RAIRO. RECHERCHE OPÉRATIONNELLE

FOUAD BADRAN

Introduction de dates échues dans l'analyse PERT-coût

RAIRO. Recherche opérationnelle, tome 24, n° 1 (1990), p. 15-27

<http://www.numdam.org/item?id=RO_1990__24_1_15_0>

Recherche opérationnelle/Operations Research
(vol. 24, n° 1, 1990, p. 15 à 27)

INTRODUCTION DE DATES ÉCHUES DANS L'ANALYSE PERT-COÛT (*)

par Fouad BADRAN (1)

Résumé. — *On présente une méthode d'ordonnancement qui s'applique à des projets décomposés en tâches élémentaires partiellement ordonnées ; de plus on suppose l'existence de contraintes de « dates échues » imposées à certains événements. Les contraintes de dates échues provoquent des incompatibilités avec les durées des tâches élémentaires évaluées par l'utilisateur (qui sont fonction des moyens mis en œuvre) : on propose donc une méthode qui permet de lever ces incompatibilités en attribuant à chaque tâche une durée admissible, tout en minimisant le coût global d'exécution du projet.*

L'intérêt de cette méthode est de permettre la réalisation d'un logiciel d'aide à la décision. Il sera capable d'assister un utilisateur soumis constamment à des imprévus, en l'aidant à réévaluer les durées des tâches et à allouer au mieux les ressources. En conséquence, il évitera par une meilleure répartition, une accumulation de retards en fin de projet.

Mots clés : Ordonnancement ; dates échues ; allocation de ressources ; aide à la décision.

Abstract. — *We present a scheduling method for projects that can be split into a set of partially ordered elementary activities. We assume also the existence of constraints such as deadlines for some of the events. Since such deadlines may be incompatible with the chosen durations for the elementary activities, our method removes such incompatibilities, it reassigns a feasible duration to each activity so as to minimize the global cost of the project. The interest of the method is to allow the conception of a decision support system aimed at the assistance of users that are working under unforeseen conditions. The system helps the user for the reevaluation of the activities durations in such a way as to allow a good ressource allocation and to avoid a delay accumulation at the end of the projet.*

Keywords : Scheduling ; deadline ; ressource allocation ; decision support systems.

(*) Reçu en février 1989.
(1) CEDRIC, Centre d'Etudes et de Recherches en Informatique du C.N.A.M., Conservatoire National des Arts et Métiers, 292, rue Saint-Martin, 75141 Paris Cedex 03, France.

Recherche opérationnelle/Operations Research, 0399-0559/90/01 15 13/$ 3.30

1. INTRODUCTION

Nous présentons dans cette étude une méthode d'ordonnancement sous contraintes. Elle s'applique à des projets décomposés en tâches élémentaires partiellement ordonnées; à chacune des tâches est associée deux durées : la première, dite normale, représente le temps le plus favorable à son exécution, la seconde est la durée minimale en dessous de laquelle il est impossible de la réaliser. Toute durée comprise entre les deux précédentes est admissible et permet d'effectuer la tâche considérée; de plus, à chaque tâche, on associe un coût qui est fonction affine de la durée.

La recherche de l'ordonnancement de coût minimal pour une durée déterminée a été résolu de manière efficace par Fulkerson [6, 7]. Sa méthode utilise un graphe PERT muni d'une entrée et d'une sortie unique. Nous avons étendu cette méthode à des problèmes d'ordonnancement plus complexes. Nous avons autorisé l'existence de plusieurs tâches terminales ainsi que l'introduction de contraintes supplémentaires sous forme de dates échues imposées aux événements : les dates au plus tôt déterminées par l'ordonnancement ne doivent pas les dépasser.

L'introduction de ces dates échues fait apparaître initialement des marges totales négatives, révélant ainsi des incompatibilités avec les durées choisies pour les tâches élémentaires lors d'une mise à jour donnée. Notre méthode a pour objet de lever ces incompatibilités et de proposer une solution admissible au moindre coût.

L'intérêt de résoudre le problème sous cette forme est de permettre une utilisation itérative de la méthode en définissant les tâches par étapes, des plus générales aux plus particulières ou des plus proches dans le temps aux plus lointaines; les solutions apportées par la méthode à la première étape interviennent alors comme contraintes pour l'étape suivante. Nous nous sommes intéressés à ce problème afin de concevoir un logiciel d'ordonnancement de type SIAD (Système Interactif d'Aide à la Décision) pouvant assister un utilisateur qui, constamment sousmis à des imprévus, désire pourvoir intervenir pour modifier les contraintes de dates échues et des durées de tâches; c'est fréquemment le cas de grands projets, notamment dans les pays en voie de développement. Les logiciels existants sont mal adaptés aux utilisateurs dont l'environnement est soumis à plusieurs facteurs imprévisibles. En effet, de par leur conception, ces logiciels amènent l'utilisateur à fournir des données non fiables car elles ne tiennent compte ni des aléas de l'environnement, ni de sa propre expérience.

Lors de l'élaboration et de la préparation des données, l'utilisateur est amené à analyser en détail son projet, le découper en tâches élémentaires et estimer leur durée. C'est donc une « micro-analyse » du projet qui doit être faite. Or, nous avons remarqué que l'utilisateur a plutôt l'habitude d'opérer une « macro-analyse » de son projet, consistant à repérer des étapes ou événements importants par le fait qu'il peut leur attribuer des dates échues qu'il juge essentiel à ne pas dépasser. Ces dates échues traduisent en partie ses connaissances et son expérience dans le domaine, permettent d'assurer un contrôle du bon déroulement du projet, et de déconcentrer dans le temps les difficultés provoquées par les accumulations de retards qui, sinon, se répercutent inévitablement en fin de projet.

Cependant, avec le jeu de dates échues qu'il se fixe en tenant compte de son expérience dans le domaine, il peut « baliser » progressivement le déroulement du projet.

Nous proposons dans cet article une méthode permettant d'éliminer les incompatibilités induites par les dates échues, et ceci en attribuant des durées admissibles aux tâches élémentaires, ou en cas d'impossibilité, de proposer des durées qui « minimisent » les effets de ces incompatibilités.

Au paragraphe 2, nous formulons le problème, nous rappelons l'algorithme de Fulkerson et nous proposons une méthode de résolution dans le cadre d'un graphe PERT. Au paragraphe 3, nous suivons la même démarche dans le cadre de la méthode potentiels-tâches tout en précisant une adaptation de l'algorithme de Fulkerson. Au paragraphe 4, nous proposons une utilisation dans un logiciel d'aide à la décision. Enfin, au paragraphe 5, nous discutons de son implémentation.

2. LA MÉTHODE DANS LE CAS D'UN GRAPHE PERT

2.1. Formulation du problème

Étant donné un graphe PERT $G=(X, U)$, où $X=\{x_0, x_1, x_2, \ldots, x_{n-1}\}$ représente l'ensemble des sommets et U l'ensemble des arcs, on suppose que G admet un seul sommet entrée, noté x_0, qui représente l'événement « début du projet ». A chaque tâche (x_i, x_j) on associe une durée normale b_{ij} et une durée minimale a_{ij} $(a_{ij} \leqq b_{ij})$. On note K_{ij} le « coût » d'exécution de la tâche (x_i, x_j) correspondant à la durée normale, et on suppose que pour une durée t_{ij} $(a_{ij} \leqq t_{ij} \leqq b_{ij})$ le coût correspondant vaut $Q_{ij}=K_{ij}+(b_{ij}-t_{ij})\,C_{ij}$; C_{ij} représente ainsi le coût supplémentaire, pour une contraction d'une unité de temps, de la durée d'exécution de la tâche (x_i, x_j).

Le graphe PERT étant sans circuit, il admet au moins un sommet sortie; on note par S l'ensemble des indices de tous les sommets sorties. On se donne un sous-ensemble J de $\{0, 1, \ldots, n-1\}$ avec $J \subset S$ et une famille de dates échues $(T_j, j \in J)$; il s'agit d'attribuer à chaque tâche $(x_i, x_j) \in U$ une durée t_{ij} de telle manière que les dates au plus tôt (t_i) des événements $x_i \in X$ vérifient la condition $t_j \leqq T_j$ pour tout $j \in J$, et que la fonction coût $F = \sum_U Q_{ij}$ soit minimale, ce qui correspond à maximiser $C = \sum_U t_{ij} C_{ij}$.

En supposant que $t_0 = 0$, le problème se formule alors comme suit :

$$t_{ij} + t_i - t_j \leqq 0, \qquad \forall (x_i, x_j) \in U. \tag{2.1}$$

$$t_j \leqq T_j, \qquad \forall j \in J. \tag{2.2}$$

$$a_{ij} \leqq t_{ij} \leqq b_{ij}, \qquad \forall (x_i, x_j) \in U. \tag{2.3}$$

$$\text{MAX } C = \sum_U t_{ij} C_{ij}. \tag{2.4}$$

Ce problème a été bien résolu [6, 7] dans le cas où le graphe admet une seule sortie, soit x_{n-1}, et où J est réduit au seul élément $n-1$.

2.2. Ordonnancement à coût minimal : Algorithme de Fulkerson

On rappelle dans cette section l'algorithme de Fulkerson, la démonstration complète étant présentée dans [6, 7].

Étant donné un graphe PERT : $G = (X, U)$, on suppose que G admet un seul sommet entrée x_0, un seul sommet sortie x_n, et qu'il vérifie les conditions formulées au paragraphe 2.1 avec $J = \varnothing$ ou $\{x_n\}$. On note par T_{norm} la durée du projet correspondante aux durées normales et par $T_{\min}$ sa durée correspondante aux durées minimales; pour chaque durée T appartenant à l'intervalle $[T_{\min}, T_{\text{norm}}]$, il s'agit d'affecter à chaque tâche (x_i, x_j) une durée t_{ij} de telle sorte que la durée correspondante du projet soit T et le coût global $C(T)$ minimal. On démontre que la fonction $C(T)$ est linéaire par morceaux; l'algorithme de Fulkerson résout ce problème pour chaque T et détermine les points anguleux ainsi que les pentes des différents morceaux de la fonction $C(T)$.

On note t_i la date au plus tôt de l'événement x_i pour un choix de durées de tâches compatibles. On pose alors :

$$\tilde{a}(i, j; k) = \begin{cases} b_{ij} + t_i - t_j & \text{pour } k = 1. \\ a_{ij} + t_i - t_j & \text{pour } k = 2. \end{cases} \tag{2.5}$$

L'algorithme de Fulkerson s'énonce ainsi :

0. Construire le réseau de transport R en dédoublant chaque arc (x_i, x_j) de G en deux arcs $(i, j; k)(k=1,2)$, et en leur attribuant les capacités a_{ijk} avec $a_{ij_1} = C_{ij}$ et $a_{ij_2} = \infty$.

Initialiser le flot à zéro ($f_{ijk}=0$). En attribuant à chaque tâche sa durée normale, calculer les dates au plus tôt t_i des événements x_i.

On a dans ce cas $T_{\text{norm}} = t_{n-1}$. faire $T_1 = T_{norm}$.

1. SI $T_1 = T_{\min}$ ALORS Fin SINON passer à l'étape 2.

2. Considérer le sous-réseau partiel R' de R défini par les arcs $(i, j; k)$ pour lesquels $\tilde{a}(i, j; k) = 0$.

3. Maximiser la valeur du flot en cours, en opérant des modifications de flux uniquement sur les arcs du réseau R'. Soit v la valeur du flot modifié.

4. Considérer les deux ensembles d'arcs suivants :

$$A1 = \{(i, j; k)\ k=1,2 \mid i \text{ marqué}, j \text{ non marqué}, \tilde{a}(i, j; k) < 0\}$$

et

$$A2 = \{(i, j; k)\ k=1,2 \mid j \text{ marqué}, i \text{ non marqué}, \tilde{a}(i, j; k) > 0\}.$$

Calculer $\partial_1 = \underset{A1}{\text{Min}}[-\tilde{a}(i, j; k)]$, $\partial_2 = \underset{A2}{\text{Min}}[\tilde{a}(i, j; k)]$, $\partial = \text{Min}(\partial_1, \partial_2)$.

5. Pour tout événement x_i non marqué faire : $t_i = t_i - \partial$; faire : $T_1 = T_1 - \partial$, effacer toutes les marques générées à l'étape 3, revenir à l'étape 1.

Au cours des étapes 3 et 4, l'algorithme détermine deux points anguleux T_1 et $T_1 - \partial$ de la fonction de coût $C(T)$; v représente son coefficient directeur entre ces deux points. A chaque itération, t_i représente les dates au plus tôt correspondant à la durée T_1 du projet. Ainsi, pour tout $T_1 - d \in [T_1 - \partial, T_1]$, les dates au plus tôt sont égales à t_i pour les sommets marqués et à $t_i - d$ pour les sommets non marqués, et les durées optimales sont données par $d_{ij} = \text{MIN}(t_j - t_i, b_{ij})$.

2.3. Méthode de résolution du problème initial

On propose dans cette section une méthode de résolution du problème formulé à 2.1.

1. Introduire un nouvel événement correspondant à la fin du projet, soit x_n, qui devient alors la seule sortie du graphe.

2. En considérant les durées normales des tâches, calculer les dates au plus tôt des événements. On note par DN_i la date au plus tôt de l'événement x_i, DN_n représente la durée T_{norm} du projet.

3. Pour tout $j \in J$ SI $DN_j \leqq T_j$ ALORS la date échue attachée à cet événement ne présente pas d'incompabilité, SINON créer un nouvel événement, noté e_j, et trois tâches fictives :

(*a*) la première est représentée par l'arc (x_0, e_j), ayant des durées normale et minimale égales à T_j;

(*b*) la seconde est représentée par l'arc (e_j, x_n), ayant des durées normale et minimale égales à $T_{\text{norm}} - T_j$;

(*d*) la troisième est représentée par l'arc (x_j, e_j) ayant des durées normale et minimale nulles.

Le coût supplémentaire par unité contractée est nul pour les trois tâches. Soit G' le nouveau graphe ainsi obtenu.

4. Appliquer l'algorithme de Fulkerson au graphe G', afin de générer la fonction « coût » en fonction des durées du nouveau graphe événement-tâche.

5. SI la durée minimale T du graphe G' est telle que $T = T_{\text{norm}}$, ALORS conclure que l'algorithme a levé toutes les incompatibilités et retenir les durées des tâches t_{ij} trouvées par l'algorithme. SINON conclure qu'il est impossible de lever totalement les incompatibilités, proposer les durées t_{ij} trouvées comme les « meilleures » solutions.

Afin de justifier la validité de cette méthode, nous allons l'expliquer avec un seul événement à date fixe, puis avec deux.

Cas d'un seul événement

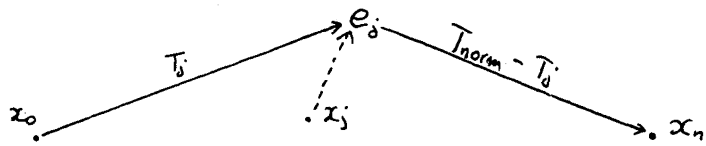

Figure 1.

Notons par x_j l'événement ayant une date échue T_j. Soit G' le graphe obtenu après la création des trois tâches. La durée normale optimale de G' est alors $T_{\text{norm}} + (DN_j - T_j) > T_{\text{norm}}$. Pour tout choix des durées t_{ij}, nous notons par t_j la date au plus tôt de l'événement x_j. A chaque contraction des durées lors de l'application de l'algorithme de Fulkerson (§ 2.2), et tant que $t_j > T_j$, les deux activités (x_j, e_j) et (e_j, x_n) sont critiques et tous les chemins critiques de G' passent par le sommet x_j; il est alors facile de remarquer que lors de la phase de marquage de l'algorithme de Ford-Fulkerson (étape 3), « x_n non marqué » équivaut à « x_j et e_j non marqués », ce qui montre bien que lors

de l'étape 5 les contractions s'effectuent au niveau des activités du sous-réseau formé par les chemins de x_0 à x_j.

Cas de deux événements

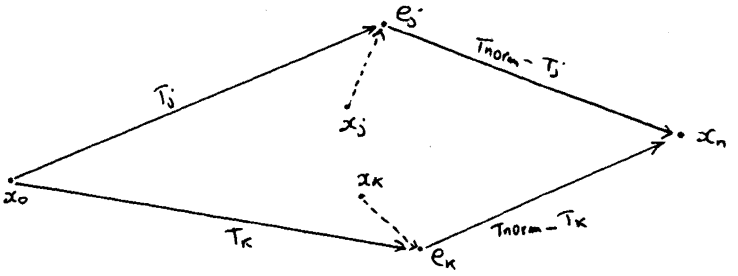

Figure 2.

Notons par x_j et x_k les deux événements avec leur date échue respective T_j et T_k et supposons que $T_k \leqq T_j$. Notons par G' le graphe obtenu après la création des six activités (étape 3 ci-dessus). La durée normale optimale de G' est alors $T' = T_{\text{norm}} + (T_j - TD_j)$. Pour tout choix des durées t_{ij}, nous notons par t_j la date au plus tôt de l'événement x_j et par t_k celle de l'événement x_k. Posons $d = T_j - T_k$; tant que $t_n > T' - d$, la contraction de G' se passe comme au cas précédent et uniquement au niveau de l'événement j, car l'événement x_k et l'activité (x_k, e_k) ne se trouvent pas encore sur les chemins critiques. Si $t_n \leqq T' - d$, alors les deux événements se comportent chacun comme au cas précédent et les contractions se font au niveau des deux sous-réseaux formés par les chemins de x_0 à x_j et de x_0 à x_k.

3. LA MÉTHODE DANS LE CAS POTENTIELS-TÂCHES

3.1. Formulation du problème

On note par $G = (Y, U)$ le graphe potentiels-tâches, avec $Y = \{y_0, y_1, \ldots, y_n\}$ l'ensemble des tâches, y_0 (resp. y_n) la tâche fictive début (resp. fin) et U l'ensemble des arcs associés aux contraintes potentielles.

A chaque tâche y_i, on associe sa durée normale b_i et sa durée minimale a_i, K_i étant le coût d'exécution de la tâche y_i correspondant à la durée normale b_i et C_i étant le coût supplémentaire par contraction d'une unité de temps; $Q_i = K_i + (b_i - d_i) C_i$ est le coût correspondant à la durée $d_i (a_i \leqq d_i \leqq b_i)$. On se donne un ensemble d'événements (ou étapes) $E = \{e_1, e_2, \ldots, e_p\}$, et des dates associées T_{e_i}, ainsi qu'une date T_n associée à y_n. On modifie alors le

graphe G en associant à chaque événement e_i une tâche fictive y_{e_i} (avec $b_{e_i} = a_{e_i} = 0$), les tâches précédentes (resp. suivantes) sont celles qui précèdent (resp. succèdent à) l'événement y_{e_i}; on note le nouveau graphe $G' = (Y', U')$. Il s'agit d'affecter à chaque tâche y_i une durée d_i de manière à résoudre le programme linéaire suivant :

$$d_i + t_i - t_j \leqq 0, \qquad \forall (y_i, y_j) \in U. \tag{3.1}$$

$$t_{e_i} \leqq T_{e_i}, \qquad \forall e_i \in E \tag{3.2}$$

$$t_n \leqq T_n \tag{3.3}$$

$$a_i \leqq d_i \leqq b_i, \qquad \forall y_i \in Y \tag{3.4}$$

$$\text{MAX } C = \sum_U d_i C_i. \tag{3.5}$$

On remarque que pour $E = \varnothing$, ce problème correspond au problème de l'ordonnancement à coût minimal.

3.2. Ordonnancement à coût minimal. Adaptation de l'algorithme de Fulkerson

On suppose que le graphe G vérifie les mêmes conditions qu'au paragraphe 3.1 avec $E = \varnothing$. On note par T_{norm} la durée du projet correspondant aux durées normales et par T_{min} sa durée correspondant aux durées minimales; il s'agit d'affecter à chaque tâche y_i une durée d_i de sorte que la durée correspondant du projet soit égale à $T \in [T_{\text{min}}, T_{\text{norm}}]$ et que le coût $C(T)$ soit minimal. En supposant $t_0 = 0$ le problème se formule ainsi [6, 7] :

$$d_i + t_i - t_j \leqq 0, \qquad \forall (y_i, y_j) \in U \tag{3.6}$$

$$t_n \leqq T \tag{3.7}$$

$$a_i \leqq d_i \leqq b_i, \qquad \forall y_i \in Y \tag{3.8}$$

$$\text{MAX } C = \sum_U d_i C_i. \tag{3.9}$$

Il est facile de voir qu'à l'optimum et pour toute tâche y_i on a :

$$d_i = \text{MIN}\,(\text{MIN}_{y_j \in \text{P}i}\,(t_j - t_i),\ b_i). \tag{3.10}$$

Les variables d_i et t_i étant de signe quelconque, le programme linéaire dual s'écrit alors :

$$\sum_{y_j \in \text{P}i} f_{ij} + g_i - h_i = C_i, \qquad \forall y_i \in Y \tag{3.11}$$

$$(3.12) \qquad \sum_{yj \in P_i} f_{ij} - \sum_{yj \in P_i^{-1}} f_{ji} = 0, \qquad \forall y_i \in Y$$

$$(3.13) \qquad v - \sum_{y_j \in P_n^{-1}} fj_n = 0.$$

avec $0 \leqq f_{ij}$, $0 \leqq g_i$ et $0 \leqq h_i$.

$$(3.14) \qquad \text{MIN } C' = Tv + \sum b_i g_i - \sum a_i h_i.$$

De (3.11) et (3.14) on déduit qu'à l'optimum on a :

$$(3.15) \qquad g_i = \text{MAX}(0,\ C_i - \sum_{yj \in P_i} f_{ij}) \text{ et } h_i = \text{MAX}(0,\ \sum_{yj \in P_i} f_{ij} - C_i)$$

On pose alors :

$$(3.16) \qquad F_{i_1} = \text{INF}(C_i,\ \sum_{yj \in P_i} f_{ij}) \qquad \text{et} \qquad F_{i_2} = \text{MAX}(0,\ \sum_{yj \in P_i} f_{ij} - C_i)$$

(3.14) est équivalente à :

$$(3.17) \qquad \text{MIN } C'' = Tv - \sum_{yj \in P_i} b_i F_{i_1} - \sum_{yj \in P_i} a_i F_{i_2}.$$

Le programme dual est équivalent à (3.12), (3.13), (3.16) et (3.17). Ce résultat suggère la construction, à partir du graphe G, du réseau de transport R' obtenu en dédoublant en deux sommets y_{i_1} et y_{i_2} tous les sommets y_i de G et en ajoutant deux arcs de y_{i_1} à y_{i_2} notés $(i;\ 1)$ et $(i;\ 2)$. Aux arcs de G, on attribue des capacités infinies et un coût nul par unité de flux, aux arcs de R' de type $(i;\ 1)$ une capacité C_i et un coût b_i, et aux arcs de R' de type $(i;\ 2)$ une capacité infinie et un coût a_i. Si on ajoute aux flux f_{ij} les quantités F_{i_1} et F_{i_2}, les relations précédentes montrent que le problème dual est un problème de flot sur R' qui minimise la fonction C'' (3.17).

Considérons le graphe PERT G' associé à G et contenant le maximum de tâches fictives (une pour chaque contrainte potentielle). Le problème de l'ordonnancement à coût minimal sur G' se formule par (2.1), (2.2) (avec $J = \{n\}$), (2.3) et (2.4) ; le dual de ce programme est équivalent au problème de flot sur R' vérifiant (3.17) [6, 7]. Ceci montre que la méthode pour résoudre le problème de l'ordonnancement à coût minimal sur un graphe potentiel-tâche consiste à appliquer l'algorithme de Fulkerson sur le graphe PERT maximal associé (contenant le maximum de tâches fictives).

La méthode de résolution qu'on propose dans ce cas est la suivante :

1. Construire le graphe G' obtenu en dédoublant en deux sommets, y_{i_1} et y_{i_2}, tous les sommets y_i de G et en ajoutant un arc de y_{i_1} à y_{i_2}. Aux arcs du type (y_{i_1}, y_{i_2}), attribuer les durées a_i et b_i comme durées minimales et normales ainsi que les coûts K_i et C_i. Attribuer aux arcs du type (y_{i_2}, y_{j_1}) des durées minimales et normales nulles ainsi que des coûts nuls. G' ainsi construit représente le graphe PERT associé à G ayant le maximum de tâches fictives.

2. Appliquer l'algorithme de Fulkerson (décrit au 2.2) au graphe G'. Retenir les durées attribuées aux arcs (y_{i_1}, y_{i_2}) comme durées des tâches y_i et retenir les dates de y_{i_1} comme dates au plus tôt du début de y_i.

3.3. Méthode de résolution du problème initial

On reprend les notations du paragraphe 3.1. La méthode s'énonce alors comme suit :

1. En procédant exactement comme au paragraphe 3.1, modifier le graphe G potentiels-tâches. Soit G' le nouveau graphe obtenu.

2. En considérant les durées normales des tâches, calculer les dates au plus tôt des débuts des tâches. On note par DN_i la date au plus tôt de la tâche y_i, DN_n représentant la durée T_{norm}.

3. Pour chaque tâche fictive y_{e_i} associée à l'événement e_i, SI $DN_{e_i} \leqq T_{e_i}$ ALORS la date fixe attachée à cet événement ne présente pas d'incompatibilité ; SINON créer un sommet $\tilde{y}_{e_i}$ et trois arcs :

(*a*) l'arc $(x_0, \tilde{y}_{e_i})$ ayant des durées normale et minimale égales à T_{e_i} ;

(*b*) l'arc $(\tilde{y}_{e_i}, x_n)$ ayant des durées normale et minimale égales à $T_{\text{norm}} - T_{e_i}$;

(*c*) l'arc $(y_{e_i}, \tilde{y}_{e_i})$ ayant des durées normale et minimale nulles. Le coût supplémentaire par unité contractée est nul pour les trois tâches. Soit G'' le nouveau graphe obtenu à partir de G'.

4. Appliquer l'algorithme de Fulkerson adapté au graphe G'' (§ 3.2) afin de générer la fonction des coûts en fonction de la durée totale de G''.

5. SI la durée totale minimale T de G'' est telle que $T = T_{\text{norm}}$ ALORS conclure que l'algorithme a levé toutes les incompatibilités et retenir les durées des tâches t_{ij} trouvées. SINON conclure qu'il est impossible de lever totalement les incompatibilités, et proposer les durées t_{ij} trouvées comme les « meilleures » solutions.

4. PROPOSITION D'UN LOGICIEL D'AIDE A LA DÉCISION

Afin de concevoir un logiciel qui réalise les objectifs fixés dans l'introduction, il est préférable de demander à l'utilisateur de proposer, pour chaque tâche, un coût C (sous forme de coefficients) traduisant la priorité qu'il accorde à la contraction de cette tâche. Les coefficients apparaissent comme étant des pénalités attachées aux tâches. Le programme résout les incompabilités entre les dates échues imposées et la « micro-analyse » du projet et ceci, en minimisant au mieux la pénalité totale (coût des contractions), c'est-à-dire en satisfaisant au mieux ses préférences.

La pénalité accordée à une activité donnée dépend, d'une part, de facteurs quantitatifs tels que le coût de contraction et, d'autre part, des facteurs qualitatifs liés aux difficultés organisationnelles et matérielles que peut provoquer une telle contraction.

De plus le logiciel permet à l'utilisateur :

– D'assurer une cohérence des données de son projet en définissant les tâches élémentaires jusqu'à une date à partir de laquelle il est incapable de proposer des données significatives. Ainsi, le graphe courant contient plusieurs tâches terminales auxquelles l'utilisateur impose des dates échues pour leur événement fin.

– De « baliser » l'ordonnancement du projet en évitant une accumulation de retards par l'introduction de contraintes de type dates échues imposées à des événements intermédiaires.

Avec le jeu des priorités à la contraction des différentes tâches, le logiciel propose des durées réalisables respectant au mieux ses préférences. En interaction avec le logiciel, l'utilisateur réalise plusieurs essais, examine différentes solutions, modifie ses données et finalement retient une répartition de durées qu'il juge convenable. Par ailleurs, cette solution retenue le guidera pour la période qui suit, dans sa politique d'allocation de ressources humaines et matérielles aux différentes tâches.

5. IMPLÉMENTATION

Nous discutons dans ce paragraphe des algorithmes et des structures de données pour une implémentation efficace de la méthode décrite au paragraphe 4. L'étape principale de cette méthode consiste en l'application de l'algorithme de Fulkerson à un graphe approprié. Nous pouvons résumer

l'algorithme de Fulkerson de la manière suivante :

FAIRE $T := T_{\text{norm}}$ et $f_{ij,k} := 0$ pour tout $(i, j; k)$ du réseau R; (étape 0)
TANT QUE $T > T_{\min}$ FAIRE
DEBUT
Résoudre le problème du flot maximal en améliorant uniquement le flux sur les arcs du sous-réseau R' correspondant à $\tilde{a}(i, j; k) = 0$ (2.5); modifier le sous réseau (étape 4); FAIRE $T = T - \partial$;
FIN;

Posons $d = T_{\text{norm}} - T_{\min}$; nous remarquons que le problème du flot maximal sera résolu au plus d fois. L'algorithme le plus efficace de ce problème est proposé par Dinic [2, 12]. L'efficacité de la méthode de Dinic dépend de l'implémentation de la procédure qui détermine le flot complet sur le référent. Plusieurs méthodes permettent la détermination de flots complets : la méthode « en profondeur d'abord » présente une complexité de $O(n^2 m)$ [2,12], l'algorithme dit « des vagues » proposé par Tarjan [12] permet une implémentation efficace de la méthode des préflots et présente une complexité de $O(n^3)$ [8,12], et enfin l'utilisation des structures de données proposées par Sleator et Tarjan [10, 12] pour fusionner et éclater des arborescences permet une complexité de $O(n.m(\log n))$ [11, 12]. Il suffit alors de multiplier ces ordres de grandeur par d pour avoir, suivant l'implémentation, l'ordre de complexité de l'agorithme de Fulkerson.

Le réseau R' sur lequel on applique l'algorithme de Dinic est très peu dense et le nombre de sommets est limité (taille d'un projet). De plus, à chaque appel de Dinic, le flot au départ n'est pas nul. Ainsi, tenant compte des considérations précédentes, l'algorithme de Dinic avec détermination de flots complets sur le référent par une méthode « en profondeur d'abord », réalise une implémentation efficace de la méthode.

6. CONCLUSION

Nous avons présenté dans cet article une méthode permettant de concevoir un logiciel d'ordonnancement de type SIAD, pouvant assister un utilisateur qui, constamment soumis à des imprévus, désire pouvoir intervenir pour changer des contraintes de dates échues et les durées de tâches.

En effet, dans un environnement soumis à des imprévus, l'estimation de la durée des tâches élémentaires est un problème en soi pour un utilisateur. La difficulté de l'estimation des durées provient du fait qu'il ne peut pas avancer

une prévision sûre des moyens et efforts qu'il faudrait allouer à chacune de ces tâches.

Nous suggérons l'introduction des contraintes supplémentaires sous forme de dates échues imposées à certains événements. Ces dates échues, traduisant en partie les connaissances dans le domaine, permettant d'assurer un contrôle du bon déroulement du projet et de déconcentrer dans le temps les difficultés provoquées par les accumulations de retard qui, sinon, se répercutent inévitablement en fin de projet.

L'introduction de ces dates échues fait apparaître initialement des marges totales négatives, révélant ainsi des incompatibilités avec les durées normales des tâches élémentaires lors d'une mise à jour. Il faut donc contracter la durée de certaines tâches afin de lever ces incompatibilités tout en minimisant le « coût » de cette opération.

Cet article présente dans les cas PERT et Potentiel-Tâche une méthode de contraction. Cette méthode utilise l'algorithme de Fulkerson (PERT-Coût) [6, 7] dans le premier cas et une adaptation de celui-ci dans le second cas. D'autre part, l'algorithme de Fulkerson ramène le problème à un problème de flots sur des réseaux particuliers, ce qui permet une implémentation exploitable de la méthode.

BIBLIOGRAPHIE

[1] J. CARLIER et P. CHRETIENNE, *Problèmes d'Ordonnancement : Modélisation/Complexité/Algorithmes*, Masson Paris, 1988.

[2] E. A. DINIC, *Algorithm for Solution of a Problem of Maximum flow in a Network with Power Estimation*, Soviet Math. Dolk., vol. 11, 1970, p. 1277-1280.

[3] R. FAURE, C. ROUCAIROL et P. TOLLA, *Chemins et Flots, Ordonnancements*, Gauthier-Villars, Paris, 1976.

[4] R. FAURE, *Précis de Recherche Opérationnelle*, Dunod, Paris, 1978.

[5] L. R. FORD et D. R. FULKERSON, *Maximal Flow Through a Network*, Canad. J. Math., vol. 8, 1956, p. 399-404.

[6] L. R. FORD et D. R. FULKERSON, *Flows in Networks*, Princeton Univ. press., 1962.

[7] D. R. FULKERSON, *A Network Flow Computation for Project Cost Curves*, Management science, vol. 7, 1961, p. 167-178.

[8] A. V. KARZANOV, *Determining the Maximal Flow in a Network by the Method of Preflows*, Soviet Math. Dokl., vol. 15, 1974, p. 434-437.

[9] ROSEAUX, *Graphes : Leurs usages, leurs algorithmes*, Masson, tome 1, 1986.

[10] D. D. SLEATOR et R. E. TARJAN, *A Data Structure for Dynamic Tree*, J. Comput. System Sci., vol. 24, 1983.

[11] D. D. SLEATOR, An $O(nm \log n)$ *Algorithm for Maximum Network Flow*, Tech. Rep. STANCS-80-831, Computer science Dept., Stanford Univ., 1980.

[12] R. E. TARJAN, *Data Structures And Network Algorithms*, Wiley, 1983.