables entières t_1 à t_5, t_5 représentant le temps de départ de l'activité de fin du projet et, par le fait même, la durée du projet.

Afin de représenter les variables avec Midaco, les variables binaires $x_{i,m}^{t}$ et $y_{i,m}^{k}$ ainsi que les variables entières t_i sont représentées uniquement par la variable x. Le tableau 2.6 présente les variables utilisées ainsi que leur conversion afin d'être utilisées avec Midaco. Il est à noter que la variable t_0 n'est pas utilisée étant donné que, logiquement, le projet débute toujours au temps zéro.

Tableau 2.6 : Tableau des variables

$x_{1,m}^t$		$x_{2,m}^t$		$x_{3,m}^t$		$x_{4,m}^t$		$y_{1,m}^k$		$y_{2,m}^k$		$y_{3,m}^k$		$y_{4,m}^k$		t	
$x_{1,0}^0$	X1	$x_{2,0}^0$	X37	$x_{3,0}^2$	X61	$x_{4,0}^5$	X97	$y_{1,0}^0$	X121	$y_{2,0}^0$	X133	$y_{3,0}^0$	X145	$y_{4,0}^0$	X157	t_1	X169
$x_{1,1}^0$	X2	$x_{2,1}^0$	X38	$x_{3,1}^2$	X62	$x_{4,1}^5$	X98	$y_{1,1}^0$	X122	$y_{2,1}^0$	X134	$y_{3,1}^0$	X146	$y_{4,1}^0$	X158	t_2	X170
$x_{1,2}^0$	X3	$x_{2,2}^0$	X39	$x_{3,2}^2$	X63	$x_{4,2}^5$	X99	$y_{1,2}^0$	X123	$y_{2,2}^0$	X135	$y_{3,2}^0$	X147	$y_{4,2}^0$	X159	t_3	X171
$x_{1,3}^0$	X4	$x_{2,3}^0$	X40	$x_{3,3}^2$	X64	$x_{4,3}^5$	X100	$y_{1,3}^0$	X124	$y_{2,3}^0$	X136	$y_{3,3}^0$	X148	$y_{4,3}^0$	X160	t_4	X172
$x_{1,0}^1$	X5	$x_{2,0}^1$	X41	$x_{3,0}^3$	X65	$x_{4,0}^6$	X101	$y_{1,0}^1$	X125	$y_{2,0}^1$	X137	$y_{3,0}^1$	X149	$y_{4,0}^1$	X161	t_5	X173
$x_{1,1}^1$	X6	$x_{2,1}^1$	X42	$x_{3,1}^3$	X66	$x_{4,1}^6$	X102	$y_{1,1}^1$	X126	$y_{2,1}^1$	X138	$y_{3,1}^1$	X150	$y_{4,1}^1$	X162		
$x_{1,2}^1$	X7	$x_{2,2}^1$	X43	$x_{3,2}^3$	X67	$x_{4,2}^6$	X103	$y_{1,2}^1$	X127	$y_{2,2}^1$	X139	$y_{3,2}^1$	X151	$y_{4,2}^1$	X163		
$x_{1,3}^1$	X8	$x_{2,3}^1$	X44	$x_{3,3}^3$	X68	$x_{4,3}^6$	X104	$y_{1,3}^1$	X128	$y_{2,3}^1$	X140	$y_{3,3}^1$	X152	$y_{4,3}^1$	X164		
$x_{1,0}^2$	X9	$x_{2,0}^2$	X45	$x_{3,0}^4$	X69	$x_{4,0}^7$	X105	$y_{1,0}^2$	X129	$y_{2,0}^2$	X141	$y_{3,0}^2$	X153	$y_{4,0}^2$	X165		
$x_{1,1}^2$	X10	$x_{2,1}^2$	X46	$x_{3,1}^4$	X70	$x_{4,1}^7$	X106	$y_{1,1}^2$	X130	$y_{2,1}^2$	X142	$y_{3,1}^2$	X154	$y_{4,1}^2$	X166		
$x_{1,2}^2$	X11	$x_{2,2}^2$	X47	$x_{3,2}^4$	X71	$x_{4,2}^7$	X107	$y_{1,2}^2$	X131	$y_{2,2}^2$	X143	$y_{3,2}^2$	X155	$y_{4,2}^2$	X167		
$x_{1,3}^2$	X12	$x_{2,3}^2$	X48	$x_{3,3}^4$	X72	$x_{4,3}^7$	X108	$y_{1,3}^2$	X132	$y_{2,3}^2$	X144	$y_{3,3}^2$	X156	$y_{4,3}^2$	X168		
$x_{1,0}^3$	X13	$x_{2,0}^3$	X49	$x_{3,0}^5$	X73	$x_{4,0}^8$	X109										
$x_{1,1}^3$	X14	$x_{2,1}^3$	X50	$x_{3,1}^5$	X74	$x_{4,1}^8$	X110										
$x_{1,2}^3$	X15	$x_{2,2}^3$	X51	$x_{3,2}^5$	X75	$x_{4,2}^8$	X111										
$x_{1,3}^3$	X16	$x_{2,3}^3$	X52	$x_{3,3}^5$	X76	$x_{4,3}^8$	X112										
$x_{1,0}^4$	X17	$x_{2,0}^4$	X53	$x_{3,0}^6$	X77	$x_{4,0}^9$	X113										
$x_{1,1}^4$	X18	$x_{2,1}^4$	X54	$x_{3,1}^6$	X78	$x_{4,1}^9$	X114										
$x_{1,2}^4$	X19	$x_{2,2}^4$	X55	$x_{3,2}^6$	X79	$x_{4,2}^9$	X115										
$x_{1,3}^4$	X20	$x_{2,3}^4$	X56	$x_{3,3}^6$	X80	$x_{4,3}^9$	X116										
$x_{1,0}^5$	X21	$x_{2,0}^5$	X57	$x_{3,0}^7$	X81	$x_{4,0}^{10}$	X117										
$x_{1,1}^5$	X22	$x_{2,1}^5$	X58	$x_{3,1}^7$	X82	$x_{4,1}^{10}$	X118										
$x_{1,2}^5$	X23	$x_{2,2}^5$	X59	$x_{3,2}^7$	X83	$x_{4,2}^{10}$	X119										
$x_{1,3}^5$	X24	$x_{2,3}^5$	X60	$x_{3,3}^7$	X82	$x_{4,3}^{10}$	X120										
$x_{1,0}^6$	X25			$x_{3,0}^8$	X85												
$x_{1,1}^6$	X26			$x_{3,1}^8$	X86												
$x_{1,2}^6$	X27			$x_{3,2}^8$	X87												
$x_{1,3}^6$	X28			$x_{3,3}^8$	X88												
$x_{1,0}^7$	X29			$x_{3,0}^9$	X89												
$x_{1,1}^7$	X30			$x_{3,1}^9$	X90												
$x_{1,2}^7$	X31			$x_{3,2}^9$	X91												
$x_{1,3}^7$	X32			$x_{3,3}^9$	X92												
$x_{1,0}^8$	X33			$x_{3,0}^{10}$	X93												
$x_{1,1}^8$	X34			$x_{3,1}^{10}$	X94												
$x_{1,2}^8$	X35			$x_{3,2}^{10}$	X95												
$x_{1,3}^8$	X36			$x_{3,3}^{10}$	X96												

Tel que le tableau 2.6 le démontre, les variables $x_{i,m}^t$ sont classées en ordre chronologique en fonction de l'activité et, pour chaque activité, chacune des variables évolue selon la ressource, puis selon le temps. Les variables $y_{i,m}^k$ sont aussi classées en ordre chronologique selon l'activité et évoluent selon la ressource, puis selon la compétence. Les variables temporelles sont positionnées à la suite des variables binaires.

2.4.4 Résolution du problème avec Midaco

Les étapes présentées à la section 2.3.1 sont reprises et appliquées à l'exemple numérique. Le code complet du problème, utilisé avec Matlab, est présenté à l'annexe A. La résolution du problème est effectuée avec la cinquième version de Midaco, cependant, la sixième version est désormais disponible.

2.4.4.1 Déterminer les dimensions, les limites et le point de départ du problème

La première étape consiste à déterminer les dimensions du problème. Pour celui-ci, bien que l'objectif principal consiste à optimiser le temps et le coût du projet, trois objectifs sont présents. De fait, MIDACO va optimiser la fonction objectif agrégée qui est une combinaison linéaire des deux sous objectifs que sont le temps et le coût du projet. Au total, 173 variables sont utilisées, elles sont toutes des variables entières, 168 contraintes, dont 32 contraintes d'égalité, doivent être résolues. Le début au plus tôt et le début au plus tard de chaque activité ainsi que l'horizon temporel du projet sont ajoutés selon les valeurs déterminées à la section 2.4.2. Cette étape est représentée par les lignes 14 à 30 du code Matlab, les voici :

```
% Étape 1.A: Dimensions du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
problem.o  = 3; % Nombre d'objectifs
problem.n  = 173; % Nombre de variables (au total)
problem.ni = 173; % Nombre de variables entières (0 <= nint <= n)
problem.m  = 168; % Nombre de contraintes (au total)
problem.me = 32; % Nombre de contraintes d'égalité (0 <= me <= m)

es1=0; % Début au plus tôt activité 1
es2=0; % Début au plus tôt activité 2
es3=2; % Début au plus tôt activité 3
es4=5; % Début au plus tôt activité 4

ls1=8; % Début au plus tard activité 1
ls2=5; % Début au plus tard activité 2
ls3=10; % Début au plus tard activité 3
```

```
ls4=10; % Début au plus tard activité 4

T=13; % Horizon temporel
```

Les variables x0 à x168 sont des variables binaires, donc, pour chacune d'entre elles, la limite inférieure est égale à 0 et la limite supérieure est égale à 1. Pour les variables x169 à x172, les limites inférieure et supérieure de chacune d'entre elles sont égales au début au plus tôt et au début au plus tard de chaque activité. Pour la variable x173, sa limite inférieure est nulle et sa limite supérieure est égale à l'horizon temporel du problème. Cette étape est représentée par les lignes 35 à 38 du code Matlab que voici :

```
% Étape 1.B: Limites inférieures et supérieures 'xl' & 'xu'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
problem.xl =
[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;es1;es2;es3;es4;8]
;

problem.xu =
[1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;
1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;
1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;ls1;ls2;ls3;ls4;T]
;
```

Le point de départ du projet est, tel que suggéré par Midaco, égal aux limites inférieures du problème. Celui-ci est représenté par la 42e ligne du code Matlab, soit :

```
problem.x  = problem.xl; % 'x' = limites inférieures 'xl'
```

2.4.4.2 Déterminer les critères d'arrêt et les options d'impression

En ce qui concerne les critères d'arrêt, le nombre maximal d'évaluations de la fonction et fixé à 10 000 000 pour la totalité des essais. Pour certains essais, toutes les solutions de Pareto sont obtenues après moins de 2 000 000 d'évaluations. Cependant,

pour d'autres essais, toutes les solutions de Pareto sont obtenues après 7 000 000 d'évaluations. Donc, le nombre d'évaluations a été fixé à 10 000 000 afin de s'assurer d'obtenir un nombre maximal de solutions pour chaque essai. Le temps limite d'évaluation du problème est resté inchangé pour tous les essais et il est d'une journée. Les critères d'arrêt sont représentés par les lignes 50 et 51 du code Matlab, les voici :

```
% Étape 2.A: Critères d'arrêt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
option.maxeval  = 10000000;    % Nombre maximal d'évaluations de la
fonction (e.g. 1000000)
option.maxtime  = 60*60*24; % Temps limite maximal en secondes (e.g. 1
jour = 60*60*24)
```

Les options d'impression sont restées inchangées par rapport à l'exemple de départ de Midaco, donc la fréquence d'impression de la meilleure solution est de 1000 et l'option d'enregistrement de la solution en fichier .txt est activée. Ces options sont représentées par les lignes 55 et 56 du code Matlab, les voici :

```
% Étape 2.B: Options d'impression
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
option.printeval  = 1000;  % Fréquence d'impression de la meilleure
solution actuelle (e.g. 1000)
option.save2file  = 1;     % Sauvegarde de l'écran et de la solution en
fichier .txt [ 0=NON/ 1=OUI]
```

2.4.4.3 Choisir les paramètres de Midaco

Pour certains essais, les paramètres demeurent nuls, tandis que pour d'autres essais, un ou quelques paramètres sont modifiés. Pour la résolution de ce problème, les paramètres modifiés sont les paramètres 2, 7, 8 et 11. Le tableau 2.7 présente les paramètres modifiés, s'il y a lieu, pour les essais retenus. Les paramètres de Midaco sont représentés par les lignes 63 à 74 du code Matlab.

2.4.4.4 Déterminer les fonctions du problème d'optimisation

La fonction utilisée afin de résoudre le problème d'optimisation avec Matlab est toujours la même, il s'agit de la fonction représentée par la ligne 95 du code, soit : `function [ f, g ] = problem_function( x ).`

2.4.4.5 Déterminer les paramètres connus du problème

Les paramètres connus du problème sont représentés par les lignes 99 à 132 du code Matlab, les voici :

```
%  Paramètres connus du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0:13; % Paramètre variable temporel

p1=2; % Durée de l'activité 1
p2=5; % Durée de l'activité 2
p3=3; % Durée de l'activité 3
p4=3; % Durée de l'activité 4

% Nombre de ressource(s) requise(s) pour la compétence k pour
l'activité i
b01=0;
b11=1;
b21=1;
b02=1;
b12=0;
b22=1;
b03=2;
b13=1;
b23=0;
b04=0;
b14=1;
b24=0;

% Coût en $ par ressource m par compétence k, pour 1 unité de temps
C00=720;
C01=800;
C02=960;
C10=640;
C11=720;
C12=800;
C20=640;
C21=720;
```

```
C22=780;
C30=560;
C31=640;
C32=720;
```

Le paramètre variable temporel, la durée des activités et le nombre de ressource(s) requise(s) par chaque activité afin de réaliser chaque compétence sont définis en fonction des données du problème. Le coût d'utilisation d'une ressource pour une compétence donnée par unité de temps est donné par le paramètre $C_{m,k}$. Ces valeurs sont attribuées arbitrairement en fonction d'un taux horaire entre 70 et 120$/h pour une journée de huit heures de travail.

2.4.4.6 Représenter les fonctions objectif.u

Ce problème se compose de trois fonctions objectif et elles sont représentées par les lignes 136 à 138 du code, les voici :

```
%  Fonctions objectif du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f(2) = x(173);
f(3) =
x(121)*p1*C00+x(122)*p1*C10+x(123)*p1*C20+x(124)*p1*C30+x(125)*p1*C01+x
(126)*p1*C11+x(127)*p1*C21+x(128)*p1*C31+x(129)*p1*C02+x(130)*p1*C12+x(
131)*p1*C22+x(132)*p1*C32+x(133)*p2*C00+x(134)*p2*C10+x(135)*p2*C20+x(1
36)*p2*C30+x(137)*p2*C01+x(138)*p2*C11+x(139)*p2*C21+x(140)*p2*C31+x(14
1)*p2*C02+x(142)*p2*C12+x(143)*p2*C22+x(144)*p2*C32+x(145)*p3*C00+x(146
)*p3*C10+x(147)*p3*C20+x(148)*p3*C30+x(149)*p3*C01+x(150)*p3*C11+x(151)
*p3*C21+x(152)*p3*C31+x(153)*p3*C02+x(154)*p3*C12+x(155)*p3*C22+x(156)*
p3*C32+x(157)*p4*C00+x(158)*p4*C10+x(159)*p4*C20+x(160)*p4*C30+x(161)*p
4*C01+x(162)*p4*C11+x(163)*p4*C21+x(164)*p4*C31+x(165)*p4*C02+x(166)*p4
*C12+x(167)*p4*C22+x(168)*p4*C32;
f(1) = f(2)+f(3);
```

Les fonctions f(2) et f(3) représentent la durée totale et le coût total du projet. La fonction f(2) consiste à minimiser la variable x(173), représentant la date de début de l'activité de fin du projet. La fonction f(3) consiste à minimiser le coût du projet, celle-ci est représentée par l'équation 2.12. Puis, la fonction f(1) représente la somme des deux

fonctions précédentes, il s'agit de la fonction principale à optimiser. De cette manière, il est possible d'attribuer des facteurs d'importance à l'une ou l'autre des fonctions de temps ou de coût. Si l'utilisateur souhaite accorder une importance supérieure au coût du projet, la fonction f(3) présente dans la fonction f(1) peut être multipliée par une pondération quelconque. La fonction f(1) pourrait donc s'écrire, par exemple, de la manière suivante en utilisant un facteur de 100 :

```
f(1) = f(2)+100*f(3);
```

Le fichier obtenu présente le résultat optimisé pour la fonction f(1) ainsi que les valeurs individuelles du temps et du coût qui la composent.

2.4.4.7 Représenter les contraintes d'égalité

Les contraintes d'égalité du problème, soient les équations 2.1, 2.10 et 2.11 présentées à la section 2.2, sont représentées par les lignes 143 à 181. Voici la section du code Matlab utilisé pour les contraintes d'égalité :

```
%  Contraintes d'Égalité G(X)=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Équation 2.1
g(1)=x(1)*t(0+1)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t
(1+1)+x(7)*t(1+1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(1
2)*t(2+1)+x(13)*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4
+1)+x(18)*t(4+1)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(
23)*t(5+1)+x(24)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(
6+1)+x(29)*t(7+1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x
(34)*t(8+1)+x(35)*t(8+1)+x(36)*t(8+1)-x(169)*(b01+b11+b21);

g(2)=x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(0+1)+x(41)*t(1+1)+x
(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x(46)*t(2+1)+x(47)*t
(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t(3+1)+x(52)*t(3+1)+
x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+x(57)*t(5+1)+x(58)*
t(5+1)+x(59)*t(5+1)+x(60)*t(5+1)-x(170)*(b02+b12+b22);

g(3)=x(61)*t(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x
(66)*t(3+1)+x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t
(4+1)+x(72)*t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+
x(77)*t(6+1)+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*
t(7+1)+x(83)*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)
```

```
+x(88)*t(8+1)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)
*t(10+1)+x(94)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1)-
x(171)*(b03+b13+b23);

g(4)=x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(100)*t(5+1)+x(101)*t(6+1)
+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(105)*t(7+1)+x(106)*t(7+1)+
x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(110)*t(8+1)+x(111)*t(8+1)+x
(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115)*t(9+1)+x(116)*t(9+1)+x(
117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(120)*t(10+1)-
x(172)*(b04+b14+b24);

%  Équation 2.10
g(5)=x(121)+x(122)+x(123)+x(124)-b01;
g(6)=x(125)+x(126)+x(127)+x(128)-b11;
g(7)=x(129)+x(130)+x(131)+x(132)-b21;
g(8)=x(133)+x(134)+x(135)+x(136)-b02;
g(9)=x(137)+x(138)+x(139)+x(140)-b12;
g(10)=x(141)+x(142)+x(143)+x(144)-b22;
g(11)=x(145)+x(146)+x(147)+x(148)-b03;
g(12)=x(149)+x(150)+x(151)+x(152)-b13;
g(13)=x(153)+x(154)+x(155)+x(156)-b23;
g(14)=x(157)+x(158)+x(159)+x(160)-b04;
g(15)=x(161)+x(162)+x(163)+x(164)-b14;
g(16)=x(165)+x(166)+x(167)+x(168)-b24;

%  Équation 2.11
g(17)=x(1)+x(5)+x(9)+x(13)+x(17)+x(21)+x(25)+x(29)+x(33)-x(121)-x(125)-
x(129);
g(18)=x(2)+x(6)+x(10)+x(14)+x(18)+x(22)+x(26)+x(30)+x(34)-x(122)-
x(126)-x(130);
g(19)=x(3)+x(7)+x(11)+x(15)+x(19)+x(23)+x(27)+x(31)+x(35)-x(123)-
x(127)-x(131);
g(20)=x(4)+x(8)+x(12)+x(16)+x(20)+x(24)+x(28)+x(32)+x(36)-x(124)-
x(128)-x(132);

g(21)=x(37)+x(41)+x(45)+x(49)+x(53)+x(57)-x(133)-x(137)-x(141);
g(22)=x(38)+x(42)+x(46)+x(50)+x(54)+x(58)-x(134)-x(138)-x(142);
g(23)=x(39)+x(43)+x(47)+x(51)+x(55)+x(59)-x(135)-x(139)-x(143);
g(24)=x(40)+x(44)+x(48)+x(52)+x(56)+x(60)-x(136)-x(140)-x(144);

g(25)=x(61)+x(65)+x(69)+x(73)+x(77)+x(81)+x(85)+x(89)+x(93)-x(145)-
x(149)-x(153);
g(26)=x(62)+x(66)+x(70)+x(74)+x(78)+x(82)+x(86)+x(90)+x(94)-x(146)-
x(150)-x(154);
g(27)=x(63)+x(67)+x(71)+x(75)+x(79)+x(83)+x(87)+x(91)+x(95)-x(147)-
x(151)-x(155);
g(28)=x(64)+x(68)+x(72)+x(76)+x(80)+x(84)+x(88)+x(92)+x(96)-x(148)-
x(152)-x(156);

g(29)=x(97)+x(101)+x(105)+x(109)+x(113)+x(117)-x(157)-x(161)-x(165);
g(30)=x(98)+x(102)+x(106)+x(110)+x(114)+x(118)-x(158)-x(162)-x(166);
g(31)=x(99)+x(103)+x(107)+x(111)+x(115)+x(119)-x(159)-x(163)-x(167);
g(32)=x(100)+x(104)+x(108)+x(112)+x(116)+x(120)-x(160)-x(164)-x(168);
```

2.4.4.8 Représenter les contraintes d'inégalité

Les contraintes d'inégalité du problème, soient les équations 2.4, 2.6, 2.7, 2.8 et 2.9 présentées à la section 2.2, sont représentées par les lignes 186 à 341. L'équation 2.5 n'est pas représentée étant donné que les limites inférieures et supérieures du projet sont présentées plus tôt dans le code. Voici la section du code Matlab utilisé pour les contraintes d'inégalité :

```
%  Contraintes d'inégalité G(x)>=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Équation 2.4
g(33)=-x(169)+x(171)-p1;
g(34)=-x(170)+x(172)-p2;
g(35)=-x(171)+x(173)-p3;
g(36)=-x(172)+x(173)-p4;

%  Équation 2.6
g(37)=-(x(1)+x(5)+x(9)+x(13)+x(17)+x(21)+x(25)+x(29)+x(33))+1;
g(38)=-(x(2)+x(6)+x(10)+x(14)+x(18)+x(22)+x(26)+x(30)+x(34))+1;
g(39)=-(x(3)+x(7)+x(11)+x(15)+x(19)+x(23)+x(27)+x(31)+x(35))+1;
g(40)=-(x(4)+x(8)+x(12)+x(16)+x(20)+x(24)+x(28)+x(32)+x(36))+1;

g(41)=-(x(37)+x(41)+x(45)+x(49)+x(53)+x(57))+1;
g(42)=-(x(38)+x(42)+x(46)+x(50)+x(54)+x(58))+1;
g(43)=-(x(39)+x(43)+x(47)+x(51)+x(55)+x(59))+1;
g(44)=-(x(40)+x(44)+x(48)+x(52)+x(56)+x(60))+1;

g(45)=-(x(61)+x(65)+x(69)+x(73)+x(77)+x(81)+x(85)+x(89)+x(93))+1;
g(46)=-(x(62)+x(66)+x(70)+x(74)+x(78)+x(82)+x(86)+x(90)+x(94))+1;
g(47)=-(x(63)+x(67)+x(71)+x(75)+x(79)+x(83)+x(87)+x(91)+x(95))+1;
g(48)=-(x(64)+x(68)+x(72)+x(76)+x(80)+x(84)+x(88)+x(92)+x(96))+1;

g(49)=-(x(97)+x(101)+x(105)+x(109)+x(113)+x(117))+1;
g(50)=-(x(98)+x(102)+x(106)+x(110)+x(114)+x(118))+1;
g(51)=-(x(99)+x(103)+x(107)+x(111)+x(115)+x(119))+1;
g(52)=-(x(100)+x(104)+x(108)+x(112)+x(116)+x(120))+1;

%  Équation 2.7
g(53)=-(x(1)+x(37))+1;
g(54)=-(x(1)+x(5)+x(37)+x(41))+1;
g(55)=-(x(5)+x(9)+x(37)+x(41)+x(45)+x(61))+1;
g(56)=-(x(9)+x(13)+x(37)+x(41)+x(45)+x(49)+x(61)+x(65))+1;
g(57)=-(x(13)+x(17)+x(37)+x(41)+x(45)+x(49)+x(53)+x(61)+x(65)+x(69))+1;
g(58)=-
(x(17)+x(21)+x(41)+x(45)+x(49)+x(53)+x(57)+x(65)+x(69)+x(73)+x(97))+1;
g(59)=-
(x(21)+x(25)+x(45)+x(49)+x(53)+x(57)+x(69)+x(73)+x(77)+x(97)+x(101))+1;
```

```
g(60)=-
(x(25)+x(29)+x(49)+x(53)+x(57)+x(73)+x(77)+x(81)+x(97)+x(101)+x(105))+1
;
g(61)=-
(x(29)+x(33)+x(53)+x(57)+x(77)+x(81)+x(85)+x(101)+x(105)+x(109))+1;
g(62)=-(x(33)+x(57)+x(81)+x(85)+x(89)+x(105)+x(109)+x(113))+1;
g(63)=-(x(85)+x(89)+x(93)+x(109)+x(113)+x(117))+1;
g(64)=-(x(89)+x(93)+x(113)+x(117))+1;
g(65)=-(x(93)+x(117))+1;

g(66)=-(x(2)+x(38))+1;
g(67)=-(x(2)+x(6)+x(38)+x(42))+1;
g(68)=-(x(6)+x(10)+x(38)+x(42)+x(46)+x(62))+1;
g(69)=-(x(10)+x(14)+x(38)+x(42)+x(46)+x(50)+x(62)+x(66))+1;
g(70)=-(x(14)+x(18)+x(38)+x(42)+x(46)+x(50)+x(54)+x(62)+x(66)+x(70))+1;
g(71)=-
(x(18)+x(22)+x(42)+x(46)+x(50)+x(54)+x(58)+x(66)+x(70)+x(74)+x(98))+1;
g(72)=-
(x(22)+x(26)+x(46)+x(50)+x(54)+x(58)+x(70)+x(74)+x(78)+x(98)+x(102))+1;
g(73)=-
(x(26)+x(30)+x(50)+x(54)+x(58)+x(74)+x(78)+x(82)+x(98)+x(102)+x(106))+1
;
g(74)=-
(x(30)+x(34)+x(54)+x(58)+x(78)+x(82)+x(86)+x(102)+x(106)+x(110))+1;
g(75)=-(x(34)+x(58)+x(82)+x(86)+x(90)+x(106)+x(110)+x(114))+1;
g(76)=-(x(86)+x(90)+x(94)+x(110)+x(114)+x(118))+1;
g(77)=-(x(90)+x(94)+x(114)+x(118))+1;
g(78)=-(x(94)+x(118))+1;

g(79)=-(x(3)+x(39))+1;
g(80)=-(x(3)+x(7)+x(39)+x(43))+1;
g(81)=-(x(7)+x(11)+x(39)+x(43)+x(47)+x(63))+1;
g(82)=-(x(11)+x(15)+x(39)+x(43)+x(47)+x(51)+x(63)+x(67))+1;
g(83)=-(x(15)+x(19)+x(39)+x(43)+x(47)+x(51)+x(55)+x(63)+x(67)+x(71))+1;
g(84)=-
(x(19)+x(23)+x(43)+x(47)+x(51)+x(55)+x(59)+x(67)+x(71)+x(75)+x(99))+1;
g(85)=-
(x(23)+x(27)+x(47)+x(51)+x(55)+x(59)+x(71)+x(75)+x(79)+x(99)+x(103))+1;
g(86)=-
(x(27)+x(31)+x(51)+x(55)+x(59)+x(75)+x(79)+x(83)+x(99)+x(103)+x(107))+1
;
g(87)=-
(x(31)+x(35)+x(55)+x(59)+x(79)+x(83)+x(87)+x(103)+x(107)+x(111))+1;
g(88)=-(x(35)+x(59)+x(83)+x(87)+x(91)+x(107)+x(111)+x(115))+1;
g(89)=-(x(87)+x(91)+x(95)+x(111)+x(115)+x(119))+1;
g(90)=-(x(91)+x(95)+x(115)+x(119))+1;
g(91)=-(x(95)+x(119))+1;

g(92)=-(x(4)+x(40))+1;
g(93)=-(x(4)+x(8)+x(40)+x(44))+1;
g(94)=-(x(8)+x(12)+x(40)+x(44)+x(48)+x(64))+1;
g(95)=-(x(12)+x(16)+x(40)+x(44)+x(48)+x(52)+x(64)+x(68))+1;
g(96)=-(x(16)+x(20)+x(40)+x(44)+x(48)+x(52)+x(56)+x(64)+x(68)+x(72))+1;
g(97)=-
```

```
(x(20)+x(24)+x(44)+x(48)+x(52)+x(56)+x(60)+x(68)+x(72)+x(76)+x(100))+1;
g(98)=-
(x(24)+x(28)+x(48)+x(52)+x(56)+x(60)+x(72)+x(76)+x(80)+x(100)+x(104))+1
;
g(99)=-
(x(28)+x(32)+x(52)+x(56)+x(60)+x(76)+x(80)+x(84)+x(100)+x(104)+x(108))+
1;
g(100)=-
(x(32)+x(36)+x(56)+x(60)+x(80)+x(84)+x(88)+x(104)+x(108)+x(112))+1;
g(101)=-(x(36)+x(60)+x(84)+x(88)+x(92)+x(108)+x(112)+x(116))+1;
g(102)=-(x(88)+x(92)+x(96)+x(112)+x(116)+x(120))+1;
g(103)=-(x(92)+x(96)+x(116)+x(120))+1;
g(104)=-(x(96)+x(120))+1;

%  Équation 2.8
g(105)=-
(b01+b11+b21)*(x(1)*t(0+1)+x(5)*t(1+1)+x(9)*t(2+1)+x(13)*t(3+1)+x(17)*t
(4+1)+x(21)*t(5+1)+x(25)*t(6+1)+x(29)*t(7+1)+x(33)*t(8+1))+(x(1)*t(0+1)
+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+1
)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)*
t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1)
+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24)
*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+1
)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(35
)*t(8+1)+x(36)*t(8+1));

g(106)=-
(b01+b11+b21)*(x(2)*t(0+1)+x(6)*t(1+1)+x(10)*t(2+1)+x(14)*t(3+1)+x(18)*
t(4+1)+x(22)*t(5+1)+x(26)*t(6+1)+x(30)*t(7+1)+x(34)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));
g(107)=-
(b01+b11+b21)*(x(3)*t(0+1)+x(7)*t(1+1)+x(11)*t(2+1)+x(15)*t(3+1)+x(19)*
t(4+1)+x(23)*t(5+1)+x(27)*t(6+1)+x(31)*t(7+1)+x(35)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));

g(108)=-
(b01+b11+b21)*(x(4)*t(0+1)+x(8)*t(1+1)+x(12)*t(2+1)+x(16)*t(3+1)+x(20)*
t(4+1)+x(24)*t(5+1)+x(28)*t(6+1)+x(32)*t(7+1)+x(36)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
```

```
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));

g(109)=-
(b02+b12+b22)*(x(37)*t(0+1)+x(41)*t(1+1)+x(45)*t(2+1)+x(49)*t(3+1)+x(53
)*t(4+1)+x(57)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(110)=-
(b02+b12+b22)*(x(38)*t(0+1)+x(42)*t(1+1)+x(46)*t(2+1)+x(50)*t(3+1)+x(54
)*t(4+1)+x(58)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(111)=-
(b02+b12+b22)*(x(39)*t(0+1)+x(43)*t(1+1)+x(47)*t(2+1)+x(51)*t(3+1)+x(55
)*t(4+1)+x(59)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(112)=-
(b02+b12+b22)*(x(40)*t(0+1)+x(44)*t(1+1)+x(48)*t(2+1)+x(52)*t(3+1)+x(56
)*t(4+1)+x(60)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(113)=-
(b03+b13+b23)*(x(61)*t(2+1)+x(65)*t(3+1)+x(69)*t(4+1)+x(73)*t(5+1)+x(77
)*t(6+1)+x(81)*t(7+1)+x(85)*t(8+1)+x(89)*t(9+1)+x(93)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(114)=-
(b03+b13+b23)*(x(62)*t(2+1)+x(66)*t(3+1)+x(70)*t(4+1)+x(74)*t(5+1)+x(78
)*t(6+1)+x(82)*t(7+1)+x(86)*t(8+1)+x(90)*t(9+1)+x(94)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
```

```
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(115)=-
(b03+b13+b23)*(x(63)*t(2+1)+x(67)*t(3+1)+x(71)*t(4+1)+x(75)*t(5+1)+x(79
)*t(6+1)+x(83)*t(7+1)+x(87)*t(8+1)+x(91)*t(9+1)+x(95)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(116)=-
(b03+b13+b23)*(x(64)*t(2+1)+x(68)*t(3+1)+x(72)*t(4+1)+x(76)*t(5+1)+x(80
)*t(6+1)+x(84)*t(7+1)+x(88)*t(8+1)+x(92)*t(9+1)+x(96)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(117)=-
(b04+b14+b24)*(x(97)*t(5+1)+x(101)*t(6+1)+x(105)*t(7+1)+x(109)*t(8+1)+x
(113)*t(9+1)+x(117)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(118)=-
(b04+b14+b24)*(x(98)*t(5+1)+x(102)*t(6+1)+x(106)*t(7+1)+x(110)*t(8+1)+x
(114)*t(9+1)+x(118)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(119)=-
(b04+b14+b24)*(x(99)*t(5+1)+x(103)*t(6+1)+x(107)*t(7+1)+x(111)*t(8+1)+x
(115)*t(9+1)+x(119)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(120)=-
```

```
(b04+b14+b24)*(x(100)*t(5+1)+x(104)*t(6+1)+x(108)*t(7+1)+x(112)*t(8+1)+
x(116)*t(9+1)+x(120)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x
(100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(
105)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(1
10)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(11
5)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(
120)*t(10+1));

%  Équation 2.9
g(121)=-x(121)+0;
g(122)=-x(125)+1;
g(123)=-x(129)+1;
g(124)=-x(122)+1;
g(125)=-x(126)+0;
g(126)=-x(130)+1;
g(127)=-x(123)+1;
g(128)=-x(127)+1;
g(129)=-x(131)+0;
g(130)=-x(124)+1;
g(131)=-x(128)+0;
g(132)=-x(132)+1;

g(133)=-x(133)+0;
g(134)=-x(137)+1;
g(135)=-x(141)+1;
g(136)=-x(134)+1;
g(137)=-x(138)+0;
g(138)=-x(142)+1;
g(139)=-x(135)+1;
g(140)=-x(139)+1;
g(141)=-x(143)+0;
g(142)=-x(136)+1;
g(143)=-x(140)+0;
g(144)=-x(144)+1;

g(145)=-x(145)+0;
g(146)=-x(149)+1;
g(147)=-x(153)+1;
g(148)=-x(146)+1;
g(149)=-x(150)+0;
g(150)=-x(154)+1;
g(151)=-x(147)+1;
g(152)=-x(151)+1;
g(153)=-x(155)+0;
g(154)=-x(148)+1;
g(155)=-x(152)+0;
g(156)=-x(156)+1;

g(157)=-x(157)+0;
g(158)=-x(161)+1;
g(159)=-x(165)+1;
g(160)=-x(158)+1;
g(161)=-x(162)+0;
g(162)=-x(166)+1;
```

```
g(163)=-x(159)+1;
g(164)=-x(163)+1;
g(165)=-x(167)+0;
g(166)=-x(160)+1;
g(167)=-x(164)+0;
g(168)=-x(168)+1;
```

2.4.4.9 Obtenir les solutions Pareto

De nombreux essais ont été réalisés avec diverses variations, cependant, sept essais ont été retenus. En effet, beaucoup plus d'essais ont été réalisés en faisant varier les paramètres ayant un effet sur les résultats, tels que ceux présents dans le tableau 2.7. Plusieurs autres simulations ont été réalisées mais comme elles ne généraient pas d'autres solutions de Pareto, nous n'en avons retenues que les sept présentées dans le tableau suivant. Avant de présenter les résultats obtenus, le tableau 2.7 présente, pour chaque essai effectué, le temps nécessaire pour résoudre le problème, la valeur des paramètres modifiés et le nombre de solutions de Pareto. Tel que mentionné, le nombre d'évaluations pour tous les essais est fixé à 10 000 000.

Tableau 2.7 : Nombre d'évaluations, temps, paramètres et points de Pareto des essais

Essai	**Temps (s)**	**Param (2)**	**Param (7)**	**Param (8)**	**Param (11)**	**Points de Pareto**
1	1927	0	0	0	0	5
2	2423	0.1	0	0	0	2
3	1993	10	0	0	0	6
4	2411	1	0	0	0.01	2
5	2346	1	30	5	0	4
6	3151	3	0	0	0.01	3
7	2510	1	0	0	0	4

À la suite de l'obtention des solutions de Pareto des sept essais, le graphique présenté à la figure 2.6 est réalisé afin d'avoir une vue d'ensemble des solutions en fonction de la durée du projet et de son coût. Bien que 26 solutions aient été obtenues au total, 20 points se retrouvent sur la figure 2.6 étant donné que les mêmes solutions peuvent être obtenues d'un essai à l'autre. Les points verts indiquent les réels points de Pareto lorsque les sept solutions sont mises en commun et les points rouges indiquent ceux qui sont rejetés et qui ne franchissent pas la prochaine étape.

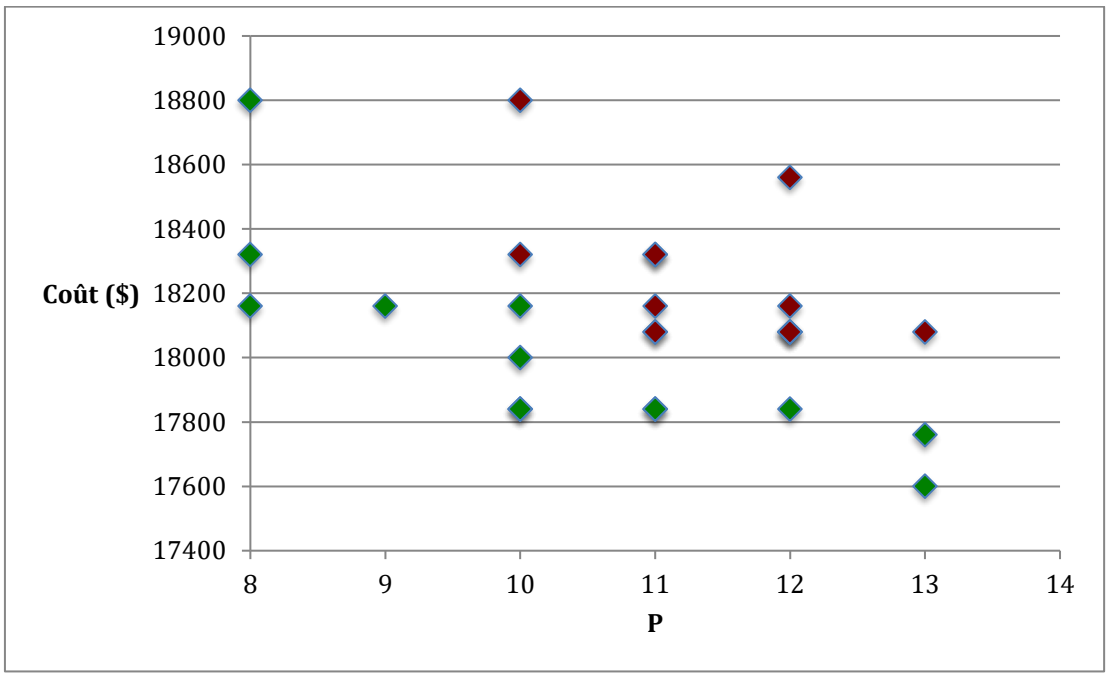

Figure 2.6 : Points de Pareto obtenus à partir des sept essais

À partir des essais retenus, soit les points verts présents à la figure 2.6, il est possible de tracer les diagrammes de Gantt de chacune des solutions. Les tableaux 2.8 à 2.20 présentent, pour chaque essai, un rappel de la valeur des fonctions f(2) et f(3), les variables binaires mises à un et la valeur des variables X169 à X173, représentant le

temps de départ de chaque activité. Les figures 2.7 à 2.19 présentent les diagrammes de Gantt de chaque solution retenue, en fonction des activités attribuées à chaque ressource.

Les compétences utilisées par les ressources sont aussi indiquées pour chacune des activités. De plus, chaque essai suivant présente une ligne verticale noire à son extrémité de droite, représentant ainsi la date prévue de fin du projet. Chaque unité de temps où une ressource est inactive est marquée par un encadré noir.

Tableau 2.8 : Résultats de l'essai 2.1

Essai	2.1
f(2) (jours)	10
f(3) ($)	17840
Variables binaires à 1	3,4,46,48,82,83,84,105,127,132,136,142,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,7,10

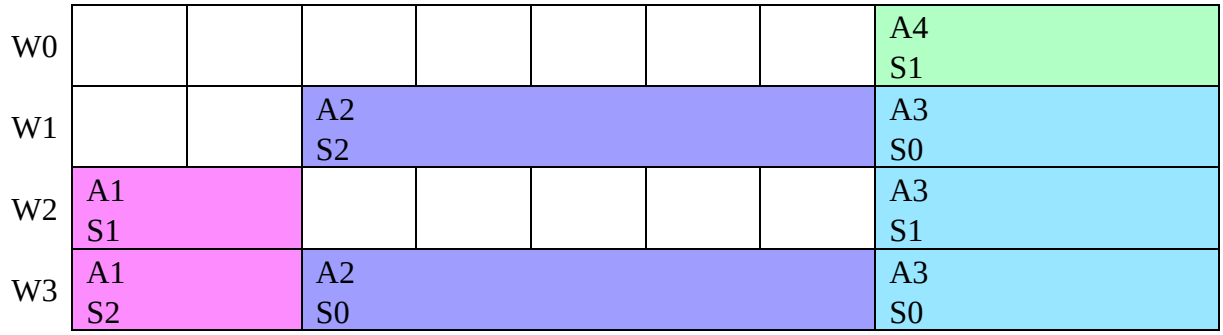

Figure 2.7 : Diagramme de Gantt de l'essai 2.1

Tableau 2.9 : Résultats de l'essai 2.2

Essai	2.2
f(2) (jours)	11
f(3) ($)	17840
Variables binaires à 1	3,4,46,48,82,83,84,109,127,132,136,142,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,8,11

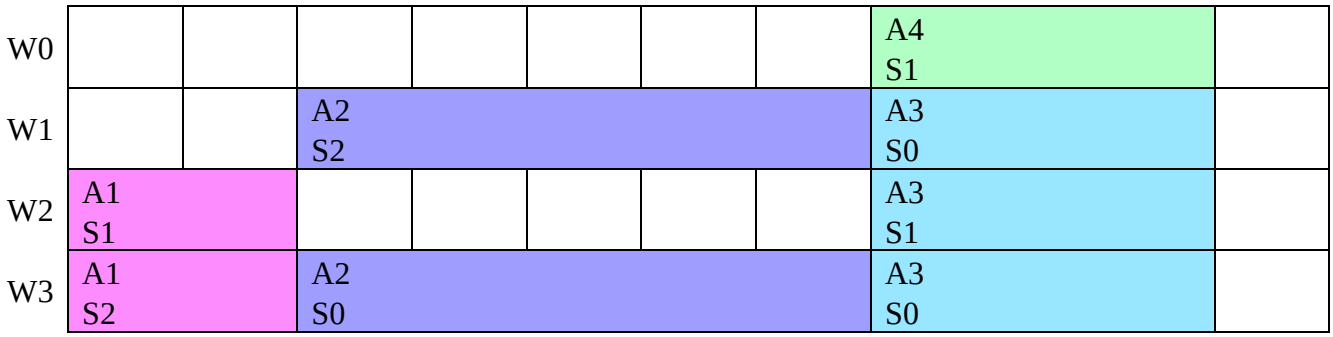

Figure 2.8 : Diagramme de Gantt de l'essai 2.2

Tableau 2.10 : Résultats de l'essai 3.1

Essai	3.1
f(2) (jours)	13
f(3) ($)	17600
Variables binaires à 1	3,4,47,48,82,83,84,119,127,132,135,144,146,148,151,163
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,10,13

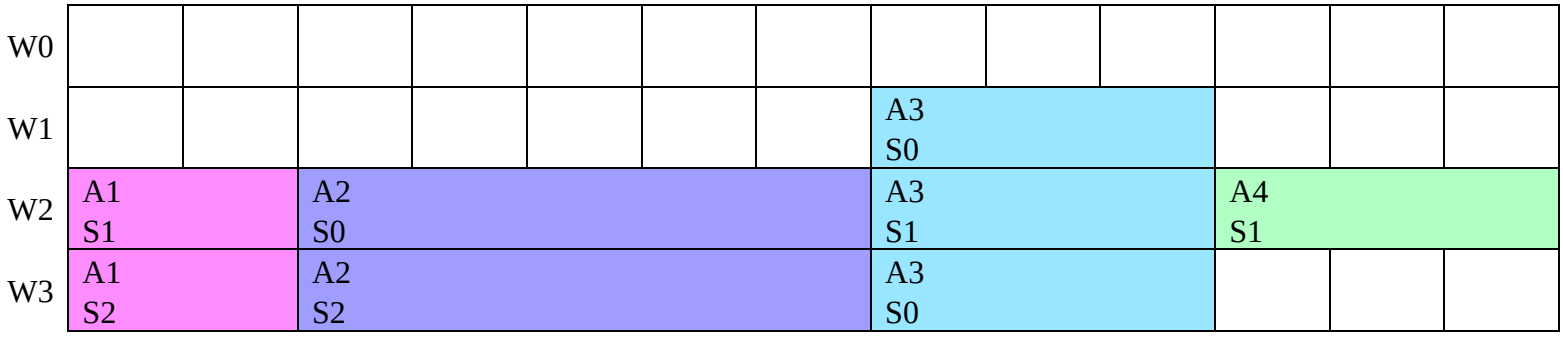

Figure 2.9 : Diagramme de Gantt de l'essai 3.1

Tableau 2.11 : Résultats de l'essai 3.4

Essai	3.4
f(2) (jours)	10
f(3) ($)	18160
Variables binaires à 1	1,2,39,40,82,83,84,97,125,130,135,144,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,0,7,5,10

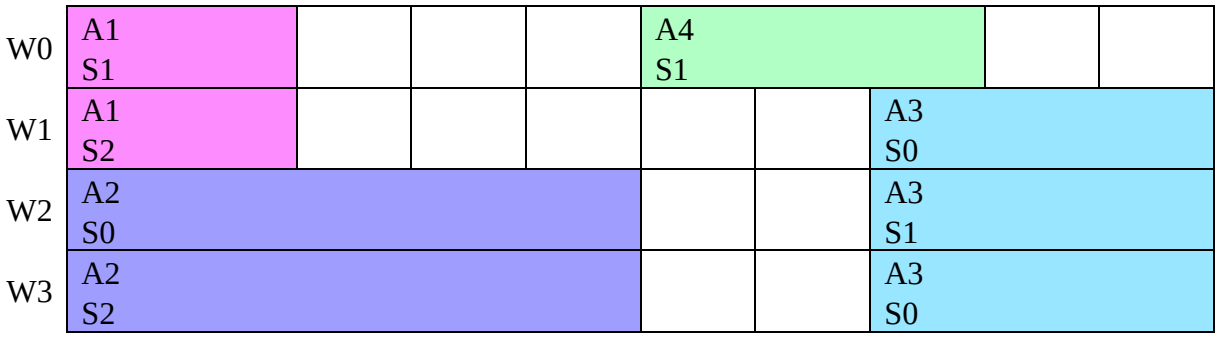

Figure 2.10 : Diagramme de Gantt de l'essai 3.4

Tableau 2.12 : Résultats de l'essai 3.5

Essai	3.5
f(2) (jours)	12
f(3) ($)	17840
Variables binaires à 1	3,4,47,48,82,83,84,113,127,132,135,144,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,9,12

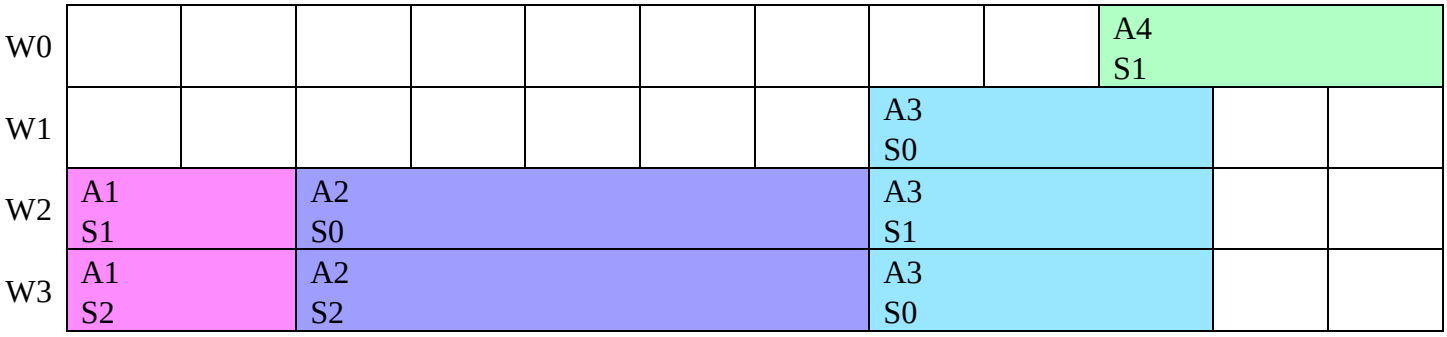

Figure 2.11 : Diagramme de Gantt de l'essai 3.5

Tableau 2.7 : Résultats de l'essai 3.6

Essai	3.6
f(2) (jours)	13
f(3) ($)	17760
Variables binaires à 1	1,4,47,48,82,83,84,119,125,132,135,144,146,148,151,163
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,10,13

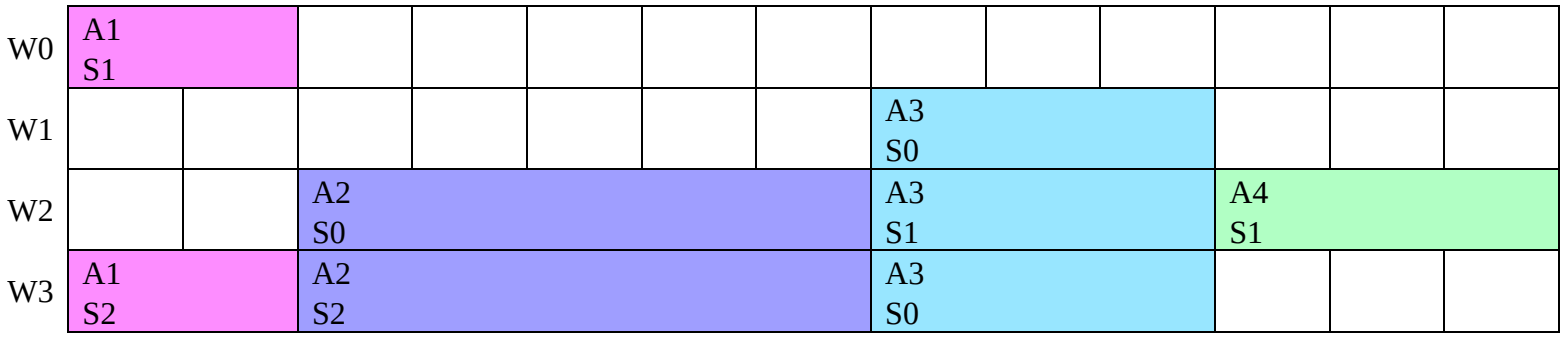

Figure 2.12 : Diagramme de Gantt de l'essai 3.6

Tableau 2.8 : Résultats de l'essai 5.2

Essai	5.2
f(2) (jours)	10
f(3) ($)	18000
Variables binaires à 1	2,3,47,48,81,82,84,107,127,130,135,144,146,148,149,163
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,7,10

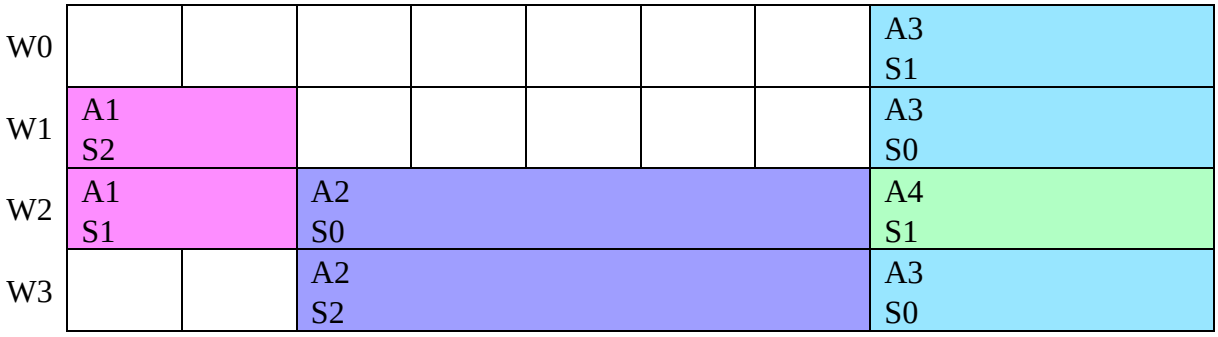

Figure 2.13 : Diagramme de Gantt de l'essai 5.2

Tableau 2.9 : Résultats de l'essai 5.3

Essai	5.3
f(2) (jours)	11
f(3) ($)	17840
Variables binaires à 1	3,4,47,48,81,82,84,107,127,132,135,144,146,148,149,163
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,7,11

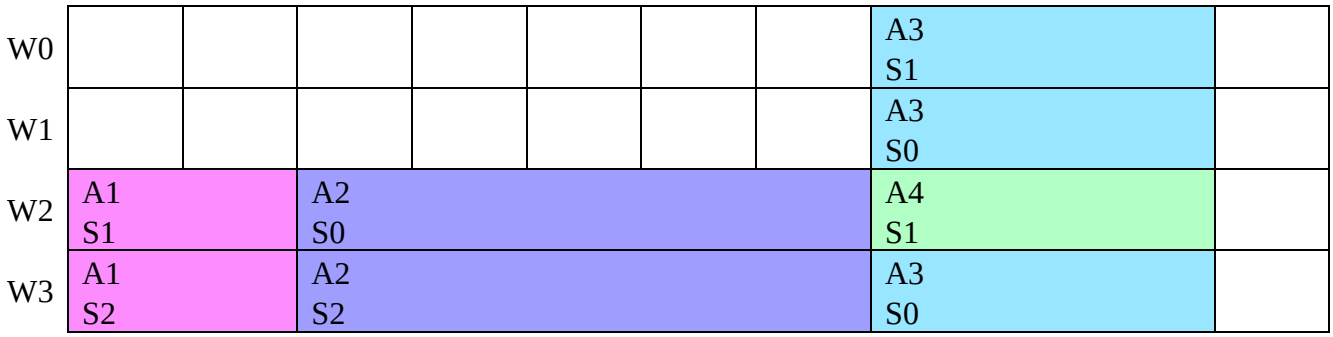

Figure 2.14 : Diagramme de Gantt de l'essai 5.3

Tableau 2.10 : Résultats de l'essai 5.4

Essai	5.4
f(2) (jours)	10
f(3) ($)	17840
Variables binaires à 1	3,4,47,48,81,82,84,107,127,132,135,144,146,148,149,163
Valeur des variables X169 à X173 (t1 à t5)	0,2,7,7,10

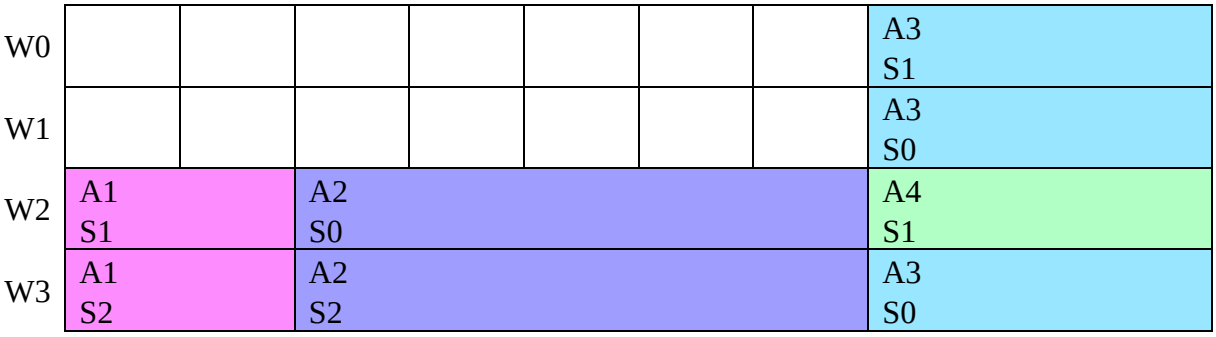

Figure 2.15 : Diagramme de Gantt de l'essai 5.4

Tableau 2.11 : Résultats de l'essai 7.1

Essai	7.1
f(2) (jours)	8
f(3) ($)	18800
Variables binaires à 1	2,3,37,40,74,75,76,97,127,130,136,141,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,0,5,5,8

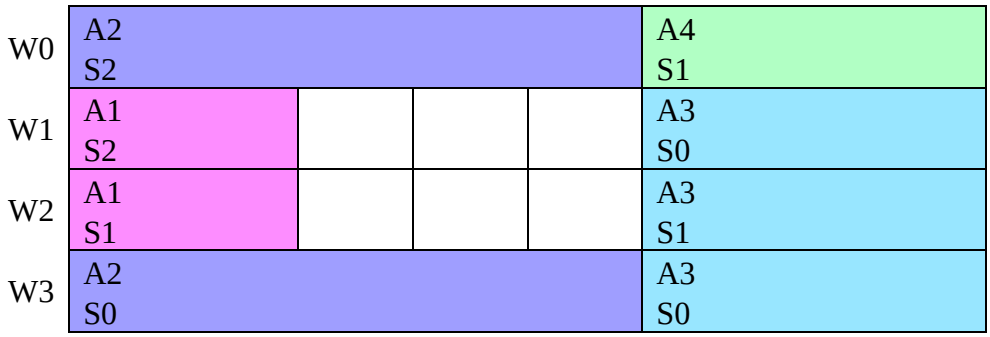

Figure 2.16 : Diagramme de Gantt de l'essai 7.1

Tableau 2.12 : Résultats de l'essai 7.2

Essai	7.2
f(2) (jours)	8
f(3) ($)	18320
Variables binaires à 1	1,3,38,40,74,75,76,97,127,129,136,142,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,0,5,5,8

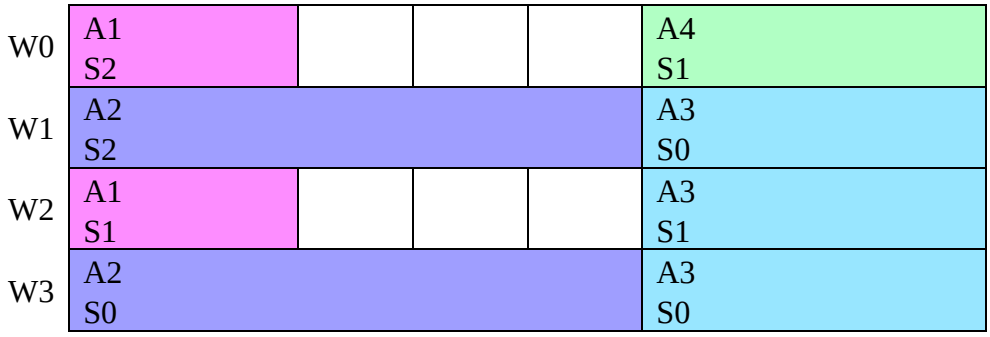

Figure 2.17 : Diagramme de Gantt de l'essai 7.2

Tableau 2.13 : Résultats de l'essai 7.3

Essai	7.3
f(2) (jours)	9
f(3) ($)	18160
Variables binaires à 1	1,2,39,40,74,75,76,101,125,130,135,144,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,0,5,6,9

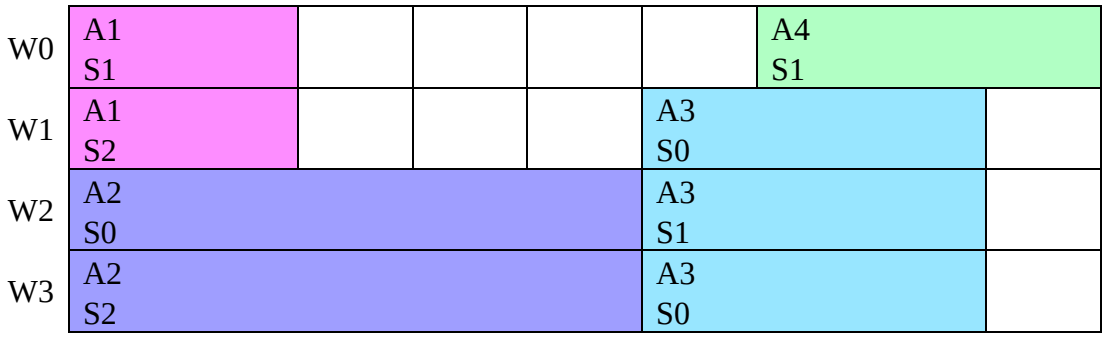

Figure 2.18 : Diagramme de Gantt de l'essai 7.3

Tableau 2.20 : Résultats de l'essai 7.4

Essai	7.4
f(2) (jours)	8
f(3) ($)	18160
Variables binaires à 1	1,2,39,40,74,75,76,97,125,130,135,144,146,148,151,161
Valeur des variables X169 à X173 (t1 à t5)	0,0,5,5,8

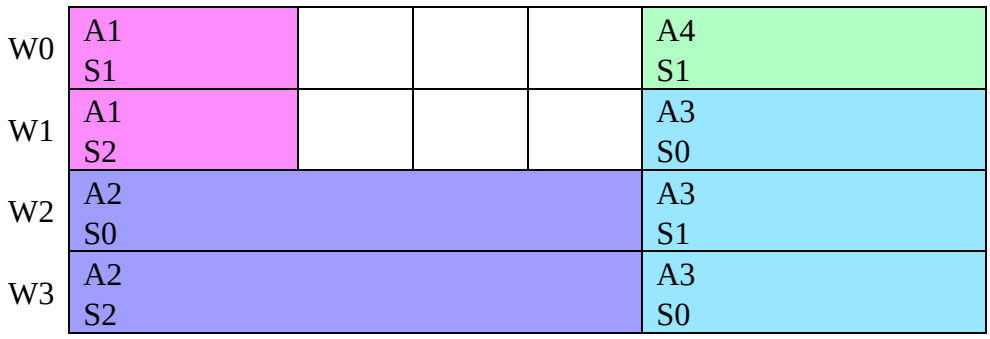

Figure 2.19 : Diagramme de Gantt de l'essai 7.4

En observant les figures et les tableaux présentés ci-haut, quelques éléments ressortent. D'abord, en observant la figure 2.6, il est possible de remarquer que 11 solutions optimales ressortent des essais, alors que 13 diagrammes de Gantt sont présentés. En effet, les solutions obtenues par les essais 2.1 et 5.4 ainsi que celles obtenues par les essais 2.2 et 5.3 présentent, pour chaque couple, les mêmes résultats pour les fonctions f(2) et f(3). Cependant, les variables binaires actives sont différentes d'un essai à l'autre, présentant ainsi des configurations différentes.

Ensuite, il est possible de remarquer que les essais 2.1 et 2.2 ainsi que les essais 5.3 et 5.4 possèdent, pour chaque couple d'essais, des diagrammes de Gantt identiques. Pour chaque couple d'essais, les variables binaires actives sont les mêmes, cependant, pour les essais 2.2 et 5.3, la variable t5 est égale à 11 tandis que pour les essais 2.1 et 5.4, la variable t5 est à 10. Comme chaque couple d'essais présente des affectations pour les ressources et des compétences identiques, mais que les valeurs des variables t5 sont différentes, les essais 2.1 et 5.4 sont automatiquement supérieurs aux essais 2.2 et 5.3. Ces deux derniers essais sont donc retirés de l'analyse avec Prométhée.

2.4.5 Choix d'une solution avec Prométhée

Les 13 solutions présentées à la section 2.4.4.9, mis à part les solutions obtenues à partir des essais 2.2 et 5.3, sont prises en considération afin d'être comparées les unes avec les autres à l'aide de Prométhée. Quelques critères seront utilisés et des simulations seront effectuées en attribuant une importance différente à chaque critère.

Il est à noter que le choix des critères utilisés ainsi des simulations effectuées est laissé au soin de l'utilisateur. Celui-ci peut adapter ces éléments selon ses besoins et ses préférences.

2.4.5.1 Choisir les critères de comparaison des solutions Pareto

Trois critères sont utilisés afin de comparer les solutions à l'aide du logiciel d'aide à la prise de décision Prométhée. D'abord, le temps et le coût de chaque solution sont pris en considération étant donné qu'il s'agit des deux critères sur lesquels l'obtention des solutions de Pareto à partir de Midaco se base.

Le troisième critère de comparaison désigné dans Prométhée est celui du temps perdu. Celui-ci est utilisé afin de minimiser le temps d'inactivité d'une ressource entre

deux tâches du projet. En observant les diagrammes de Gantt obtenu, il est possible de constater que ce temps d'inactivité se retrouve dans plusieurs solutions. Par exemple, pour l'essai 3.4, la ressource W_0 est utilisée pour la première activité durant deux jours dès le début du projet et est utilisée, par la suite, pour la quatrième activité à partir du septième jour. La ressource W_1, quant à elle, est utilisée pour la première activité durant les deux premiers jours du projet et participe ensuite à la troisième activité au septième jour du projet. Pour les ressources W_2 et W_3, elles sont utilisées pour la seconde activité du projet durant les cinq premiers jours du projet et sont réutilisées du septième au dixième jour pour la troisième activité. Donc, pour l'essai 3.4, le temps perdu est de 12 unités de temps. Bref, il s'agit de minimiser le temps d'inactivité d'une ressource entre deux activités non successives.

En observant le diagramme de Gantt de l'essai 3.1 présenté à la figure 2.9, il est possible de constater que la ressource W_0 est inutilisée pour la totalité du projet. Un utilisateur cherchant à minimiser la quantité de ressources utilisées pour son projet pourrait utiliser ce critère de décision afin de comparer les solutions dans Prométhée. En situation réelle, les seuils de préférence ou d'indifférence associés aux fonctions de préférence seraient définies par le ou les décideurs du projet, ou par son planificateur, aidé par l'analyste. Tout dépendant de l'entreprise dans laquelle elle se trouve, une même personne pourrait d'ailleurs être en charge de ces trois rôles.

2.4.5.2 Modéliser les préférences des décideurs

Prométhée propose de nombreux outils afin de comparer les différentes solutions entre elles. Uniquement quelques-unes sont utilisées dans ce cas-ci, cependant, l'utilisateur est en mesure d'explorer et d'exploiter le logiciel de la manière dont il le désire.

Afin de choisir la solution finale, et dans le but de présenter un classement global pour le projet entier, les critères sont considérés à poids égaux lors de la comparaison des

essais. La solution finale est choisie avec des critères à poids égaux étant donné qu'il s'agit d'un exemple didactique et qu'aucun entretien n'a été réalisé avec un gestionnaire de projet afin d'obtenir des informations supplémentaires sur un projet quelconque. Cependant, afin de maximiser l'utilisation de Prométhée dans le cadre de cette recherche, des simulations avec différents poids sont réalisées.

Les différents outils utilisés pour analyser les solutions sont l'arc-en-ciel Prométhée, présentant un résumé des critères positifs et négatifs de chaque essai, les «walking weights», présentant un classement des essais selon l'importance de chaque critère et les intervalles de stabilité, présentant l'intervalle selon lequel chaque essai se déplace dans le classement des solutions.

Tous ces outils sont utilisés pour la simulation présentant des critères à poids égaux. Pour les autres simulations, uniquement l'outil des «walking weights» est utilisé. D'autres outils sont disponibles pour visualiser et comparer les flux tels que présentés à la section 2.3.2.2, cependant, en guise de simplicité et de facilité de compréhension, ceux-ci ne sont pas analysés dans la section 2.4.5.3.

Le tableau 2.21 présente le poids des critères pour les différentes simulations réalisées.

Tableau 2.21 : Poids des critères pour les simulations à l'aide de Prométhée

Simulation	1	2	3	4	5	6	7	8	9
Poids de la durée (%)	100	0	0	50	0	50	75	25	33
Poids du coût (%)	0	100	0	50	50	0	25	75	33
Poids du temps perdu (%)	0	0	100	0	50	50	0	0	33

2.4.5.3 Évaluer les solutions Pareto selon les critères retenus

D'abord, avant de présenter les solutions obtenues à l'aide de Prométhée, le tableau 2.22 présente un résumé des données des solutions retenues.

Tableau 2.22 : Données des solutions de Pareto retenues

Planification	Essai	Durée (jours)	Coût ($)	Temps perdu (unité de temps)
1	2.1	10	17840	5
2	3.1	13	17600	0
3	3.4	10	18160	12
4	3.5	12	17840	0
5	3.6	13	17760	0
6	5.2	10	18000	5
7	5.4	10	17840	0
8	6.1	8	18800	6
9	6.2	8	18320	6
10	6.3	9	18160	7
11	6.4	8	18160	6

Les trois premières simulations, telles que présentées au tableau 2.21, sont probablement les trois éléments principaux à vérifier à la suite de l'obtention des solutions de Pareto. Il s'agit du classement des solutions lorsque la durée est l'unique critère important, du classement lorsque le coût du projet est le seul critère important et du classement lorsque le temps perdu est important à 100%. Ces trois simulations n'auraient pas été nécessaires étant donné qu'il est possible de constater ces trois classements en observant le tableau 2.22. Par contre, le fait de réaliser ces simulations

permet d'obtenir un classement visuel plus facile à interpréter. Les figures 2.20 à 2.22 présentent les classements obtenus pour ces simulations à l'aide de l'outil des «walking weights».

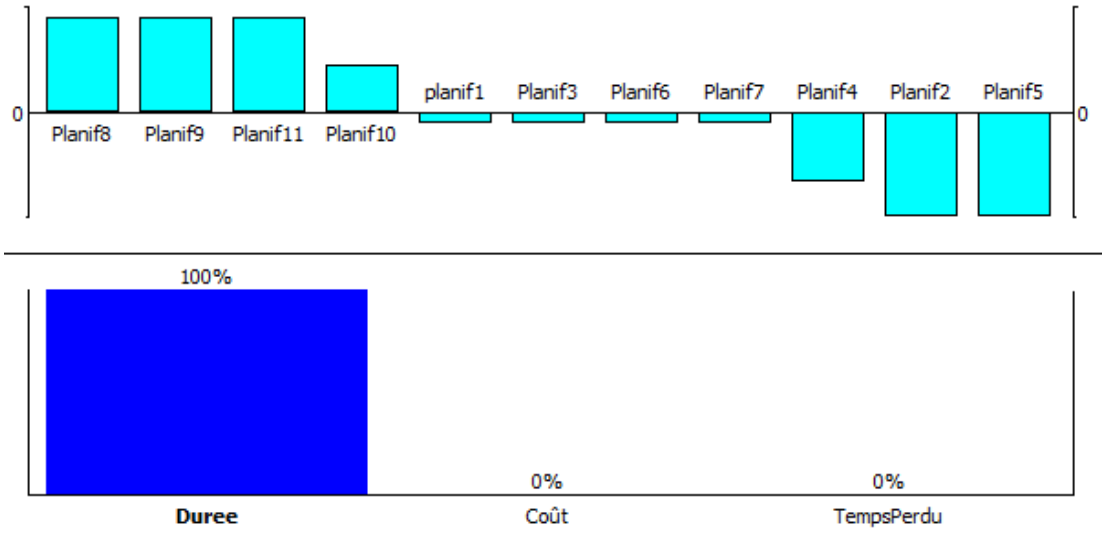

Figure 2.20 : Classement pour un poids sur la durée de 100%

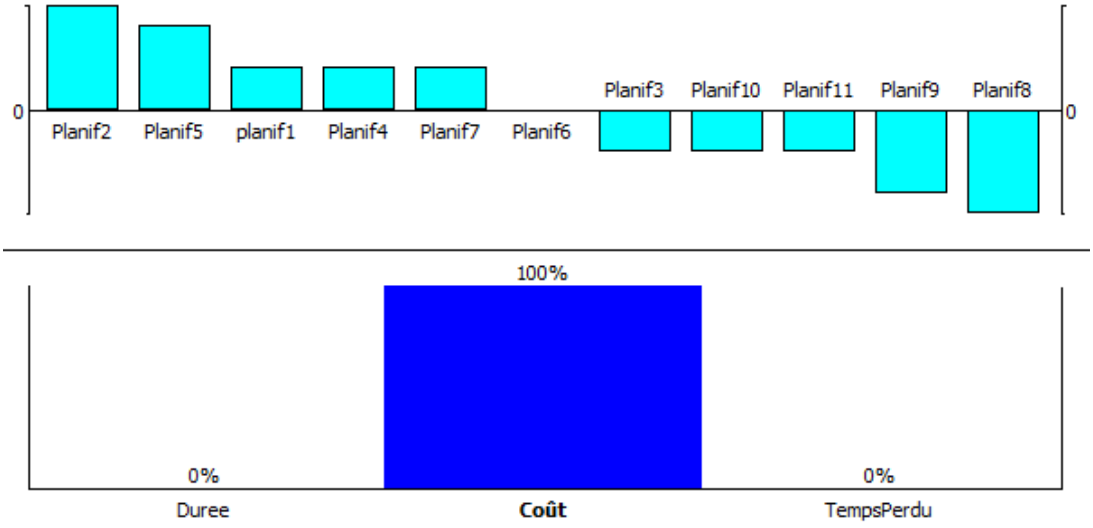

Figure 2.21 : Classement pour un poids sur le coût de 100%

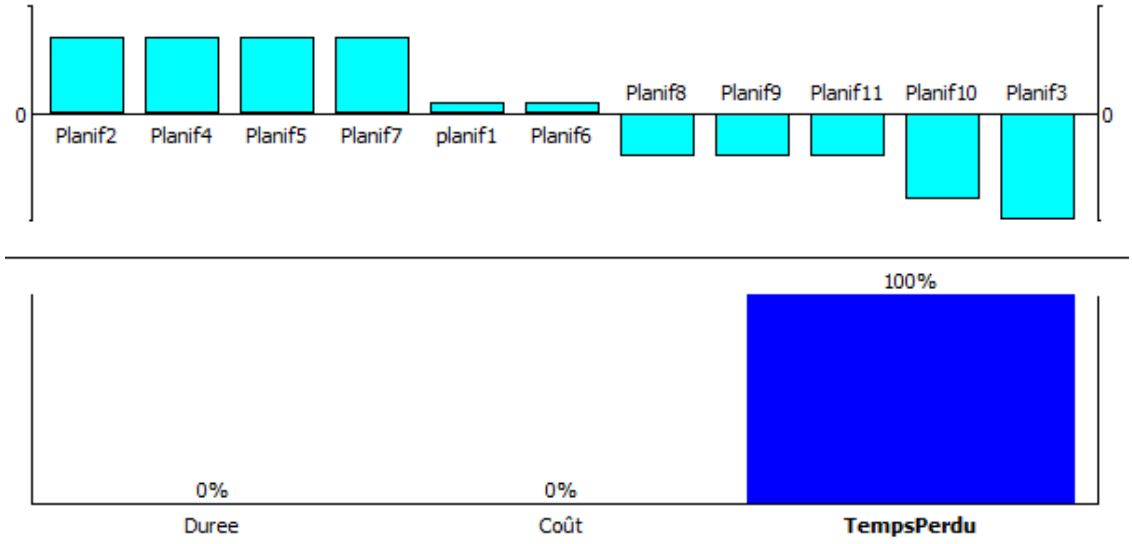

Figure 2.22 : Classement pour un poids sur le temps perdu de 100%

Pour les figures 2.20 à 2.22, le classement de la meilleure solution à la pire est présenté de gauche à droite. Lorsque la bande est supérieure à zéro, il s'agit d'une solution acceptable tandis que lorsque la bande se trouve sous zéro, il s'agit d'une moins bonne solution. Plus la valeur de la bande est élevée, plus la solution est intéressante.

En ce qui concerne le classement de la figure 2.20, les meilleures solutions selon la durée du projet sont les planifications 8, 9 et 11 étant donné qu'elles ont toutes une durée de huit jours, soit la durée la plus faible obtenue parmi les solutions générées. Il s'agit d'ailleurs de la durée la plus faible qu'il est possible d'obtenir selon les données du problème. Sans surprise, les planifications 2 et 5 se retrouvent au dernier rang, avec la durée maximale permise selon le code Matlab utilisé, soit 13 jours.

Concernant le classement pour le coût total du projet, présenté à la figure 2.21, la deuxième planification est la moins dispendieuse, pour un total de 17600$, suivie de la cinquième planification, à 17760$. Les planifications 9 et 8 sont les moins intéressantes selon le coût, avec des coûts respectifs de 18320$ et 18800$. Il est possible de constater,

en observant les classements des figures 2.20 et 2.21, que les solutions proposant une durée minimale sont celles proposant le coût le plus élevé, et vice versa. Il s'agit d'une constatation intéressante étant donné qu'elle montre que, bien qu'une solution soit très bien classée selon un critère, elle n'est pas nécessairement la meilleure pour chacun d'entre eux. Cela démontre l'importance de bien prendre en considération ses besoins et d'accorder une attention particulière au choix des critères de décision.

En observant la figure 2.22, il n'est pas surprenant de constater que les meilleures solutions concernant le temps perdu sont celles où aucune ressource n'est inactive entre deux activités, soit les planifications 2, 4, 5 et 7. Le temps perdu de ces solutions est nul, il s'agit de la valeur souhaitée pour optimiser ce critère.

Les trois simulations suivantes ont été réalisées en prenant en considération deux critères à poids égaux, accordant ainsi une importance nulle au troisième d'entre eux. Les figures 2.23 à 2.25 présentent les classements pour ces simulations.

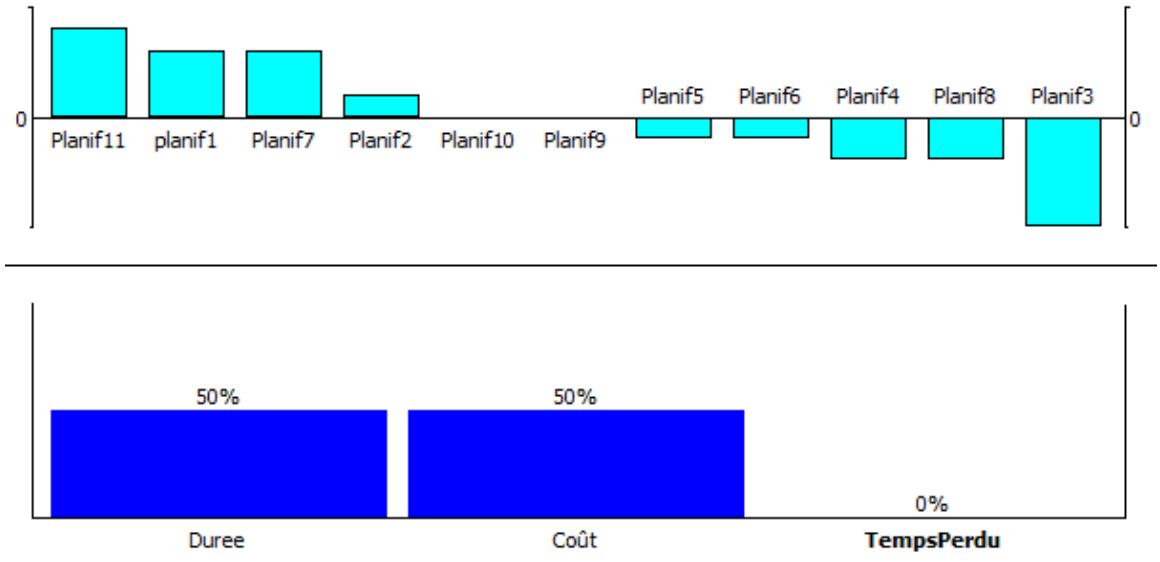

Figure 2.23 : Classement pour des poids de 50% sur la durée et le coût du projet

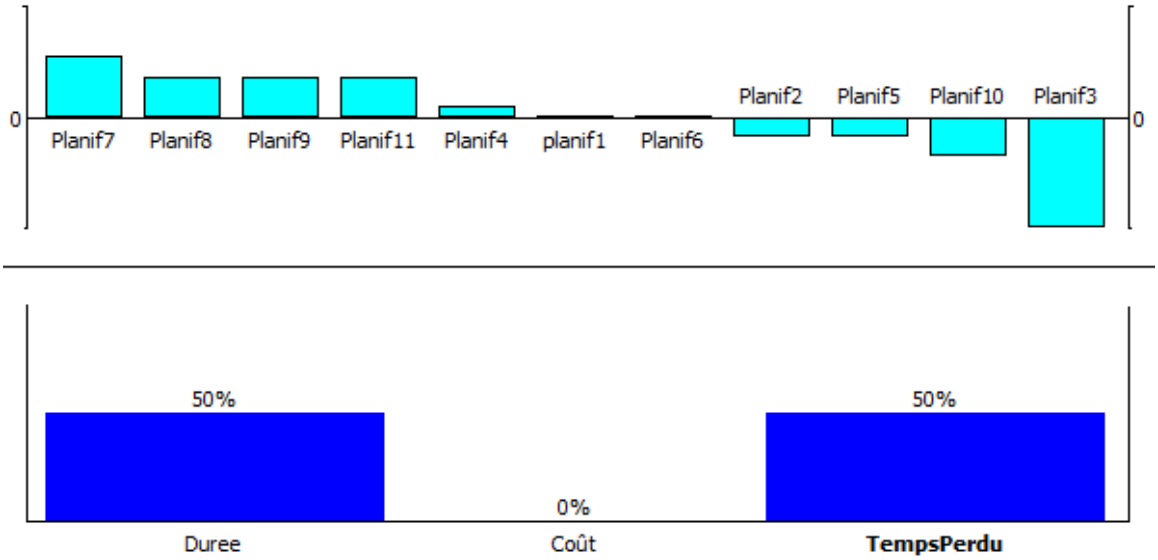

Figure 2.24 : Classement pour des poids de 50% sur la durée et le temps perdu du projet

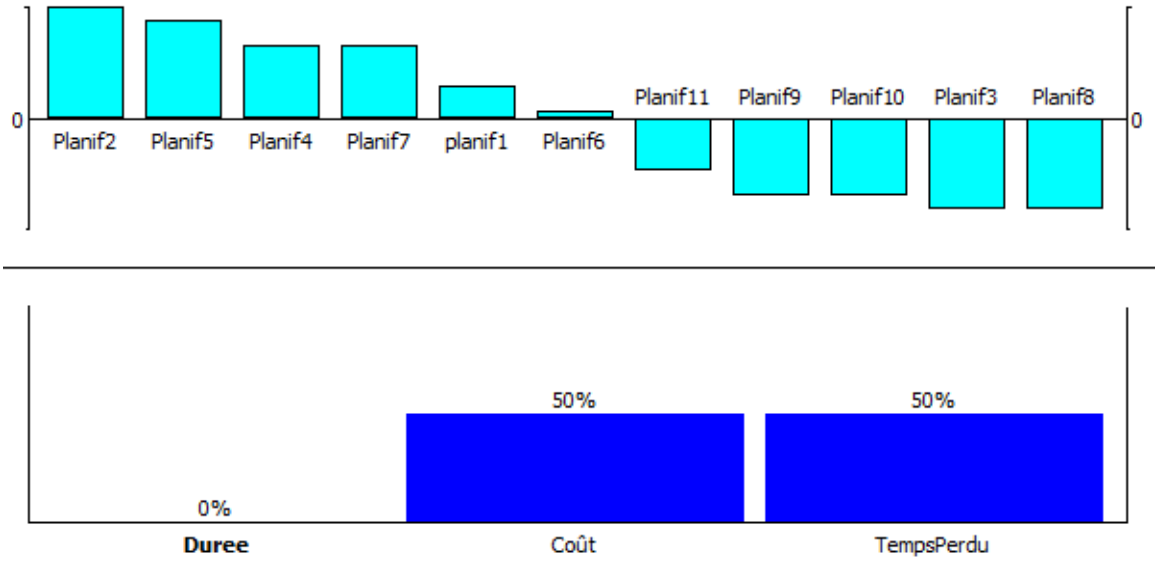

Figure 2.25 : Classement pour des poids de 50% sur le coût et le temps perdu du projet

Lorsque la durée et le coût du projet sont les deux critères considérés avec un poids de 50 % chacun, comme la figure 2.23 le présente, les planifications 11 et 1 sont à privilégier tandis que la troisième planification est celle à éviter.

Dans le cas où la durée et le temps perdu du projet sont les deux critères considérés à parts égales, comme la figure 2.24 le présente, la septième planification est la meilleure alors que la troisième planification est la pire.

Puis, lorsque le coût du projet et son temps perdu sont les deux critères pris en considération avec une importance de 50% pour chacun d'eux, la deuxième planification obtient le meilleur score tandis que la huitième planification se classe au dernier rang.

Il peut être intéressant de constater, en observant les figures 2.23 à 2.25, que les bandes pour les planifications 1 et 7 se situent toujours au-dessus de zéro tandis que les bandes se situent toujours sous zéro pour l’essai 3 et sont égales ou inférieures à zéro pour l’essai 10. Il sera possible de vérifier, lors de la présentation de la simulation où les trois critères ont une importance égale, si cela se reflète dans les résultats.

Les deux simulations suivantes ont été réalisées en prenant en considération la durée et le coût du projet avec des poids inégaux. La figure 2.26 présente le classement lorsque la durée et le coût ont des poids de 75% et 25% alors que la figure 2.27 présente le classement lorsque la durée et le coût ont des importances respectives de 25% et 75%.

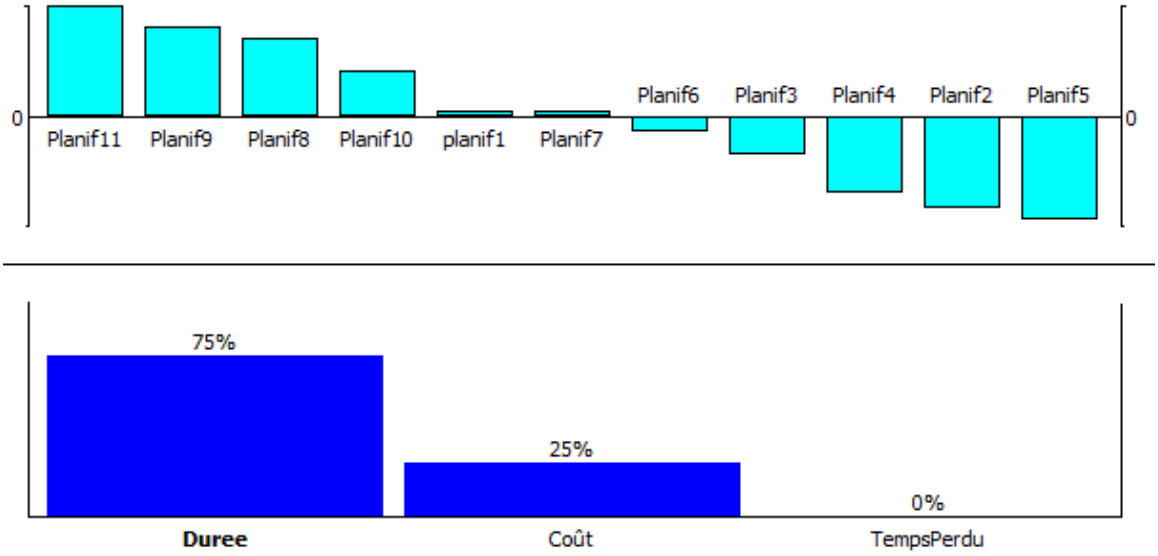

Figure 2.26 : Classement pour des poids de 75% et 25% sur la durée et le coût du projet

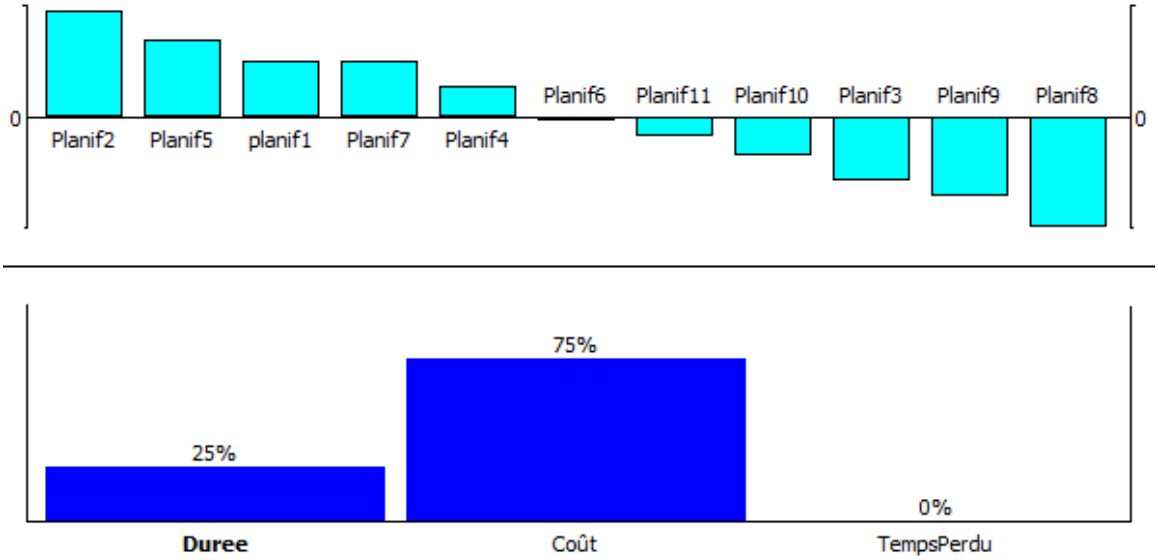

Figure 2.27 : Classement pour des poids de 25% et 75% sur la durée et le coût du projet

Tel que la figure 2.26 le présente, lorsque la durée et le coût ont des poids de 75% et 25%, les planifications à privilégier sont la onzième et la neuvième, alors que celles à éviter sont les deuxième et cinquième planifications.

En ce qui concerne le classement présenté à la figure 2.27, les meilleures planifications sont la deuxième et la cinquième, alors que les moins intéressantes sont la neuvième et la huitième.

La majorité des solutions intéressantes de la figure 2.26 se retrouvent en moins bonne position sur la figure 2.27, alors que les solutions les moins intéressantes présentées à la figure 2.26 se retrouvent en meilleure position à la figure 2.27. De plus, en comparant ces résultats avec ceux obtenus pour une durée et un coût avec des poids égaux, soit ceux présentés à la figure 2.23, aucune ressemblance marquante ne se présente. À la figure 2.26, la meilleure solution est la même que celle présentée à la figure 2.23, alors qu'à la figure 2.27 et à la figure 2.23, les planifications 8 et 3 se retrouvent parmi les trois dernières solutions. Cela montre donc que, bien que les mêmes critères soient pris en considération pour trois simulations différentes, le choix de la pondération des critères est un élément non négligeable et apporte une importance diversification des résultats.

La dernière simulation réalisée est celle où les trois critères ont une importance équivalente, soit chacun un poids un 33%. La figure 2.28 présente d'abord le classement à l'aide des «walking weights», tel que présenté pour les simulations précédentes.

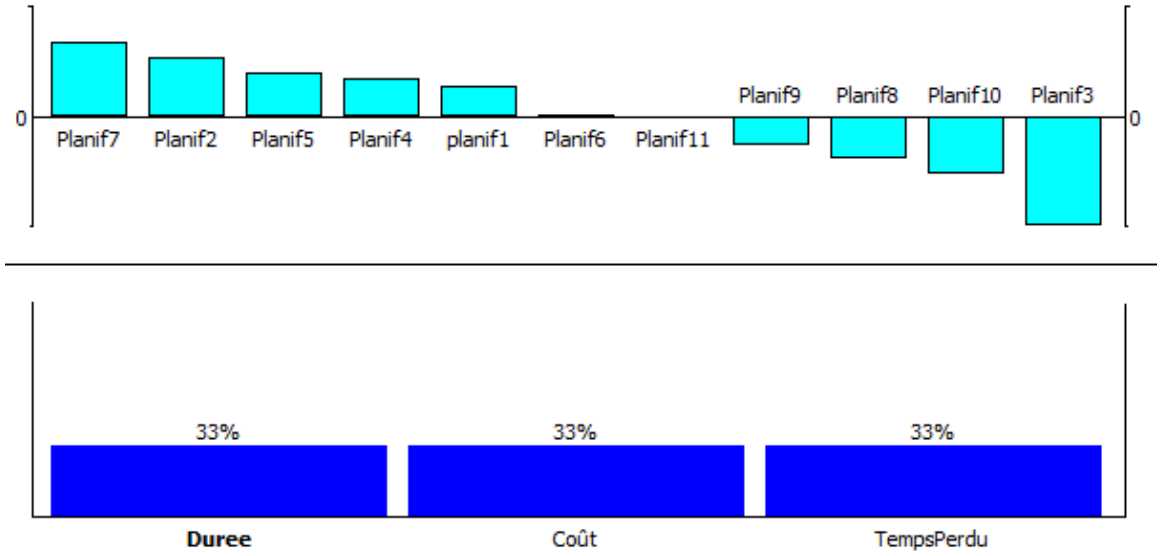

Figure 2.28 : Classement pour des poids de 33% pour tous les critères

Le classement de la figure 2.28 permet de constater que les deux meilleures planifications sont la septième et la deuxième tandis que les trois pires solutions sont la huitième, la dixième et la troisième. Cela confirme donc la constatation effectuée sous la figure 2.25 pour la planification 7, étant la meilleure et pour les planifications 10 et 3, étant les pires lorsque les trois critères sont à poids égaux. La première planification, bien qu'elle soit supérieure à zéro pour les essais présentés aux figures 2.23 à 2.25, où deux critères ont un poids de 50% et où le poids du troisième critère est nul, ne se retrouve pas parmi les meilleures solutions sur la figure 2.27. Cependant, elle est tout de même supérieure à zéro et se classe au cinquième rang.

Le diagramme obtenu à l'aide de l'outil de l'arc-en-ciel Prométhée est présenté à la figure 2.29.

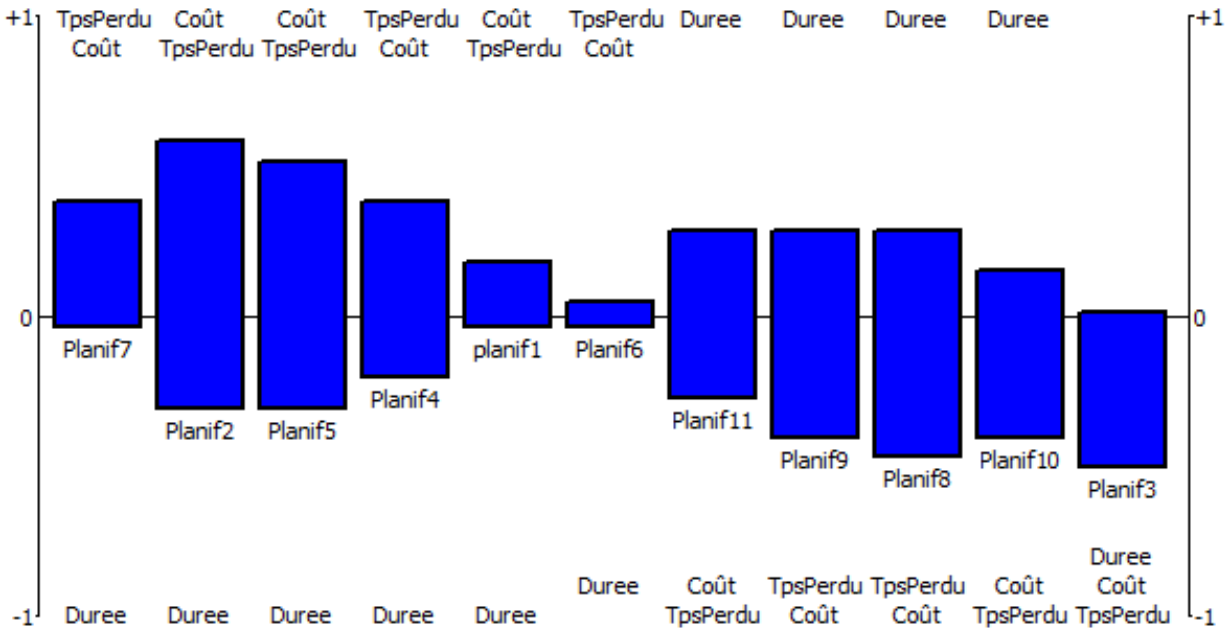

Figure 2.29 : Arc-en-ciel Prométhée pour des poids de 33% pour tous les critères

Cet outil est intéressant puisque, bien qu'il présente le classement des différentes simulations, il est aussi possible de déterminer les critères faisant en sorte qu'une planification obtienne un score faible ou élevé. Par exemple, pour la septième planification, soit la plus intéressante, le score positif est dû au temps perdu et au coût, alors que le score négatif est dû à la durée. En effet, la septième planification se classe plutôt bien pour les simulations du coût et du temps perdu présentées aux figures 2.21 et 2.22 ainsi qu'environ au centre de la distribution pour la simulation de la durée, présentée à la figure 2.20. La deuxième planification se classe au second rang et, en observant sa bande à la figure 2.29, il est possible de remarquer qu'elle obtient un score positif plus élevé que la neuvième planification. En observant la figure 2.25 où le coût et le temps perdu sont les deux critères considérés, la seconde planification se classe effectivement devant la septième. Par contre, la deuxième planification se classe au dernier rang selon sa durée, la faisant ainsi chuter au second rang pour le classement présenté à la figure 2.29.

Les figures 2.30 à 2.32 présentent, pour chaque critère, les intervalles pour lesquels un classement reste inchangé. Cela permet d'observer, pour un critère, l'évolution du classement selon la variation du poids du critère.

Pour ces trois figures, il est possible d'observer l'évolution du classement des planifications selon l'évolution du poids de chaque critère. Le classement pour un poids donné est visible en traçant une ligne verticale sur la figure observée. La meilleure solution est positionnée vers le haut de la figure tandis que la moins bonne solution est la plus basse de la figure. Une ligne verticale est présente à 33%, représentant le classement lorsque tous les critères ont un poids de 33%, soit la solution analysée dans ce cas. Évidemment, le classement est le même sur la ligne verticale pour chacune des trois figures. La planification 9 est celle prise en exemple afin d'analyser les trois figures représentant l'intervalle de confiance. Le même principe s'applique pour les autres planifications.

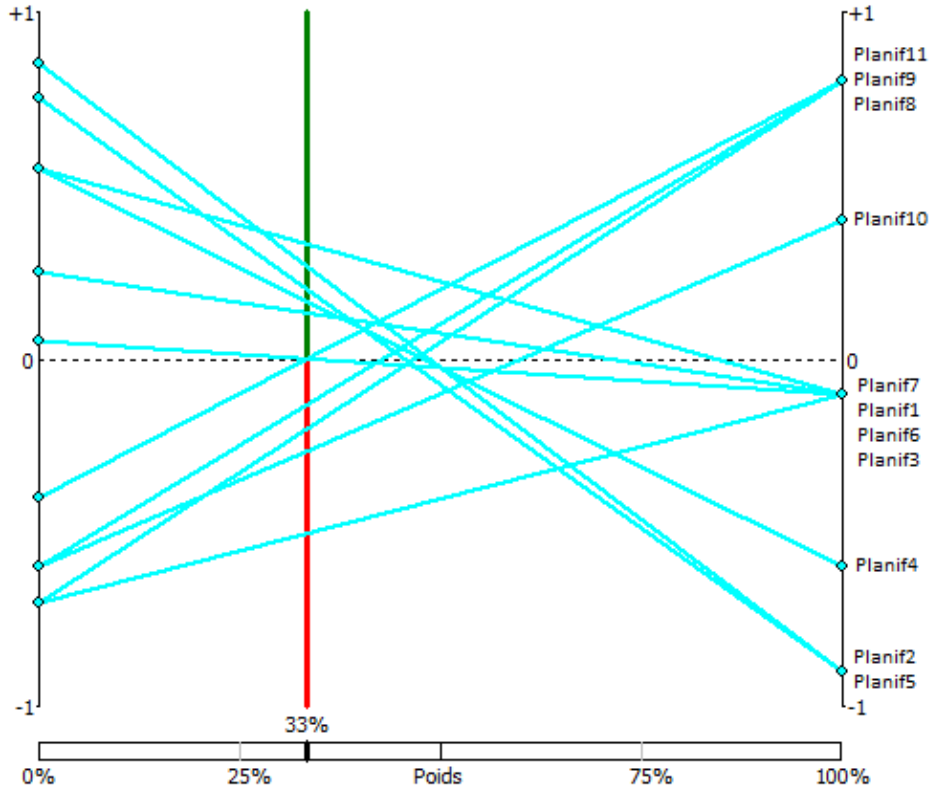

Figure 2.30 : Intervalles de stabilité pour la durée avec des poids égaux

À poids égaux, la septième planification se classe au premier rang, comme le confirme la figure 2.28. En observant la figure 2.30, il est possible de constater que cette planification conserve son premier rang lorsque le poids de la durée se situe entre 30% et 50%. Lorsque le poids de la durée est nul, la planification 7 se trouve au troisième rang, à égalité avec la planification 4. Il est possible de confirmer cette affirmation en observant la figure 2.25. Pour un poids variant entre 1% et 20%, la planification 7 se retrouve seule au troisième rang et, pour un poids d'environ 25%, elle se retrouve à égalité au second rang avec la cinquième planification. À 30%, elle se classe au premier rang à égalité avec la planification 2. Lorsque le poids de la durée est de 100%, la septième planification se classe au troisième rang. Il est à noter que, pour un poids de 100%, les planifications 7, 1, 6 et 3 se retrouvent toutes au troisième rang, comme la figure 2.20 le confirme. Donc,

lorsque le poids de la durée se situe entre 30% et 50%, la septième planification est la meilleure.

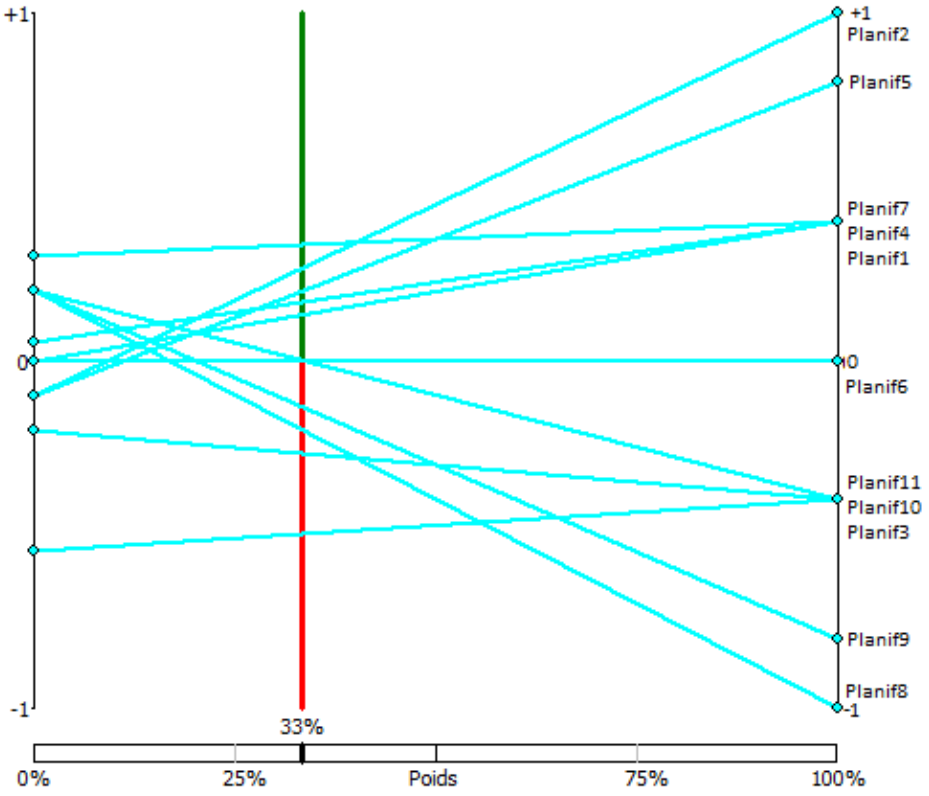

Figure 2.31 : Intervalles de stabilité pour le coût avec des poids égaux

La figure 2.31 présente les intervalles de stabilité pour le coût du projet. À poids égaux, la septième planification se classe au premier rang, tel que la figure 2.28 le confirme. Lorsque l'importance du coût est nulle, la planification 7 se classe au premier rang, tel que la figure 2.24 le montre. Sur cette même figure, il est possible d'observer que la valeur de la bande de la planification 9 est supérieure à celles des planifications 8, 9 et 11 se trouvant au second rang. Cela se reflète sur la figure 2.31, où la planification 7 conserve son premier rang jusqu'à ce que le coût atteigne une importance d'environ 40%. Lorsque le poids du coût atteint 100%, la planification 7 se trouve au troisième rang, à

égalité avec les planifications 4 et 1. Il est possible de confirmer ce fait par l'observation de la figure 2.21.

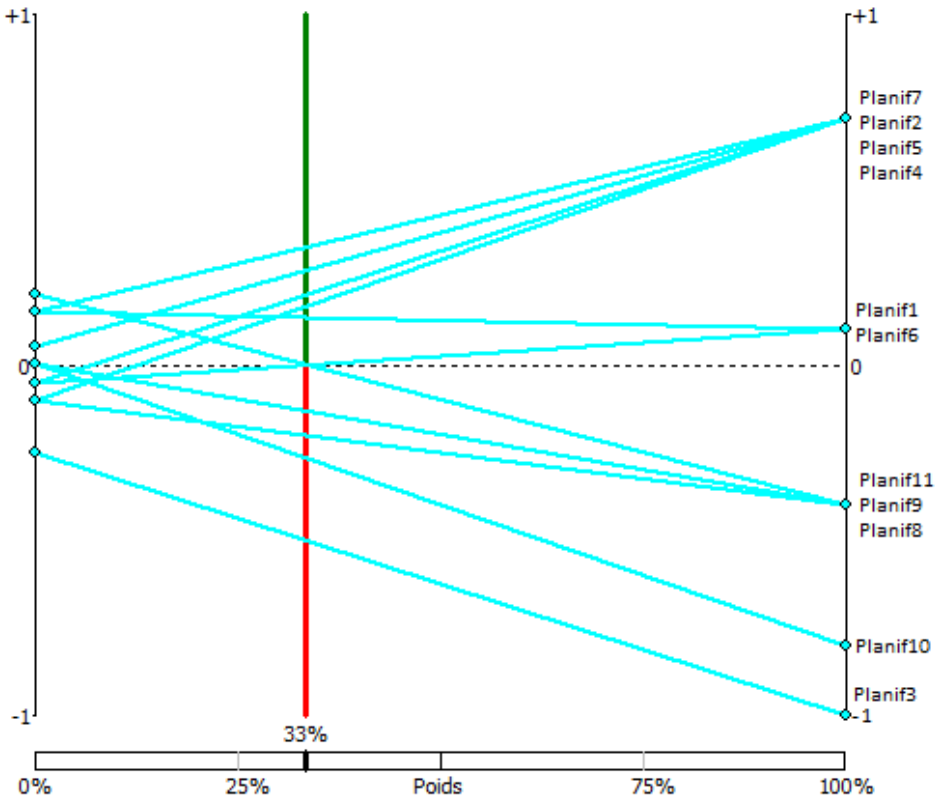

Figure 2.32 : Intervalles de stabilité pour le temps perdu avec des poids égaux

La figure 2.32 présente les intervalles de stabilité pour le critère du temps perdu. En observant cette figure, il est possible de constater que la septième planification se trouve au second rang à égalité avec la première planification lorsque le temps perdu a un poids de 0% à environ 5%. Lorsque le poids du temps perdu est supérieur à 5%, la planification 7 conserve son premier rang jusqu'à un poids de 100%.

2.4.5.4 Choisir la solution finale

Tel que mentionné à la section 2.4.5.2, le choix de la solution finale s'effectue selon des poids égaux pour les trois critères. Dans ce cas, comme les figures 2.28 et 2.29 le démontrent, la meilleure planification est la septième. Cette dernière propose une durée de dix jours, un coût de 17840$ et un temps perdu nul. Il s'agit de l'essai 5.4, réalisé avec des paramètres d'une valeur unitaire pour le second, de 30 pour le septième et de 5 pour le huitième. Il est à noter que la planification 2.1 propose une même durée et un même coût pour le projet. Cependant, le temps perdu de celle-ci est de cinq, contrairement à un temps perdu nul pour la planification 5.4.

La planification 5.4 est la meilleure solution selon les critères retenus pour cette recherche. Cependant, tel que mentionné à quelques reprises, il s’agit d’une méthode générale pouvant être utilisée pour tout problème à critères et à compétences multiples. Le choix des critères ainsi que la pondération associée à chacun d'entre eux sont laissés au soin de l'utilisateur et méritent une attention particulière. En effet, le classement des planifications peut rapidement changer selon la variation du poids des critères. Il est donc essentiel pour l'utilisateur de définir ses besoins afin d'obtenir la solution optimale à son projet.

CONCLUSION GÉNÉRALE

Le RCPSP est un problème étudié par de nombreux chercheurs et une attention particulière a été accordée ces dernières années à diverses extensions telles que le RCPSP à modes multiples, à projets multiples, à compétences multiples et à objectifs multiples. Cette étude a été réalisée afin de proposer une méthode de résolution à critères multiples pour le RCPSP prenant en considération les compétences maîtrisées par les ressources ainsi que les compétences requises par chaque activité.

Un modèle a d'abord été créé à l'aide du logiciel d'optimisation multicritère Midaco version 5.0. Ce modèle créé en code Matlab a permis, pour un problème de départ et selon divers paramètres, de déterminer des solutions optimales, dans lesquelles le temps et le coût du projet étaient optimisés. À cette étape, aucune préférence n'est accordée pour l'un ou l'autre des critères. Par la suite, les solutions de Pareto obtenues ont été analysées à l'aide du logiciel d'aide à la décision Prométhée. C'est à cette étape que la préférence de l'utilisateur est insérée dans le problème. La durée du projet, son coût ainsi que le temps perdu sont les critères utilisés dans le choix de la solution finale. Il est d'ailleurs à noter qu'il ne semble y avoir aucune corrélation entre la durée du projet et le temps d'inactivité des ressources.

Lorsque les critères multiples de la première étape se retrouvent dans la seconde, de nouveaux critères peuvent être ajoutés à celle-ci, tel qu'effectué dans cette étude. Le lecteur pourrait donc se demander la raison pour laquelle les critères utilisés à la seconde étape n'ont pas directement été utilisés à la première. Cela s'explique par le fait que certains critères ne peuvent être évalués que lorsqu'un ordonnancement est défini, comme pour cette étude, où le temps inoccupé par les ressources peut être déterminé seulement lorsque la première étape est terminée.

Les solutions obtenues à partir des essais réalisés avec la version 5.0 de Midaco proposent toutes des résultats réalisables. Chaque solution obtenue respecte les équations utilisées pour représenter le problème, prouvant ainsi le bon fonctionnement de la méthode. Il est à noter que la version 6.0 de ce logiciel est disponible et que des changements ont été apportés au traitement du multiobjectif.

La méthode présentée dans ce mémoire peut être utilisée en pratique pour tous les types de projets, à partir du moment où le gestionnaire est en mesure de planifier le projet, d'évaluer la durée des tâches, la quantité de ressources nécessaires et le coût d'utilisation de chacune d'entre elles avec certitude. Il est nécessaire d'avoir toutes les contraintes nécessaires, d'être en mesure d'effectuer un bilan des compétences des ressources et d'avoir accès à une base de données sur les anciens projets afin d'avoir les informations nécessaires, de manière à ce que l'utilisation de cette méthode apporte un gain important à l'entreprise.

Dans le cas où cette recherche serait poursuivie, plusieurs éléments pourraient être améliorés ou ajoutés. D'autres variations des différents paramètres utilisés par Midaco peuvent être réalisées, de manière à obtenir des résultats différents de ceux présentés dans cette étude, dans le but d'améliorer les solutions obtenues.

Un élément intéressant à modifier se situe au niveau de la définition des équations du problème utilisées avec Midaco. Le projet étudié a été repris de Montoya (2012) et présente quatre activités, quatre ressources ainsi que trois compétences. Les équations utilisées afin de représenter le problème, soient celles présentées à la section 2.2, ont chacune été traitées individuellement et à la main pour la quantité de variables définies. Même s'il s'agit d'un problème de petite taille, le temps requis pour cette opération était considérable. Il serait donc intéressant de développer un second modèle à l'aide d'un logiciel de calcul quelconque ou d'ajouter une section au modèle actuel permettant de créer automatiquement les équations et les inéquations à résoudre avec Midaco version 5.0. Le temps utilisé afin de développer les équations pour ce problème est tout de même raisonnable. Cependant, un projet réel ne comporte rarement que quatre activités ou quatre ressources. Donc, dans le cas d'un projet réel où le nombre d'activités, de

ressources et de compétences est largement supérieur à celui présenté dans cette étude, le temps requis afin de développer les équations serait probablement trop élevé.

Cette recherche présente un modèle prenant en considération deux extensions du RCPSP, le problème à compétences et à objectifs multiples. Il pourrait être intéressant, en plus de ces deux dernières, d'ajouter l'extension du problème de planification à projets multiples. En effet, les entreprises œuvrant dans le domaine de la gestion de projet travaillent rarement sur un seul projet à la fois. Les ressources doivent partager leur temps entre divers projets selon les priorités définies. Le fait d'ajouter cette extension au problème à critères et compétences multiples présenté dans cette étude pourrait proposer une méthode s'approchant davantage de la réalité des organisations orientées projets.

La méthode présentée dans cette étude est tout à fait pertinente afin de proposer des solutions fonctionnelles respectant toutes les contraintes émises dans le problème. En y apportant quelques améliorations telles que celles présentées ci-haut, cette méthode pourrait certainement être utilisée par des entreprises afin d'optimiser les critères correspondant à leurs besoins.

RÉFÉRENCES BIBLIOGRAPHIQUES

Al-Rashdan, D., B. Al-Kloub, A. Dean et T. Al-Shemmeri (1999). "Environmental impact assessment and ranking the environmental projects in Jordan." European Journal of Operational Research **118**(1): 30-45.

Alvarez-Valdes, R. et J. Tamarit (1989a). «Heuristic algorithms for resource-constrained project scheduling : A review and an empirical analysis.» Dans *R. Slowinski et J. Weglarz, Advances in Project Scheduling*, 113-134. Amsterdam.

Alvarez-Valdes, R. et J. Tamarit (1989b). «Algoritmos heuristicos deterministas y aleatorios en sucuenciacion de proyectos con recusdod limitados.» Qüestiio **13**: 173-191.

Alvarez-Valdés, R. et J. Tamarit (1993). «The project scheduling polyhedron: Dimension, facets and lifting theorems.» European Journal of Operational Research **67**(2): 204–220.

Artigues, C., S. Demassey et E. Néron (2008). *Resource-constrained project scheduling, models, algorithms, extensions and applications*, 1 édition. «Control Systems, Robotics and Manufacturing Series.» Wiley-ISTE, 288 p.

Artigues, C., P. Michelon et S. Reusser (2003). «Insertion techniques for static and dynamic resource-constrained project scheduling.» European Journal of Operational Research **149**(2): 249-267.

Artigues, C. et F. Roubellat (2000). «A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes.» European Journal of Operational Research **127**(2): 297-316.

Baker, K.R. (1974). «Introduction to sequencing and scheduling.» John Wiley, New York.

Balas, E. (1971), "Project scheduling with resource constraints." Dans E.M.L. Beale, Applications of Mathematical Programming techniques, 187-200. London.

Baptiste, P. et S. Demassey (2004). «Tight LP bounds for resource constrained project scheduling.» OR Spectrum **26**(2): 251-262.

Baptiste, P., C. Le Pape et W. Nuijten (1999). «Satisfiability tests and time-bound adjustments for cumulative scheduling problems.» Annals of Operations Research **92**(1-4): 305-333.

Bedworth, D.D. et J.E. Bailey (1982). *Integrated production control systems - Management, analysis, design.* Deuxième Édition. New York: Wiley, 387-420.

Bell, C.E., et K. Park (1990). «Solving resource-constrained project scheduling problems by a* search.» Naval Research Logistics **37**(1): 61-84.

Bellenguez O, Néron E (2005). «Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills.» Dans *Practice and theory of automated timetabling V: 5th international conference*, 229-243. Pittsburgh.

Bellenguez-Morineau, O. (2008). «Methods to solve multi-skill project scheduling problem.» *4OR: A Quarterly Journal of Operations Research* **6**(1): 85-88.

Bellenguez-Morineau, O. et E. Néron (2007). «A branch-and-bound method for solving multi-skill project scheduling problem.» RAIRO Operations Research **41**: 155-170.

Boctor, F.F. (1990). «Some efficient multi-heuristic procedures for resource-constrained project scheduling.» Europe Journal of Operational Research **49**(1): 3-13.

Boctor, F. F. (1996). «Resource-constrained project scheduling by simulated annealing.» International Journal of Production Research **34**(8): 2335.

Bouleimen, K. et H. Lecocq (2003). «A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version.» European Journal of Operational Research **149**(2): 268-281.

Brans, J. P. et P. H. Vincke (1985). "A Preference Ranking Organization Method: The PROMETHEE Method for Multiple Criteria Decision Making" Management Science **31**(6) : 647-656.

Brans, J. P., B. Mareschal et Ph. Vincke (1986). "How to Select and how to Rank Projects : The PROMETHEE Method." E.J.O.R. **24** : 228-238.

Briand, C. et J. Bezanger (2006). «An any-order SGS for project scheduling with scarce resources and precedence constraints.» Dans *10th International Workshop on Project Management and Scheduling*, 95-100, Poznan, Pologne.

Browning, T.R. et A.A. Yassine (2010). «Resource-constrained multi-project scheduling: Priority rule performance revisited.» International Journal of Production Economics **126**(2): 212-228.

Brucker, P., A. Drexl, R. Möhring, K. Neumann and E. Pesch (1999). «Resource-constrained project scheduling: Notation, classification, models, and methods.» European Journal of Operational Research **112**(1): 3-41.

Brucker, P. and S. Knust (2000). «A linear programming and constraint propagation-based lower bound for the RCPSP.» European Journal of Operational Research **127**(2): 355-362.

Brucker, P., S. Knust, A. Schoo et O. Thiele (1998). «A branch and bound algorithm for the resource-constrained project scheduling problem.» European Journal of Operational Research **107**(2): 272-288.

Cai, X. et K. N. Li (2000). «A genetic algorithm for scheduling staff of mixed skills under multi-criteria.» European Journal of Operational Research **125**(2): 359-369.

Carlier, J. (1987). «Scheduling jobs with release dates and tails on identical machines to minimize the makespan.» European Journal of Operational Research **29**(3): 298-306.

Carlier, J. et B. Latapie (1991). «Une méthode arborescente pour résoudre les problèmes cumulatifs.» RAIRO Operations Research **25**(3): 311-340.

Carlier, J. et E. Néron (2003). «On linear lower bounds for the resource constrained project scheduling problem.» European Journal of Operational Research **149**(2): 314-324.

Certa, A., M. Enea, G. Galante et C. Manuela La Fata (2009). «Multi-objective human resources allocation in R&D projects planning.» International Journal of Production Research **47**(13): 3503-3523.

Cho, J.-H. et Y.-D. Kim (1997). «A simulated annealing algorithm for resource constrained project scheduling problems.» The Journal of the Operational Research Society **48**(7): 736-744.

Christofides, N., R. Alvarez-Valdes, J.M. Tamarit (1987). «Project scheduling with resource constraints: a branch-and-bound approach.» European Journal of Operational Research **29**(2): 262–273.

Colorni, A., M. Dorigo et V. Maniezzo (1992). «Distributed optimization by ant colonies." Dans *Varela, F. et P. Bourgine, editors, Proceedings of ECAL'91 - First European Conference on Artificial Life*, 134-142. Paris.

Cooper, D. (1976). «Heuristic for scheduling resource-constrained projects: An experimental investigation.» Management science **22**(11): 1186-1194.

Correia, I., L. Lourenço et F. Saldanha-da-Gama (2012). «Project scheduling with flexible resources: formulation and inequalities.» OR Spectrum **34**(3): 635-663.

Correia, I. et F. Saldanha-da-Gama (2014). «The impact of fixed and variable costs in a multi-skill project scheduling problem: An empirical study.» Computers & Industrial Engineering **72**: 230-238.

D'Avignon, G. et B. Mareschal (1989). "Specialization of Hospital Services in Quebec : an Application of the PROMETHEE and GAIA Methods." Mathematical and Computer Modelling **12**(10/11) : 1393-1400.

Dalfard, V. M. et V. Ranjbar (2012). «Multi-project scheduling with resource constraints & priority rules by the use of simulated annealing algorithm.» Technical Gazette. **19**(3): 493-499.

Damak, N., B. Jarboui, P. Siarry et T. Loukil (2009). «Differential evolution for solving multi-mode resource-constrained project scheduling problems.» Computers & Operations Research **36**(9): 2653-2659.

Davis, E.W., et G.E. Heidorn (1971). «An algorithm for optimal project scheduling under multiple resource constraints.» Management Science **17**(12): 803-816.

Davis, E.W. et J.H. Patterson (1975). «A comparison of heuristic and optimum solutions in resource-constrained project scheduling.» Management science **21**(8): 944-955.

Debels, D., B. De Reyck, R. Leus et M. Vanhoucke (2006). «A hybrid scatter search/electromagnetism meta-heuristic for project scheduling.» European Journal of Firat, M. et C.A. J. Hurkens (2012). «An improved MIP-based approach for a multi-skill workforce scheduling problem.» Journal of Scheduling **15**(3): 363-380. Operational Research **169**(2): 638-653.

Demassey, S., C. Artigues, Ph. Baptiste et P. Michelon (2004). «Lagrangean relaxation-based lower bounds for the RCPSP.» dans *Proceedings of PMS*, 76-79.

Demassey, S., C. Artigues et P. Michelon (2002). «A hybrid constraint propagation-based cutting plane algorithm for the RCPSP.» Dans *Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems*, France.

Demeulemeester, E. and W. Herroelen (1992). «A branch-and-bound procedure for the multiple resource-constrained project scheduling problem.» Management Science **38**(12): 1803-1818.

Demeulemeester, E. L. et W. S. Herroelen (1997). «A branch-and-bound procedure for the generalized resource-constrained project scheduling problem.» Operations Research **45**(2): 201.

Dorigo, M. et L. M. Gambardella (1997a). «Ant colonies for the traveling salesman problem.» BioSystems **43**(2): 73–81.

Dorigo, M. et L. M. Gambardella (1997b). "Ant colony system: A cooperative learning approach to the traveling salesman problem." IEEE Transactions on Systems, Man and Cybernetics **1**(1): 53–66.

Dorigo, M., V. Maniezzo et A. Colorni (1996). «The ant system: optimization by a colony of cooperating agents.» IEEE Transactions on Systems, Man and Cybernetics, part B **26**(1):29–41.

Dréo, J., A. Pétrowski, P. Siarry et É. Taillard (2003). «Métaheuristiques pour l'optimisation difficile.» Eyrolles, 356 p. Paris.

Drexl, A. (1997). «Local search methods for project scheduling under partially renewable resource constraints.» Dans *INFORMS San Diego Meeting*, 4-7.

Elsayed, E.A. (1982). «Algorithms for project scheduling with resource constraints.» International Journal of Production Research **20**(1): 95-103.

Fendley, L.G. (1968). «Towards the development of a complete multi-project scheduling System.» Journal of Industrial Engineering **19**: 505–515.

Fernández-Castro, A. S. et M. Jiménez (2005). "PROMETHEE: an extension through fuzzy mathematical programming." Journal of the Operational Research Society **56**(1) : 119-122.

Fisher, M. (1973). «Optimal solution of scheduling problems using Lagrange multipliers. Part I.» Operational Research **21**(5): 1114-1127.

Fleszar, K. et K. S. Hindi (2004). «Solving the resource-constrained project scheduling problem by a variable neighbourhood search.» European Journal of Operational Research **155**(2): 402-413.

Fogel, L.J., A.J. Owens and M.J. Walsh (1966). *Artificial intelligence through simulated evolution.* l'Université du Michigan: John Wiley & Sons, 170 p.

Fraser, A.S. (1957). «Simulation of genetic systems by automatic digital computers.» Australian Journal of Biological Sciences **10**(4): 484-491.

Gambardella, L., E. Taillard et G. Agazzi (1999). «MACS-VPTW: A multiple ant colony system for vehicle routing problems with time windows.» Dans *D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization*, 63-76. London.

Garey, M.R. et D.S. Johnson (1979). «Computers and intractability: a guide to the theory of NP-completeness.» Computers and Intractability 340 p.

Geldermann, J., T. Spengler et O. Rentz (2000). "Fuzzy outranking for environmental assessment. Case study: iron and steel making industry." Fuzzy Sets and Systems **115**(1): 45-65.

Glover, F. (1989a). «Tabu search - Part I.» ORSA Journal on Computing **1(3)**: 190-206.

Glover, F. (1989b). «Tabu search - Part II.» ORSA Journal on Computing **2**(1): 4-32.

Godard, D., P. Laborie and W. Nuijten (2005). «Randomized large neighborhood search for cumulative scheduling.» dans *Proceedings of the International Conference on Automated Planning and Scheduling*, 81-89. Monterrey.

Goldberg, D.E. (1989). «Genetic algorithms in search, optimization and machine learning.» Machine learning **3**(2): 95-99.

Gomar, J.E., R.K. Terrien, C.T. Haas, D.P. Morton et R.L. Tucker (2000). *Assignment and allocation optimisation of a partially multiskilled workforce.* Internal report of Center for Construction Industry Studies, University of Texas.

Gonçalves, J. F., J. J. M. Mendes et M. G. C. Resende (2007). «A genetic algorithm for the resource constrained multi-project scheduling problem.» European Journal of Operational Research **189**(3): 1171-1190.

Gonguet, L. (1969). «Comparison of three heuristic procedures for allocating resources and producing schedules.» Dans H.J.M Lombaers, Project Planning by Netwok Analysis, 249-255. Amsterdam.

Goumas, M et V. Lygerou, V (2000). "An extension of the ORESTE method for decision making in fuzzy environment: ranking of alternative energy exploitation." European Journal of Operational Research **1**(23) : 606-613.

Gruat La Forme, F.-A. a. B.-G., Valerie et Campagne, Jean Pierre (2006). «Modélisation d'un problème d'ordonnancement avec prise en compte des compétences.» dans *6ème conférence francophone de Modélisation et Simulation*, Rabat.

Gul, M., E. Celik, A. Tsakin Gumus et A. Fuat Guneri (2017). «A fuzzy logic based Promethee method for material selection problems.» Journal of Basic and Applied Sciences **7**(1): 68-79.

Hao, J.-K., P. Galinier et M. Habib (1999). «Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes.» Revue d'intelligence artificielle **13**(2):283-324.

Hartmann, S. (2002). «A self-adapting genetic algorithm for project scheduling under resource constraints.» Naval Research Logistics **49**(5): 433-448.

Hartmann, S. et D. Briskorn (2010). «A survey of variants and extensions of the resource-constrained project scheduling problem.» European Journal of Operational Research **207**(1): 1–14.

Hartmann, S. et A. Drexl (1997). "Project scheduling with multiple modes: A comparison of exact algorithms." Networks **32**(4) : 283-297.

Hartmann, S. et R. Kolisch (2000). «Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem.» European Journal of Operational Research **127**(2): 394-407.

Hastings, N.A.J. (1972). «On resource allocation in project networks.» Operational Research Quarterly **23**(2): 217-221.

Herroelen, W., B. De Reyck et E. Demeulemeester (1998). «Resource-constrained project scheduling: A survey of recent developments.» Computers & Operations Research **25**(4): 279-302.

Herroelen, W., et R. Leus (2004). «The construction of stable project baseline schedules.» European Journal of Operational Research **156**(3), 550-565.

Holland, J.H. (1962). «Outline for logical theory of adaptive systems.» Journal of the Association for Computing Machinery **9**(3): 297–314.

Icmeli, O., S. Erenguc et C.J. Zappe (1993). «Project scheduling Problems: A Survey.» International Journal of Operations & Production Management **13**(11) : 80-91.

Jozefowska, J., M. Mika, R. Rozycki, G. Waligora et J. Weglarz (2001). «Simulated annealing for multi-mode resource-constrained project scheduling.» Annals of Operations Research **102**(1–4): 137–155.

Kelley, J.E. (1963). «The critical-path method : Resources planning and scheduling.» Dans *Thompson, Industrial Scheduling*, 347-365. Englewood Cliffs.

Kesen, S., S. Das and Z. Güngör (2010). "A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (vmcs)." Computers & Operations Research **37**(6) : 1148-1156.

Kim, K.W., Y.S. Yun, J.M. Yoon, M. Gend et G. Yamazaki (2005). «Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling.» Computers in industry **56**(2): 143-160.

Kirkpatrick, S., C.D. Gelatt jr. et M.P. Vecchi (1983). «Optimization by simulated annealing.» Science 220(4592): 671-680.

Kochetov, Y. et A. Stolyar (2003). «Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem.» dans *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies.*

Kohlmorgen, J., K.R. Müller et K. Pawelzik (1998). «Analysis of drifting dynamics with neural network hidden markov models.» Dans *Advances in Neural Information Processing Systems*, 735-741.

Kolisch, R. (1995). «Project scheduling under resource constraints - Efficient heuristics for several problem classes.» Physica, Heidelberg.

Kolisch, R. (1996a). «Efficient priority rules for the resource-constrained project scheduling problem.» Journal of Operations Management **14**(3): 179-192.

Kolisch, R. (1996b). «Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation.» European Journal of Operational Research **90**(2): 320-333.

Kolisch, R. et R. Padman (2001). «An integrated survey of deterministic project scheduling.» Omega **29**(3): 249–272.

Kolisch, R. et S. Hartmann (2006). «Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis.» Dans *Project Scheduling: Recent Models, Algorithms and Applications*, 147-178. Amsterdam.

Koshijima, I. et T. Umeda (2001). «Human resource allocation in project management – management science approach.» Internal report of Chiba Institute of Technology Tsudanuma, Narashino, Chiba.

Krzeszowska, B. (2009). «Evolutionary algorithm with direct chromosome representation in multi-criteria project scheduling.» Multiple Criteria Decision Making/University of Economics in Katowice 5: 181-195.

Kumanan, S., G. J. Jose et K. Raja (2006). «Multi-project scheduling using an heuristic and a genetic algorithm.» International Journal of Advanced Manufacturing Technology **31**(3/4): 360-366.

Kurtulus, I. et E.W. Davis (1982). «Multiproject scheduling: categorisation of heuristic rules performance.» Management Science **28**(2): 161-172.

Laborie, P. (2005). «Complete MCS-based search: Application to resource constrained project scheduling.» Dans *Proceedings of the nineteenth international joint conference on artificial intelligence*, 181-186.

Le Teno, J. F. et B. Mareschal (1998). "An interval version of PROMETHEE for the comparison of building products' design with ill-defined data on environmental quality." European Journal of Operational Research **109**(2) : 522-529.

Lee, J.-K. et Y.-D. Kim (1996). «Search heuristics for resource-constrained project scheduling.» Journal of the Operational Research Society **47**(5): 678-689.

Lenstra, J.K. et A. Kan (1978). «Complexity of scheduling under precedence constraints.» Operations Research **26**(1): 22–35.

Leon, V. J. et B. Ramamoorthy (1995). «Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling.» Operations Research-Spektrum **17**(2): 173-182.

Leu, S.S. et C.H. Yang (1999). «Ga-based multicriteria optimal model for construction scheduling.» Joumal of Construction Engineering and Management **125**(6): 420-427.

Li, R.-Y. et J. Willis (1992). «An iterative scheduling technique for resource-constrained project scheduling.» European Journal of Operational Research **56**(3): 370-379.

Lova, A., C. Maroto et P. Tormos (2000). «A multi criteria heuristic method to improve resource allocation in multi project scheduling.» European Journal of Operational Research **127**(2): 408-424.

Mareschal, B. (2018). The PROMETHEE Bibliographical Database. Repéré à http://www.promethee-gaia.net/bibliographical-database.html

Maystre, L. Y., J. Pictet et J. Simos (1994). "Méthodes multicritères ELECTRE : description, conseils pratiques et cas d'application à la gestion environnementale." Lausanne, Presses polytechniques et universitaires romandes.

Mendes, J. J. M., J. F. Gonçalves et M. G. C. Resende (2009). «A random key based genetic algorithm for the resource constrained project scheduling problem.» Computers & Operations Research **36**(1): 92-109.

Merkle, D. et M. Middendorf (2002). «Modeling the Dynamics of Ant Colony Optimization.» Evolutionary Computation **10**(3): 235-262.

Metropolis, N., A. Rosembluth, M. Rosembluth et A. Teller (1953). «Equation of state calculations by fast computing machines.» Journal of Chemical Physics **21**(6): 1087-1092

Mingozzi, A., V. Maniezzo, S. Ricciardelli et L. Bianco (1998). «An exact algorithm for the resource-constrained project scheduling problem based on a new.» Management Science **44**(5): 714-729.

Mladineo, N., J" Margeta, J. P. Brans et B. Mareschal (1987). "Multicriteria ranking of alternative locations for small scale hydro plants." European Journal of Operational Research **31**(2) : 215-222.

Möhring, R. H., A. S. Schulz, F. Stork et M. Uetz (2003). «Solving Project Scheduling Problems by Minimum Cut Computations.» Management Science **49**(3): 330-350.

Montoya, Casas Carlos Eduardo (2012). «New methods for the multi-skills project scheduling problem.» Thèse de doctorat. Ecole des Mines de Nantes.

Naoum, Salah Eddine Anouar (2010). "Constitution d'un portefeuille de projets à partir d'un indice de centralité des projets." Mémoire.Rimouski, Québec, Université du Québec à Rimouski, Unité départementale des sciences de la gestion du campus de Rimouski, 121 p.

Neumann, K., C. Schwindt et J. Zimmermann (2003). *Project scheduling with time windows and scarce resources: Temporal and resource-constrained project scheduling with regular and nonregular objective functions*, Deuxième Édition. Springer-Verlag Berlin Heidelberg, 385 p.

Özdamar, L. et G. Ulusoy (1995). «A survey on the resource-constrained project scheduling problem.» AIIE Transactions **27**(5): 574-586.

Palpant, M., C. Artigues et P. Michelon (2004). «LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search.» Annals of Operations Research **131**(1-4): 237-257.

Pascoe, T. L. (1966). «Allocation of resources C.M.P.» Revue Française Recherche Opérationnelle **38**: 31-38.

Patterson, J. H. (1984). «A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem.» Management Science **30**(7): 854-867.

Patterson, J. H., R. Slowinski, F. B. Talbot et J. Weglarz (1989). "An algorithm for a general class of precedence and resource-constrained scheduling problem." Dans Advances in Project Scheduling, 3-27. Amsterdam.

Peteghem, V.V. et M. Vanhoucke (2010). «A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem.» European Journal of Operational Research **201**(2): 409-418.

Pinson, E., C. Prins et F. Rullier (1994). «Using tabu search for solving the resource-constrained project scheduling problem» Dans *Proceedings of the 4th international workshop on project management and scheduling*, 102-106. Belgium,

Pritsker, A.A.B., L.J. Watters et P. M. Wolfe (1969). «Multiproject scheduling with limited resources: a zero-one programming approach» Management Science **16**(1): 93-108.

Radojevic, D. et S. Petrovic (1997). "A fuzzy approach to preference structure in multicriteria ranking." International Transactions in Operational Research **4**(5): 419-430.

Rechenberg, I. (1965). «Cybernetic solution path of an experimental problem." Dans *Royal Aircraft Establishment Library Translation*, 301-310.

Roy, B. (1985). "Méthodologie Multicritère d'Aide à la Décision." Economical Paris. 406 p.

Schrimer, A. et S. Riesenberg (1997). "Parametrized heuristics for project scheduling - Biased random sampling methods." Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 456.

Schwindt, C. (2005). *Resource allocation in project management*, Première Édition. Springer-Verlag Berlin Heidelberg, 194 p.

Shaffer, L.R., J.B. Ritter et W.L. Meyer (1965). *The critical-path method*, McGraw-Hill: New-York, 212 p.

Singh, A. (2013). «Resource Constrained Multi-Project Scheduling with Priority Rules & Analytic Hierarchy Process.» Annals of DAAAM & Proceedings **24**(1): 725-734.

Sprecher, A. (2000). «Scheduling resource-constrained projects competitively at modest memory requirements.» Management Science **46**(5): 710.

Sprecher, A., S. Hartmann et A. Drexl (1994). «Project scheduling with discrete time-resource and resource-resource tradeoffs.» Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, wp357, Alemanha, 24 p.

Sprecher, A., R. Kolisch et A. Drexl (1995). «Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem.» European Journal of Operational Research **80**(1): 94-102.

Srinivasa Raju, K. (1995). "Studies on Multi Criterion Decision Making Methods and Management of Irrigation Systems." Ph.D Thesis Kharagupur.: Indian Institute of Technology.

Stinson, J.P., E.W. Davis et B.M. Khumawala (1978). «Multiple resource-constrained scheduling using branch-and-bound.» AIIE Transactions 10(3): 252-259.

Struys, W. et H. Pastijn (1988). "The Recourse to Several Criteria in Determining a Fair Burden Sharing Within an Alliance : The Case of NATO." Operational Research **87** : 566-581.

Stüzle, T. et H. Hoos (1997). «Improvements on the ant System: Introducing MAX-MIN ant system." Dans *Artificial Neural Networks and Genetic Algorithms*, 245-249. Vienna.

Stüzle, T. et H. Hoos, (2000). «MAX-MIN Ant System.» Future Generation Computer System **16**(8): 889–914.

Talbot, B. et J.H. Patterson (1978). «An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems.» Management Science **24**(11): 1163-1174.

Thesen, A. (1976). «Heuristic scheduling of activities under resource and precedence restrictions.» Management Science **23**(4): 412-422.

Thomas, P.R. et S. Salhi (1998). «A Tabu Search Approach for the Resource Constrained Project Scheduling Problem.» Journal of Heuristics **4**(2): 123-139.

Toth, P. et D. Vigo (2002). "The vehicle routing problem." Society for Industrial Mathematics **9**.

Uetz, M. (2001). *Algorithms for deterministic and stochastic scheduling*. Thèse de doctorat, Université de Berlin.

Ulusoy, G. et L. Özdamar (1989). «Heuristic performance and network/resource characteristics in resource-constrained project scheduling.» Journal of the Operational Research Society **40**(12): 1145-1152.

Ulusoy, G. et L. Özdamar (1996). «An iterative local constraint based analysis for solving the resource constrained project scheduling problem.» Journal of the Operational Research Society **14**(3): 193-208.

Valls, V., S. Quintanilla et F. Ballestin (2003). «Resource-constrained project scheduling: A critical activity reordering heuristic.» European Journal of Operational Research **149**(2): 282–301.

Valls, V., F. Ballestín et S. Quintanilla (2004). «A population-based approach to the resource-constrained project scheduling problem.» Annals of Operations Research **131**(1-4): 305-324.

Valls, V., F. Ballestín et S. Quintanilla (2008). «A hybrid genetic algorithm for the resource-constrained project scheduling problem.» European Journal of Operational Research **185**(2): 495-508.

Valls, V., Á. Pérez et S. Quintanilla (2009). «Skilled workforce scheduling in Service Centres.» European Journal of Operational Research **193**(3): 791-804.

Vartouni, A.M. et L.M. Khanli (2014). «A hybrid genetic algorithm and fuzzy set applied to multi-mode resource-constrained project scheduling problem.» Journal of Intelligent & Fuzzy Systems **26**(3): 1103-1112.

Vercellis, C. (1994). «Constrained multi-project planning problems: a Lagrangean decomposition approach.» European Journal of Operational Research **78**(2): 267-275.

Vianna, A. et J.P. de Sousa (2000). «Using metaheuristic in multiobjective resource constrained project scheduling.» European Journal of Operational Research **120**(2): 359-374.

Vincke, Ph. (1989). "L'aide multicritère à la décision." Bruxelles, Éditions de l'Université de Bruxelles, Paris, Éditions Ellipses, 1989, 179 p.

Whitehouse, G.E. et J.R. Brown (1979). «Genres : an extension of brooks algorithm for project scheduling with resource constraints.» Computer & Industrial Engineering **3**(3): 261-268.

Zhu, G., J.F. Bard et G. Yu (2006). "A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem." INFORMS Journal on Computing **18**(3): 377–390.

ANNEXE A : Code Matlab complet du problème

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Midaco
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%  MAIN PROGRAM   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
 problem.func = @problem_function; % Call is [f,g] =
problem_function(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Étape 1: Définition du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Étape 1.A: Dimensions du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
problem.o  = 3; % Nombre d'objectifs
problem.n  = 173; % Nombre de variables (au total)
problem.ni = 173; % Nombre de variables entières (0 <= nint <= n)
problem.m  = 168; % Nombre de contraintes (au total)
problem.me = 32; % Nombre de contraintes d'égalité (0 <= me <= m)

es1=0; % Début au plus tôt de l'activité 1
es2=0; % Début au plus tôt de l'activité 2
es3=2; % Début au plus tôt de l'activité 3
es4=5; % Début au plus tôt de l'activité 4

ls1=8; % Début au plus tard de l'activité 1
ls2=5; % Début au plus tard de l'activité 2
ls3=10; % Début au plus tard de l'activité 3
ls4=10; % Début au plus tard de l'activité 4

T=13; % Horizon temporel

% Étape 1.B: Limites inférieures et supérieures 'xl' & 'xu'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
problem.xl =
[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;es1;es2;es3;es4;8]
;
problem.xu =
[1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;
1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;
1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;ls1;ls2;ls3;ls4;T]
```

```
;

% Étape 1.C: Point de départ 'x'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
problem.x  = problem.xl; % 'x' = limites infÈrieures 'xl'

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Étape 2: Choix des critères d'arrêt et des options d'impression
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Étape 2.A: Critères d'arrêt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
option.maxeval  = 10000000;    % Nombre maximal d'évaluations de la
fonction (e.g. 1000000)
option.maxtime  = 60*60*24; % Temps limite maximal en secondes (e.g. 1
jour = 60*60*24)

% Étape 2.B: Options d'impression

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
option.printeval  = 1000;  % Fréquence d'impression de la meilleure
solution actuelle (e.g. 1000)
option.save2file  = 1;     % Sauvegarde de l'écran et de la solution en
fichier .txt [ 0=NON/ 1=OUI]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Étape 3: Choix des paramètres Midaco
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

option.param( 1) =  0;  % ACCURACY
option.param( 2) =  0;  % SEED
option.param( 3) =  0;  % FSTOP
option.param( 4) =  0;  % ALGOSTOP
option.param( 5) =  0;  % EVALSTOP
option.param( 6) =  0;  % FOCUS
option.param( 7) =  0;  % ANTS
option.param( 8) =  0;  % KERNEL
option.param( 9) =  0;  % ORACLE
option.param(10) =  0;  % PARETOMAX
option.param(11) =  0;  % EPSILON
option.param(12) =  0;  % CHARACTER

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Étape 4: Choix du facteur de parallélisation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

option.parallel = 0;  % Série: 0 or 1, Parallèle: 2,3,4,5,6,7,8...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Call Midaco solver
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ solution ] = midaco( problem, option, key);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Problème d'optimisation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ f, g ] = problem_function( x )

%  Paramètres connus du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0:13; % Paramètre variable temporel

p1=2; % Durée de l'activité 1
p2=5; % Durée de l'activité 2
p3=3; % Durée de l'activité 3
p4=3; % Durée de l'activité 4

% Nombre de ressource(s) requise(s) pour la compétence k pour
l'activité i
b01=0;
b11=1;
b21=1;
b02=1;
b12=0;
b22=1;
b03=2;
b13=1;
b23=0;
b04=0;
b14=1;
b24=0;

% Coût en $ par ressource m par compétence k, pour 1 unité de temps
C00=720;
C01=800;
C02=960;
C10=640;
C11=720;
C12=800;
C20=640;
C21=720;
C22=780;
C30=560;
C31=640;
C32=720;

%  Fonctions objectif du problème
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f(2) = x(173);
```

```
f(3) =
x(121)*p1*C00+x(122)*p1*C10+x(123)*p1*C20+x(124)*p1*C30+x(125)*p1*C01+x
(126)*p1*C11+x(127)*p1*C21+x(128)*p1*C31+x(129)*p1*C02+x(130)*p1*C12+x(
131)*p1*C22+x(132)*p1*C32+x(133)*p2*C00+x(134)*p2*C10+x(135)*p2*C20+x(1
36)*p2*C30+x(137)*p2*C01+x(138)*p2*C11+x(139)*p2*C21+x(140)*p2*C31+x(14
1)*p2*C02+x(142)*p2*C12+x(143)*p2*C22+x(144)*p2*C32+x(145)*p3*C00+x(146
)*p3*C10+x(147)*p3*C20+x(148)*p3*C30+x(149)*p3*C01+x(150)*p3*C11+x(151)
*p3*C21+x(152)*p3*C31+x(153)*p3*C02+x(154)*p3*C12+x(155)*p3*C22+x(156)*
p3*C32+x(157)*p4*C00+x(158)*p4*C10+x(159)*p4*C20+x(160)*p4*C30+x(161)*p
4*C01+x(162)*p4*C11+x(163)*p4*C21+x(164)*p4*C31+x(165)*p4*C02+x(166)*p4
*C12+x(167)*p4*C22+x(168)*p4*C32;
f(1)=f(2)+ f(3);

%  Contraintes d'égalité G(X)=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Équation 2.1
g(1)=x(1)*t(0+1)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t
(1+1)+x(7)*t(1+1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(1
2)*t(2+1)+x(13)*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4
+1)+x(18)*t(4+1)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(
23)*t(5+1)+x(24)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(
6+1)+x(29)*t(7+1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x
(34)*t(8+1)+x(35)*t(8+1)+x(36)*t(8+1)-x(169)*(b01+b11+b21);

g(2)=x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(0+1)+x(41)*t(1+1)+x
(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x(46)*t(2+1)+x(47)*t
(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t(3+1)+x(52)*t(3+1)+
x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+x(57)*t(5+1)+x(58)*
t(5+1)+x(59)*t(5+1)+x(60)*t(5+1)-x(170)*(b02+b12+b22);

g(3)=x(61)*t(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x
(66)*t(3+1)+x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t
(4+1)+x(72)*t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+
x(77)*t(6+1)+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*
t(7+1)+x(83)*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)
+x(88)*t(8+1)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)
*t(10+1)+x(94)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1)-
x(171)*(b03+b13+b23);

g(4)=x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(100)*t(5+1)+x(101)*t(6+1)
+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(105)*t(7+1)+x(106)*t(7+1)+
x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(110)*t(8+1)+x(111)*t(8+1)+x
(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115)*t(9+1)+x(116)*t(9+1)+x(
117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(120)*t(10+1)-
x(172)*(b04+b14+b24);

%  Équation 2.10
g(5)=x(121)+x(122)+x(123)+x(124)-b01;
g(6)=x(125)+x(126)+x(127)+x(128)-b11;
g(7)=x(129)+x(130)+x(131)+x(132)-b21;
g(8)=x(133)+x(134)+x(135)+x(136)-b02;
g(9)=x(137)+x(138)+x(139)+x(140)-b12;
g(10)=x(141)+x(142)+x(143)+x(144)-b22;
g(11)=x(145)+x(146)+x(147)+x(148)-b03;
```

```
g(12)=x(149)+x(150)+x(151)+x(152)-b13;
g(13)=x(153)+x(154)+x(155)+x(156)-b23;
g(14)=x(157)+x(158)+x(159)+x(160)-b04;
g(15)=x(161)+x(162)+x(163)+x(164)-b14;
g(16)=x(165)+x(166)+x(167)+x(168)-b24;

%  Équation 2.11
g(17)=x(1)+x(5)+x(9)+x(13)+x(17)+x(21)+x(25)+x(29)+x(33)-x(121)-x(125)-
x(129);
g(18)=x(2)+x(6)+x(10)+x(14)+x(18)+x(22)+x(26)+x(30)+x(34)-x(122)-
x(126)-x(130);
g(19)=x(3)+x(7)+x(11)+x(15)+x(19)+x(23)+x(27)+x(31)+x(35)-x(123)-
x(127)-x(131);
g(20)=x(4)+x(8)+x(12)+x(16)+x(20)+x(24)+x(28)+x(32)+x(36)-x(124)-
x(128)-x(132);

g(21)=x(37)+x(41)+x(45)+x(49)+x(53)+x(57)-x(133)-x(137)-x(141);
g(22)=x(38)+x(42)+x(46)+x(50)+x(54)+x(58)-x(134)-x(138)-x(142);
g(23)=x(39)+x(43)+x(47)+x(51)+x(55)+x(59)-x(135)-x(139)-x(143);
g(24)=x(40)+x(44)+x(48)+x(52)+x(56)+x(60)-x(136)-x(140)-x(144);

g(25)=x(61)+x(65)+x(69)+x(73)+x(77)+x(81)+x(85)+x(89)+x(93)-x(145)-
x(149)-x(153);
g(26)=x(62)+x(66)+x(70)+x(74)+x(78)+x(82)+x(86)+x(90)+x(94)-x(146)-
x(150)-x(154);
g(27)=x(63)+x(67)+x(71)+x(75)+x(79)+x(83)+x(87)+x(91)+x(95)-x(147)-
x(151)-x(155);
g(28)=x(64)+x(68)+x(72)+x(76)+x(80)+x(84)+x(88)+x(92)+x(96)-x(148)-
x(152)-x(156);

g(29)=x(97)+x(101)+x(105)+x(109)+x(113)+x(117)-x(157)-x(161)-x(165);
g(30)=x(98)+x(102)+x(106)+x(110)+x(114)+x(118)-x(158)-x(162)-x(166);
g(31)=x(99)+x(103)+x(107)+x(111)+x(115)+x(119)-x(159)-x(163)-x(167);
g(32)=x(100)+x(104)+x(108)+x(112)+x(116)+x(120)-x(160)-x(164)-x(168);

%  Contraintes d'inégalité G(x)>=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Équation 2.4
g(33)=-x(169)+x(171)-p1;
g(34)=-x(170)+x(172)-p2;
g(35)=-x(171)+x(173)-p3;
g(36)=-x(172)+x(173)-p4;

%  Équation 2.6
g(37)=-(x(1)+x(5)+x(9)+x(13)+x(17)+x(21)+x(25)+x(29)+x(33))+1;
g(38)=-(x(2)+x(6)+x(10)+x(14)+x(18)+x(22)+x(26)+x(30)+x(34))+1;
g(39)=-(x(3)+x(7)+x(11)+x(15)+x(19)+x(23)+x(27)+x(31)+x(35))+1;
g(40)=-(x(4)+x(8)+x(12)+x(16)+x(20)+x(24)+x(28)+x(32)+x(36))+1;

g(41)=-(x(37)+x(41)+x(45)+x(49)+x(53)+x(57))+1;
g(42)=-(x(38)+x(42)+x(46)+x(50)+x(54)+x(58))+1;
g(43)=-(x(39)+x(43)+x(47)+x(51)+x(55)+x(59))+1;
g(44)=-(x(40)+x(44)+x(48)+x(52)+x(56)+x(60))+1;
```

```
g(45)=-(x(61)+x(65)+x(69)+x(73)+x(77)+x(81)+x(85)+x(89)+x(93))+1;
g(46)=-(x(62)+x(66)+x(70)+x(74)+x(78)+x(82)+x(86)+x(90)+x(94))+1;
g(47)=-(x(63)+x(67)+x(71)+x(75)+x(79)+x(83)+x(87)+x(91)+x(95))+1;
g(48)=-(x(64)+x(68)+x(72)+x(76)+x(80)+x(84)+x(88)+x(92)+x(96))+1;

g(49)=-(x(97)+x(101)+x(105)+x(109)+x(113)+x(117))+1;
g(50)=-(x(98)+x(102)+x(106)+x(110)+x(114)+x(118))+1;
g(51)=-(x(99)+x(103)+x(107)+x(111)+x(115)+x(119))+1;
g(52)=-(x(100)+x(104)+x(108)+x(112)+x(116)+x(120))+1;

%  Équation 2.7
g(53)=-(x(1)+x(37))+1;
g(54)=-(x(1)+x(5)+x(37)+x(41))+1;
g(55)=-(x(5)+x(9)+x(37)+x(41)+x(45)+x(61))+1;
g(56)=-(x(9)+x(13)+x(37)+x(41)+x(45)+x(49)+x(61)+x(65))+1;
g(57)=-(x(13)+x(17)+x(37)+x(41)+x(45)+x(49)+x(53)+x(61)+x(65)+x(69))+1;
g(58)=-
(x(17)+x(21)+x(41)+x(45)+x(49)+x(53)+x(57)+x(65)+x(69)+x(73)+x(97))+1;
g(59)=-
(x(21)+x(25)+x(45)+x(49)+x(53)+x(57)+x(69)+x(73)+x(77)+x(97)+x(101))+1;
g(60)=-
(x(25)+x(29)+x(49)+x(53)+x(57)+x(73)+x(77)+x(81)+x(97)+x(101)+x(105))+1
;
g(61)=-
(x(29)+x(33)+x(53)+x(57)+x(77)+x(81)+x(85)+x(101)+x(105)+x(109))+1;
g(62)=-(x(33)+x(57)+x(81)+x(85)+x(89)+x(105)+x(109)+x(113))+1;
g(63)=-(x(85)+x(89)+x(93)+x(109)+x(113)+x(117))+1;
g(64)=-(x(89)+x(93)+x(113)+x(117))+1;
g(65)=-(x(93)+x(117))+1;

g(66)=-(x(2)+x(38))+1;
g(67)=-(x(2)+x(6)+x(38)+x(42))+1;
g(68)=-(x(6)+x(10)+x(38)+x(42)+x(46)+x(62))+1;
g(69)=-(x(10)+x(14)+x(38)+x(42)+x(46)+x(50)+x(62)+x(66))+1;
g(70)=-(x(14)+x(18)+x(38)+x(42)+x(46)+x(50)+x(54)+x(62)+x(66)+x(70))+1;
g(71)=-
(x(18)+x(22)+x(42)+x(46)+x(50)+x(54)+x(58)+x(66)+x(70)+x(74)+x(98))+1;
g(72)=-
(x(22)+x(26)+x(46)+x(50)+x(54)+x(58)+x(70)+x(74)+x(78)+x(98)+x(102))+1;
g(73)=-
(x(26)+x(30)+x(50)+x(54)+x(58)+x(74)+x(78)+x(82)+x(98)+x(102)+x(106))+1
;
g(74)=-
(x(30)+x(34)+x(54)+x(58)+x(78)+x(82)+x(86)+x(102)+x(106)+x(110))+1;
g(75)=-(x(34)+x(58)+x(82)+x(86)+x(90)+x(106)+x(110)+x(114))+1;
g(76)=-(x(86)+x(90)+x(94)+x(110)+x(114)+x(118))+1;
g(77)=-(x(90)+x(94)+x(114)+x(118))+1;
g(78)=-(x(94)+x(118))+1;

g(79)=-(x(3)+x(39))+1;
g(80)=-(x(3)+x(7)+x(39)+x(43))+1;
g(81)=-(x(7)+x(11)+x(39)+x(43)+x(47)+x(63))+1;
g(82)=-(x(11)+x(15)+x(39)+x(43)+x(47)+x(51)+x(63)+x(67))+1;
```

```
g(83)=-(x(15)+x(19)+x(39)+x(43)+x(47)+x(51)+x(55)+x(63)+x(67)+x(71))+1;
g(84)=-
(x(19)+x(23)+x(43)+x(47)+x(51)+x(55)+x(59)+x(67)+x(71)+x(75)+x(99))+1;
g(85)=-
(x(23)+x(27)+x(47)+x(51)+x(55)+x(59)+x(71)+x(75)+x(79)+x(99)+x(103))+1;
g(86)=-
(x(27)+x(31)+x(51)+x(55)+x(59)+x(75)+x(79)+x(83)+x(99)+x(103)+x(107))+1
;
g(87)=-
(x(31)+x(35)+x(55)+x(59)+x(79)+x(83)+x(87)+x(103)+x(107)+x(111))+1;
g(88)=-(x(35)+x(59)+x(83)+x(87)+x(91)+x(107)+x(111)+x(115))+1;
g(89)=-(x(87)+x(91)+x(95)+x(111)+x(115)+x(119))+1;
g(90)=-(x(91)+x(95)+x(115)+x(119))+1;
g(91)=-(x(95)+x(119))+1;

g(92)=-(x(4)+x(40))+1;
g(93)=-(x(4)+x(8)+x(40)+x(44))+1;
g(94)=-(x(8)+x(12)+x(40)+x(44)+x(48)+x(64))+1;
g(95)=-(x(12)+x(16)+x(40)+x(44)+x(48)+x(52)+x(64)+x(68))+1;
g(96)=-(x(16)+x(20)+x(40)+x(44)+x(48)+x(52)+x(56)+x(64)+x(68)+x(72))+1;
g(97)=-
(x(20)+x(24)+x(44)+x(48)+x(52)+x(56)+x(60)+x(68)+x(72)+x(76)+x(100))+1;
g(98)=-
(x(24)+x(28)+x(48)+x(52)+x(56)+x(60)+x(72)+x(76)+x(80)+x(100)+x(104))+1
;
g(99)=-
(x(28)+x(32)+x(52)+x(56)+x(60)+x(76)+x(80)+x(84)+x(100)+x(104)+x(108))+
1;
g(100)=-
(x(32)+x(36)+x(56)+x(60)+x(80)+x(84)+x(88)+x(104)+x(108)+x(112))+1;
g(101)=-(x(36)+x(60)+x(84)+x(88)+x(92)+x(108)+x(112)+x(116))+1;
g(102)=-(x(88)+x(92)+x(96)+x(112)+x(116)+x(120))+1;
g(103)=-(x(92)+x(96)+x(116)+x(120))+1;
g(104)=-(x(96)+x(120))+1;

%  Équation 2.8
g(105)=-
(b01+b11+b21)*(x(1)*t(0+1)+x(5)*t(1+1)+x(9)*t(2+1)+x(13)*t(3+1)+x(17)*t
(4+1)+x(21)*t(5+1)+x(25)*t(6+1)+x(29)*t(7+1)+x(33)*t(8+1))+(x(1)*t(0+1)
+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+1
)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)*
t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1)
+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24)
*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+1
)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(35
)*t(8+1)+x(36)*t(8+1));

g(106)=-
(b01+b11+b21)*(x(2)*t(0+1)+x(6)*t(1+1)+x(10)*t(2+1)+x(14)*t(3+1)+x(18)*
t(4+1)+x(22)*t(5+1)+x(26)*t(6+1)+x(30)*t(7+1)+x(34)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
```

```
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));

g(107)=-
(b01+b11+b21)*(x(3)*t(0+1)+x(7)*t(1+1)+x(11)*t(2+1)+x(15)*t(3+1)+x(19)*
t(4+1)+x(23)*t(5+1)+x(27)*t(6+1)+x(31)*t(7+1)+x(35)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));

g(108)=-
(b01+b11+b21)*(x(4)*t(0+1)+x(8)*t(1+1)+x(12)*t(2+1)+x(16)*t(3+1)+x(20)*
t(4+1)+x(24)*t(5+1)+x(28)*t(6+1)+x(32)*t(7+1)+x(36)*t(8+1))+(x(1)*t(0+1
)+x(2)*t(0+1)+x(3)*t(0+1)+x(4)*t(0+1)+x(5)*t(1+1)+x(6)*t(1+1)+x(7)*t(1+
1)+x(8)*t(1+1)+x(9)*t(2+1)+x(10)*t(2+1)+x(11)*t(2+1)+x(12)*t(2+1)+x(13)
*t(3+1)+x(14)*t(3+1)+x(15)*t(3+1)+x(16)*t(3+1)+x(17)*t(4+1)+x(18)*t(4+1
)+x(19)*t(4+1)+x(20)*t(4+1)+x(21)*t(5+1)+x(22)*t(5+1)+x(23)*t(5+1)+x(24
)*t(5+1)+x(25)*t(6+1)+x(26)*t(6+1)+x(27)*t(6+1)+x(28)*t(6+1)+x(29)*t(7+
1)+x(30)*t(7+1)+x(31)*t(7+1)+x(32)*t(7+1)+x(33)*t(8+1)+x(34)*t(8+1)+x(3
5)*t(8+1)+x(36)*t(8+1));

g(109)=-
(b02+b12+b22)*(x(37)*t(0+1)+x(41)*t(1+1)+x(45)*t(2+1)+x(49)*t(3+1)+x(53
)*t(4+1)+x(57)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(110)=-
(b02+b12+b22)*(x(38)*t(0+1)+x(42)*t(1+1)+x(46)*t(2+1)+x(50)*t(3+1)+x(54
)*t(4+1)+x(58)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(111)=-
(b02+b12+b22)*(x(39)*t(0+1)+x(43)*t(1+1)+x(47)*t(2+1)+x(51)*t(3+1)+x(55
)*t(4+1)+x(59)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(112)=-
(b02+b12+b22)*(x(40)*t(0+1)+x(44)*t(1+1)+x(48)*t(2+1)+x(52)*t(3+1)+x(56
)*t(4+1)+x(60)*t(5+1))+(x(37)*t(0+1)+x(38)*t(0+1)+x(39)*t(0+1)+x(40)*t(
0+1)+x(41)*t(1+1)+x(42)*t(1+1)+x(43)*t(1+1)+x(44)*t(1+1)+x(45)*t(2+1)+x
```

```
(46)*t(2+1)+x(47)*t(2+1)+x(48)*t(2+1)+x(49)*t(3+1)+x(50)*t(3+1)+x(51)*t
(3+1)+x(52)*t(3+1)+x(53)*t(4+1)+x(54)*t(4+1)+x(55)*t(4+1)+x(56)*t(4+1)+
x(57)*t(5+1)+x(58)*t(5+1)+x(59)*t(5+1)+x(60)*t(5+1));

g(113)=-
(b03+b13+b23)*(x(61)*t(2+1)+x(65)*t(3+1)+x(69)*t(4+1)+x(73)*t(5+1)+x(77
)*t(6+1)+x(81)*t(7+1)+x(85)*t(8+1)+x(89)*t(9+1)+x(93)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(114)=-
(b03+b13+b23)*(x(62)*t(2+1)+x(66)*t(3+1)+x(70)*t(4+1)+x(74)*t(5+1)+x(78
)*t(6+1)+x(82)*t(7+1)+x(86)*t(8+1)+x(90)*t(9+1)+x(94)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(115)=-
(b03+b13+b23)*(x(63)*t(2+1)+x(67)*t(3+1)+x(71)*t(4+1)+x(75)*t(5+1)+x(79
)*t(6+1)+x(83)*t(7+1)+x(87)*t(8+1)+x(91)*t(9+1)+x(95)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(116)=-
(b03+b13+b23)*(x(64)*t(2+1)+x(68)*t(3+1)+x(72)*t(4+1)+x(76)*t(5+1)+x(80
)*t(6+1)+x(84)*t(7+1)+x(88)*t(8+1)+x(92)*t(9+1)+x(96)*t(10+1))+(x(61)*t
(2+1)+x(62)*t(2+1)+x(63)*t(2+1)+x(64)*t(2+1)+x(65)*t(3+1)+x(66)*t(3+1)+
x(67)*t(3+1)+x(68)*t(3+1)+x(69)*t(4+1)+x(70)*t(4+1)+x(71)*t(4+1)+x(72)*
t(4+1)+x(73)*t(5+1)+x(74)*t(5+1)+x(75)*t(5+1)+x(76)*t(5+1)+x(77)*t(6+1)
+x(78)*t(6+1)+x(79)*t(6+1)+x(80)*t(6+1)+x(81)*t(7+1)+x(82)*t(7+1)+x(83)
*t(7+1)+x(84)*t(7+1)+x(85)*t(8+1)+x(86)*t(8+1)+x(87)*t(8+1)+x(88)*t(8+1
)+x(89)*t(9+1)+x(90)*t(9+1)+x(91)*t(9+1)+x(92)*t(9+1)+x(93)*t(10+1)+x(9
4)*t(10+1)+x(95)*t(10+1)+x(96)*t(10+1));

g(117)=-
(b04+b14+b24)*(x(97)*t(5+1)+x(101)*t(6+1)+x(105)*t(7+1)+x(109)*t(8+1)+x
(113)*t(9+1)+x(117)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
```

```
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(118)=-
(b04+b14+b24)*(x(98)*t(5+1)+x(102)*t(6+1)+x(106)*t(7+1)+x(110)*t(8+1)+x
(114)*t(9+1)+x(118)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(119)=-
(b04+b14+b24)*(x(99)*t(5+1)+x(103)*t(6+1)+x(107)*t(7+1)+x(111)*t(8+1)+x
(115)*t(9+1)+x(119)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x(
100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(1
05)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(11
0)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(115
)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(1
20)*t(10+1));

g(120)=-
(b04+b14+b24)*(x(100)*t(5+1)+x(104)*t(6+1)+x(108)*t(7+1)+x(112)*t(8+1)+
x(116)*t(9+1)+x(120)*t(10+1))+(x(97)*t(5+1)+x(98)*t(5+1)+x(99)*t(5+1)+x
(100)*t(5+1)+x(101)*t(6+1)+x(102)*t(6+1)+x(103)*t(6+1)+x(104)*t(6+1)+x(
105)*t(7+1)+x(106)*t(7+1)+x(107)*t(7+1)+x(108)*t(7+1)+x(109)*t(8+1)+x(1
10)*t(8+1)+x(111)*t(8+1)+x(112)*t(8+1)+x(113)*t(9+1)+x(114)*t(9+1)+x(11
5)*t(9+1)+x(116)*t(9+1)+x(117)*t(10+1)+x(118)*t(10+1)+x(119)*t(10+1)+x(
120)*t(10+1));

%  Équation 2.9
g(121)=-x(121)+0;
g(122)=-x(125)+1;
g(123)=-x(129)+1;
g(124)=-x(122)+1;
g(125)=-x(126)+0;
g(126)=-x(130)+1;
g(127)=-x(123)+1;
g(128)=-x(127)+1;
g(129)=-x(131)+0;
g(130)=-x(124)+1;
g(131)=-x(128)+0;
g(132)=-x(132)+1;

g(133)=-x(133)+0;
g(134)=-x(137)+1;
g(135)=-x(141)+1;
g(136)=-x(134)+1;
g(137)=-x(138)+0;
g(138)=-x(142)+1;
g(139)=-x(135)+1;
g(140)=-x(139)+1;
g(141)=-x(143)+0;
g(142)=-x(136)+1;
```

```
g(143)=-x(140)+0;
g(144)=-x(144)+1;

g(145)=-x(145)+0;
g(146)=-x(149)+1;
g(147)=-x(153)+1;
g(148)=-x(146)+1;
g(149)=-x(150)+0;
g(150)=-x(154)+1;
g(151)=-x(147)+1;
g(152)=-x(151)+1;
g(153)=-x(155)+0;
g(154)=-x(148)+1;
g(155)=-x(152)+0;
g(156)=-x(156)+1;

g(157)=-x(157)+0;
g(158)=-x(161)+1;
g(159)=-x(165)+1;
g(160)=-x(158)+1;
g(161)=-x(162)+0;
g(162)=-x(166)+1;
g(163)=-x(159)+1;
g(164)=-x(163)+1;
g(165)=-x(167)+0;
g(166)=-x(160)+1;
g(167)=-x(164)+0;
g(168)=-x(168)+1;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%    END OF FILE    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```