REVUE FRANÇAISE D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. SÉRIE VERTE

YVES TABOURIER

Problème d'ordonnancement à contraintes purement disjonctives

Revue française d'informatique et de recherche opérationnelle. Série verte, tome 3, n° V3 (1969), p. 51-65

<http://www.numdam.org/item?id=RO_1969__3_3_51_0>

R.I.R.O.
(3e année, V-3, 1969, p. 51-66)

PROBLEME D'ORDONNANCEMENT A CONTRAINTES PUREMENT DISJONCTIVES

Yves Tabourier (1)

Résumé. — *Parmi les problèmes d'ordonnancement à contraintes disjonctives, le cas particulier où il n'y a que de telles contraintes est propice à l'essai de nouveaux outils ; on présente ici un exemple, un algorithme du type SEP et un programme expérimental.*

INTRODUCTION

De très nombreux travaux ont été consacrés aux problèmes d'ordonnancement à contraintes disjonctives ; la liste proposée en bibliographie ([1], [2], [3], [4], [6], [8]) ne représente qu'une faible part des études consacrées à cet important problème, surtout si l'on y joint les recherches consacrées au « problème du voyageur de commerce » dans un esprit « ordonnancement ».

Le cas particulier où un problème d'ordonnancement n'est soumis qu'à des contraintes disjonctives, est propice à l'expérimentation de nouveaux outils.

Nous avons construit l'algorithme et le programme TADSEP à l'occasion d'un tel problème.

Cet algorithme est fondé sur la procédure SEP (Séparation et Évaluation Progressives). Il apparaît donc comme une continuation naturelle des travaux effectués par la Direction Scientifique de la SEMA, concernant aussi bien la prise en compte des contraintes disjonctives en ordonnancement ([4], [6]) que la programmation linéaire en variables binaires ([5]).

La première partie de cet article est consacrée au problème posé. Elle est suivie d'un exposé de l'algorithme TADSEP, puis d'un bref aperçu sur le programme.

(1) SEMA.

I. LE PROBLEME

Nous exposerons d'abord le problème concret qui a servi de support à nos réflexions. Un second paragraphe sera consacré au problème plus général qui en découle, et aux quelques notions de la théorie de l'ordonnancement dont nous avons besoin.

1. LE PROBLEME CONCRET

Dans les premiers mois de 1969, nous eûmes à étudier la synchronisation des feux le long d'un axe routier près de Paris, et notre première tâche fut de calculer l'ordonnancement des feux verts à chaque cerrefour.

1.1. Définitions

Cycle d'un feu. — Un feu montre successivement les états suivants : vert, orange, rouge, rouge et orange (dans certains pays), puis de nouveau : vert, orange, etc... Dans de nombreuses réalisations, la durée de chacun des 4 (ou des 3) états est donnée, et la succession d'états se produit donc avec une certaine périodicité. Les durées des états définissent ce que l'on appelle le cycle du feu, et leur somme (la période) est appelée durée du cycle.

Cycle adjoint. — Dans de nombreuses études théoriques, le cycle de chaque feu est remplacé par un cycle de même durée, considéré comme équivalent, où seuls le vert et le rouge ont une durée non nulle : on l'appelle le cycle adjoint. Dans la suite, nous ne considérerons que des feux à deux états : vert et rouge.

Courant de trafic. — On appelle courant de trafic la partie du trafic contrôlée par un feu, selon les règles du code de la route.

Flot de trafic. — Un flot de trafic est un courant de trafic auquel on adjoint un débit, généralement mesuré en véhicules/heure.

Phase de signalisation. — Pour des raisons de sécurité évidentes, qui sont la raison d'être des feux, les cycles des feux d'un carrefour sont synchronisés de façon à montrer vert en même temps à des courants de trafic qui ne se gênent pas (ou peu). Une combinaison (acceptable) d'états verts et rouges montrés par les différents feux du carrefour pendant une durée d est appelée une phase de signalisation, de durée d.

Cycle d'un carrefour. — Dans toutes les réalisations, les feux ont tous la même durée de cycle, ce qui donne au phénomène global de succession des phases un aspect périodique, avec la même période, appelée durée du cycle.

Un cycle est donc une succession de phases (acceptables) que l'on reproduit périodiquement. Cette succession doit être telle que les états qu'elle induit sur les feux permettent de rencontrer sur chacun d'eux

un cycle de feu de même durée, au sens défini plus haut pour les cycles de feux. C'est ce que l'on entend lorsque l'on dit que « chaque feu montre le vert une seule fois par cycle ».

Point rouge. — Le cycle du carrefour provient donc d'une imbrication de cycles de feux de même durée. Il peut avoir une particularité spéciale : choisissons une origine des temps quelconque, et faisons croître le temps t jusqu'à la durée d du cycle. Il peut y avoir ou non une valeur privilégiée t_0, $0 \leqslant t_0 \leqslant d$, telle que, pour aucun feu, on n'ait à la fois l'état vert aux instants $t_0 - \varepsilon$ et $t_0 + \varepsilon$, $\varepsilon > 0$ étant arbitrairement petit. Si c'est le cas, c'est que tout feu vert immédiatement avant t_0 devient rouge à l'instant t_0, et que tout feu vert à partir de t_0 était rouge avant.

On dit que le cycle contient un « point rouge » (il peut y en avoir plus) (1). La figure 1 illustre cette notion : il s'agit de deux diagrammes à barres avec le temps en abscisse et les différents feux en ordonnée, chaque barre représentant l'état vert d'un feu.

Nota. — Le lecteur intéressé par des détails plus complets sur ces notions peut les trouver dans [7].

1.2. Le problème proprement dit

On considère dans la journée un certain nombre de périodes pendant lesquelles le trafic est relativement stable. Pour chacune de ces grandes périodes on va calculer la synchronisation des feux à l'intérieur de chaque carrefour, selon le point de vue périodique du paragraphe précédent et avec un « point rouge ».

La première idée est de garantir à chaque feu un temps de vert par cycle proportionnel à ses besoins, représentés par le quotient :

$$\frac{\text{flot du trafic contrôlé par le feu}}{\text{largeur de voie contrôlée par le feu}}$$

La durée de vert du feu qui a les plus grands besoins est prise comme unité de temps. La durée relative de vert de tous les feux étant fixée, on recherche un cycle de durée la plus courte possible, de façon à maximiser pour chaque feu la proportion du temps pendant laquelle il est vert.

On peut ensuite appliquer des homothéties sur les durées réelles des cycles en vue de synchroniser les carrefours le long de l'axe.

Il nous faut maintenant examiner les contraintes ; les plus banales sont celles de sécurité : si deux feux contrôlent des courants de trafic qui

(1) L'intérêt de cette notion résulte des simplifications qu'elle entraîne pour le matériel et pour les changements de rythme des feux.

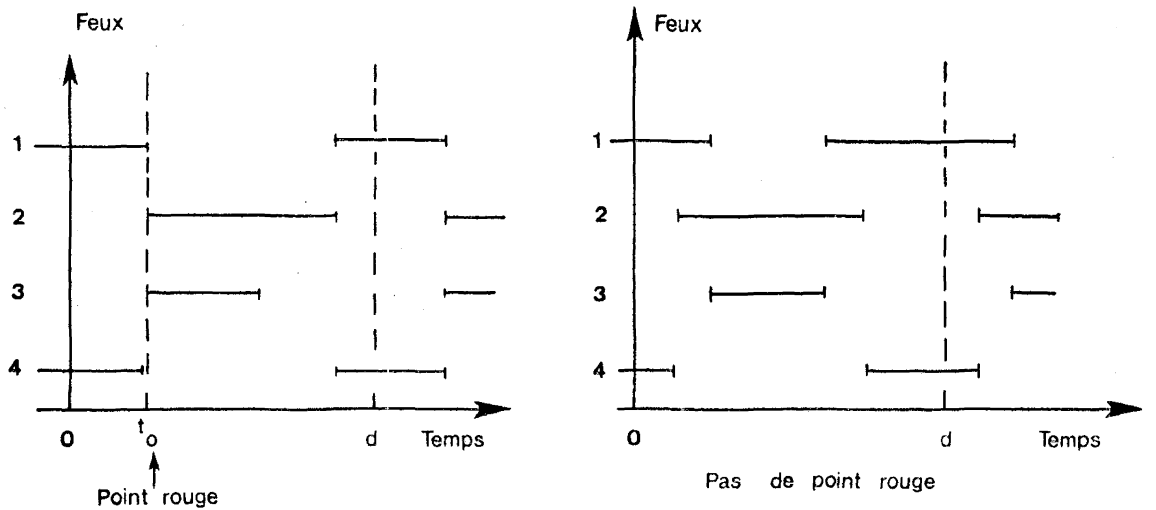

Feux
1
2
3
4
0
t_o
d
Temps
Point rouge
Feux
1
2
3
4
0
d
Temps
Pas de point rouge

Figure 1

empruntent des zones géographiques communes, on décide en général d'interdire que ces deux feux soient verts en même temps.

Une autre contrainte est celle du point rouge. Si l'on prend un point rouge comme origine des temps (début de cycle), on va le retrouver (fin de cycle) au bout de la durée d du cycle, puis en 2 d, 3 d, etc... Entre les instants 0 et d, ou $i \times d$ et $(i + 1) \times d$, chaque feu doit recevoir l'intégralité de son temps de vert garanti.

2. LE PROBLEME THEORIQUE

Il nous faut d'abord préciser le vocabulaire emprunté à la théorie de l'ordonnancement ; après quoi nous exposerons le problème théorique tiré du problème concret décrit ci-dessus.

2.1. Vocabulaire tiré de la théorie de l'ordonnancement

Un problème *d'ordonnancement* consiste en l'établissement d'un *calendrier* pour accomplir un certain nombre de *tâches*, dont la *durée* est généralement supposée connue. L'ensemble des tâches, lorsque chacune ne doit être exécutée qu'une fois, est généralement appelé *projet*. Souvent le calendrier doit être établi en optimisant un certain *objectif* : délai total de réalisation du projet, coût total du projet, etc... ; en ordonnancement de production, on a souvent affaire à des tâches répétitives (fabrication d'une série de pièces) et l'objectif est souvent différent : période du phénomène de fabrication (cadence de production), nombre de pièces en cours de réalisation, etc...

Les tâches du projet sont généralement soumises à des *contraintes* ; ces contraintes peuvent être temporelles : par exemple si une tâche T_j ne peut commencer qu'à partir d'un délai a_{ij} après le début d'une autre tâche T_i, ou si la tâche T_i ne peut commencer avant une date a_{0i}, ou encore si la tâche T_j doit commencer au plus tard un certain délai $(-a_{ji})$ après le début de la tâche T_i. Toutes ces contraintes peuvent se mettre sous la forme $t_j - t_i \geqslant a_{ij}$, que l'on appelle inégalité de potentiel. Parmi les contraintes temporelles, on peut distinguer celles qui doivent être toutes vérifiées, formant un système de contraintes *conjonctives*, de certains groupes de contraintes dont une seule a besoin d'être vérifiée (groupes non conjonctifs). Parmi les groupes non conjonctifs, certains sont formés de deux contraintes, qui ne peuvent être vérifiées simultanément, et entre lesquelles il faut donc choisir : on dit que l'on a affaire à des *contraintes disjonctives* ou *paires de disjonction*. D'autres contraintes peuvent ne pas être temporelles : ce sont par exemple des contraintes « de charge » (volume de main-d'œuvre limité à un certain seuil, etc...).

Problème central. — On appelle problème central de l'ordonnancement un problème d'ordonnancement où toutes les contraintes sont des contraintes temporelles conjonctives et où l'objectif est de minimiser la durée totale du projet. Il existe un grand nombre de représentations

de ce problème sous forme de graphe (MPM, Pert, etc...). Presque toutes les méthodes s'appuient sur la notion de *chemin critique*, c'est-à-dire de plus long chemin dans un graphe, entre deux sommets représentant les étapes de début du projet et de fin du projet, ce plus long chemin représentant la limite inférieure de la durée du projet.

De même, le plus long chemin entre le début du projet et le début d'une tâche donne le *début au plus tôt* de la tâche, tandis que le plus long chemin entre la tâche et la fin du projet dont la date est supposée connue, permet d'obtenir le *début au plus tard* de cette tâche.

2.2. Le problème théorique posé

Le fait que le feu i doit être vert pendant la durée d_i peut être appelé la tâche T_i. *S'il* y a N feux, on a ainsi N tâches T_i, $i = 1, 2 \ldots N$. On leur ajoutera deux tâches « fictives » correspondant au point rouge : le début T_0 du cycle et sa fin T_{N+1}.

Certaines paires de feux ne peuvent montrer le vert en même temps, pour des raisons de sécurité. Soit $P_k = (i, j)$ une telle paire : cela signifie que les tâches T_i et T_j ne peuvent se recouvrir. Si D_i et D_j sont les instants début de T_i et T_j, on doit satisfaire une des deux contraintes :

$$\text{ou} \begin{cases} D_i + d_i \leqslant D_j \\ D_j + d_j \leqslant D_i \end{cases}$$

ce qui montre que l'on a affaire à des contraintes disjonctives.

Pour plus de généralité, on peut remplacer ces contraintes par d'autres :

$$\text{ou} \begin{cases} D_i + a_{ij} \leqslant D_j \\ D_j + a_{ji} \leqslant D_i \end{cases}$$

car dans certains problèmes le délai entre D_i et D_j n'est pas forcément la durée de la tâche T_i, et peut dépendre de la tâche T_j. Nous supposerons néanmoins les $a_{ij} > 0$. Nous appellerons M le nombre de telles paires de disjonction. A toute paire de feux incompatibles (paire de disjonction) $P_k = (i, j)$, où i et j sont supposés écrits dans l'ordre des numéros, on peut associer une variable X_k:

$$\begin{cases} X_k = 1 \text{ si on a décidé de faire } T_i \text{ avant } T_j \\ X_k = 0 \text{ si on a décidé de faire } T_i \text{ après } T_j \end{cases}$$

Si maintenant on choisit une valeur, 0 ou 1, pour chaque X_k, on ne garde qu'une seule contrainte pour chaque paire de disjonction. Ces contraintes, unies aux contraintes triviales :

$$D_0 \leqslant D_i \text{ et } D_i + d_i \leqslant D_{N+1},\ i = 1,2 \ldots N.$$

forment donc un système conjonctif. Chaque M-vecteur booléen

$$[X] = (X_1, X_2 \ldots X_M)$$

conduit donc à un problème central d'ordonnancement, que l'on sait résoudre par des méthodes classiques de chemin critique. Certains vecteurs X_k sont inacceptables, ne respectant pas la transitivité de l'ordre chronologique. Il faut choisir un vecteur acceptable $[X]$ tel que le délai de réalisation du projet, pour son problème central, soit minimal.

II. PRINCIPES DE LA SOLUTION PROPOSEE

La solution que l'on propose ici est fondée sur la procédure SEP. Les concepts SEP nécessaires à sa compréhension sont exposés à mesure des besoins. Le premier paragraphe donne quelques définitions *ex-abrupto*, tandis que les autres introduisent un par un les concepts un peu plus difficiles.

1. DEFINITIONS

1.1. Arbitrage

C'est une décision selon laquelle, pour une paire de disjonction $P_k = (T_i, T_j)$, on choisit d'effectuer T_i avant T_j (ou l'inverse) en posant $X_k = 1$ (ou $X_k = 0$).

1.2. Famille d'arbitrages

Un ensemble d'arbitrages concernant une partie de l'ensemble des paires incompatibles ; en d'autres termes, un vecteur $(X_1, X_2, \ldots, X_k, \ldots X_M)$ dans lequel certaines composantes peuvent être laissées en blanc (composantes non décidées).

1.3. Famille d'arbitrages cohérente

Une famille d'arbitrages qui respecte la transitivité de l'ordre chronologique.

1.4. Famille complète d'arbitrages

Un vecteur $(X_1, X_2, \ldots, X_k, \ldots, X_M)$ où toutes les composantes ont reçu une valeur.

Une famille complète d'arbitrages peut être cohérente ou non. Les familles complètes d'arbitrages cohérentes (f.c.a.c.) sont les solutions réalisables du problème. Celles qui permettent la durée minimale d'exécution du projet sont les solutions optimales cherchées.

2. LES FAMILLES D'ARBITRAGES COMME SOMMETS D'UNE ARBORESCENCE : PRINCIPE DE SEPARATION

Étant donnée une famille d'arbitrages cohérente, supposons que nous disposions d'une règle (appelons-la « règle-S ») qui permette de choisir une des paires dont la variable X_k n'est pas décidée par la famille d'arbitrages. Nous pouvons maintenant obtenir deux nouvelles familles, l'une en posant $X_k = 1$, l'autre en posant $X_k = 0$. Nous disons que nous avons séparé la famille initiale. Si c'était une famille complète, elle ne serait pas séparable et nous la dirions terminale. Le choix d'une variable X_k non décidée au moyen d'une règle-S et le remplacement de sa famille d'arbitrages par deux nouvelles familles s'appelle *principe de séparation.*

La relation de filiation par laquelle une famille d'arbitrages en fournit deux nouvelles grâce au principe de séparation, est représentée habituellement par une arborescence. Dessinons donc un premier sommet, représentant la famille d'arbitrages « vide » (tous les X_k en blanc), et appliquons-lui le principe de séparation, ce qui permet de construire deux nouvelles familles d'arbitrages. Représentons-les par deux nouveaux sommets : nous avons « séparé » le premier (fig. 2).

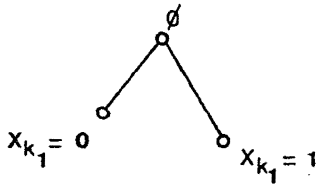


Figure 2

L'un d'entre eux peut être séparé à son tour, donnant lui-même deux nouveaux sommets (fig. 3) : nous sommes en train de dessiner une arborescence. Si nous y prenons un sommet, et si nous grimpons le long de l'arborescence, nous obtenons une famille d'arbitrages : il est commode de dire que le sommet représente cette famille d'arbitrages (dans la figure 3, le sommet D représente : $X_{k_2} = 0$, puis en remontant en C : $X_{k_1} = 1$, les autres X étant non décidés).

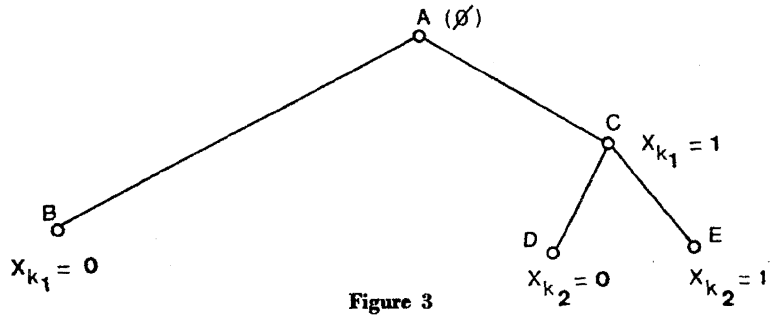


Figure 3

Nous ne voulons pas tracer de sommets non cohérents (représentant des familles non cohérentes). Aussi, lorsque nous avons construit un nouveau sommet, si sa famille d'arbitrages impose de nouveaux choix pour respecter la transivité de l'ordre chronologique, nous faisons ces choix aussitôt.

EXEMPLE : Supposons les paires (1.2), 1.3), (2.3) incompatibles (fig. 4).

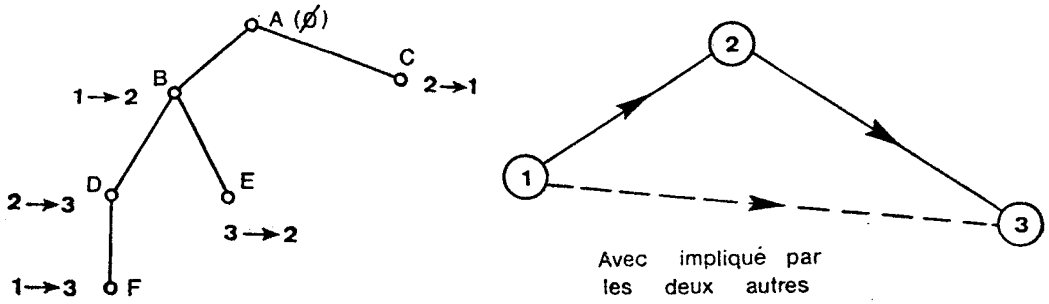


Figure 4

Dès que le sommet D est dessiné, il faut dessiner F, car si 1 précède 2 et 2 précède 3, 1 doit précéder 3. Le cas que nous avons montré ici est le plus simple : dans la figure 5, si $1 \rightarrow 2$ et $3 \rightarrow 4$ ont déjà été choisis, et si (2.3), (1.3), (1.4) sont d'autres paires incompatibles le choix de $2 \rightarrow 3$ implique que nous ayons $1 \rightarrow 3$ et $1 \rightarrow 4$. Nous n'aborderons pas ici la technique employée, qui dépend de l'organisation des données en machine: l'essentiel est que, par ce moyen, nous sommes sûrs de n'obtenir que des familles d'arbitrages cohérentes, si bien que toute famille complète obtenue sera une solution réalisable.

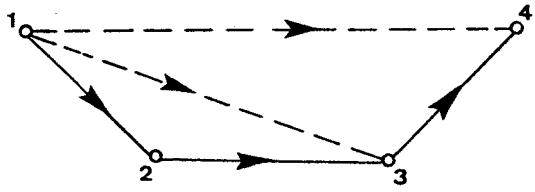
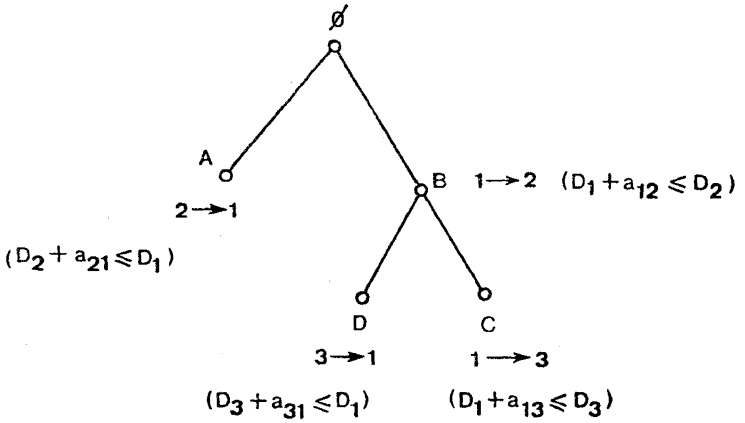


Figure 5

3. EVALUATION DES FAMILLES D'ARBITRAGES ENGENDREES

Nous devons maintenant chercher des solutions optimales. Pour ce but nous allons d'abord associer à tout sommet de l'arbre, c'est-à-dire à toute famille d'arbitrages cohérente, un graphe orienté défini comme suit, et appelé « graphe conjonctif du sommet » : chaque tâche est représentée par un sommet, y compris T_0 et T_{N+1}, début et fin du projet, souvent appelées « fictives », par opposition aux autres tâches T_i, $1 \leqslant i \leqslant N$, appelées « vraies ». Chaque contrainte conjonctive est représentée par

un arc : de la tâche de début partent N arcs allant vers chaque tâche réelle, et de chaque tâche réelle un arc part vers la tâche de fin ; les premiers ont une longueur nulle ; les longueurs des seconds sont les durées des tâches correspondantes. Enfin, chaque contrainte décidée de la famille d'arbitrages, devenue donc conjonctive, est représentée par un arc issu de la première tâche i vers la seconde j, selon l'arbitrage fait. La longueur de cet arc est a_{ij} (i et j définis ci-dessus), c'est-à-dire le délai tiré de la contrainte choisie (fig. 6).



Graphe conjonctif du sommet C :

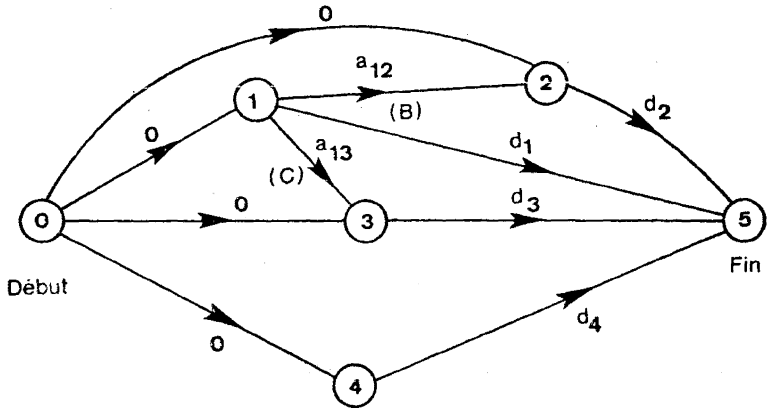


Figure 6

La longueur d'un chemin est la somme des longueurs de ses arcs. Le chemin critique est le plus long chemin qui joint T_0 à T_{N+1}. En séparant la famille d'arbitrages, nous allons simplement ajouter des arcs, si bien que la longueur du chemin critique ne peut pas diminuer : si nous appe-

lons « évaluation » de la famille d'arbitrages la longueur de ce chemin critique, cette évaluation est un minorant de la durée minimum du projet que l'on peut obtenir en séparant la famille jusqu'à l'obtention d'une solution. Quand la famille est complète (sommet terminal), son évaluation est égale à la durée minimum du projet pour les choix effectués sur les X_k.

4. CHOIX DU SOMMET A SEPARER CONVERGENCE DE L'ALGORITHME

A chaque pas, on choisit un sommet non séparé qui fournit la plus petite évaluation, et on tente de lui appliquer le principe de séparation. Si sa famille n'est pas complète, on obtient deux nouvelles familles qui sont évaluées à leur tour. Si la famille est complète, elle correspond à une solution optimale : son évaluation est juste égale au délai du projet qui lui correspond, et elle est un minorant de toutes les solutions que l'on peut obtenir en séparant les autres sommets. D'autre part, comme l'arbre est fini, on est sûr d'obtenir à un moment une famille complète. Ainsi on est sûr d'obtenir une solution optimale. Signalons que la convergence est rendue bien plus rapide si, lorsque deux sommets ont la même évaluation, on choisit celui dont la famille d'arbitrages est la plus fournie.

5. LA REGLE-S DU PRINCIPE DE SEPARATION

Il nous reste maintenant à préciser la « règle-S ». Seule son existence est nécessaire pour prouver que l'algorithme conduit à l'optimal. La définition précise de la règle-S est néanmoins capitale, car la taille de l'arborescence construite par l'algorithme dépend directement de la règle-S choisie.

Nous avons essayé pour cela plusieurs méthodes heuristiques, et nous avons obtenu les meilleurs résultats avec la suivante :

Quand on a choisi un sommet de l'arbre, on construit sa famille d'arbitrages et son graphe conjonctif. Supposant le délai égal à l'évaluation du sommet, on calcule pour chaque i son début au plut tôt $\lambda(i)$, et son début au plus tard $\lambda'(i)$. Pour chaque paire $P_k = (T_i, T_j)$ non encore décidée, on calcule alors :

$$\begin{cases} A_{ij} = \lambda(i) - \lambda'(j) + a_{ij} \\ A_{ji} = \lambda(j) - \lambda'(i) + a_{ji} \\ \Delta_k = \text{Max}\,(A_{ij}, A_{ji}) \end{cases}$$

et on choisit la paire P_k qui maximise Δ_k.

L'idée est que, si par exemple $A_{ij} > 0$, le choix d'accomplir T_i avant T_j augmentera évidemment la longueur du chemin critique de la valeur

A_{ij}. Plus grand est A_{ij}, plus on peut s'attendre que le sommet obtenu en choisissant $X_k = 1$ aura une évaluation supérieure au délai optimal, si bien qu'il ne sera plus séparé : les mauvaises combinaisons de choix seront alors découvertes et éliminées très haut dans l'arborescence.

Nota. — Le lecteur désireux de renseignements plus approfondis sur la procédure SEP en général, pourra les trouver dans [5].

III. LE PROGRAMME TADSEP

Nous avons programmé la procédure en FORTRAN, pour le CDC 6600 de la SEMA. Ce programme, TADSEP, présente en fait quelques particularités dues au problème concret à résoudre.

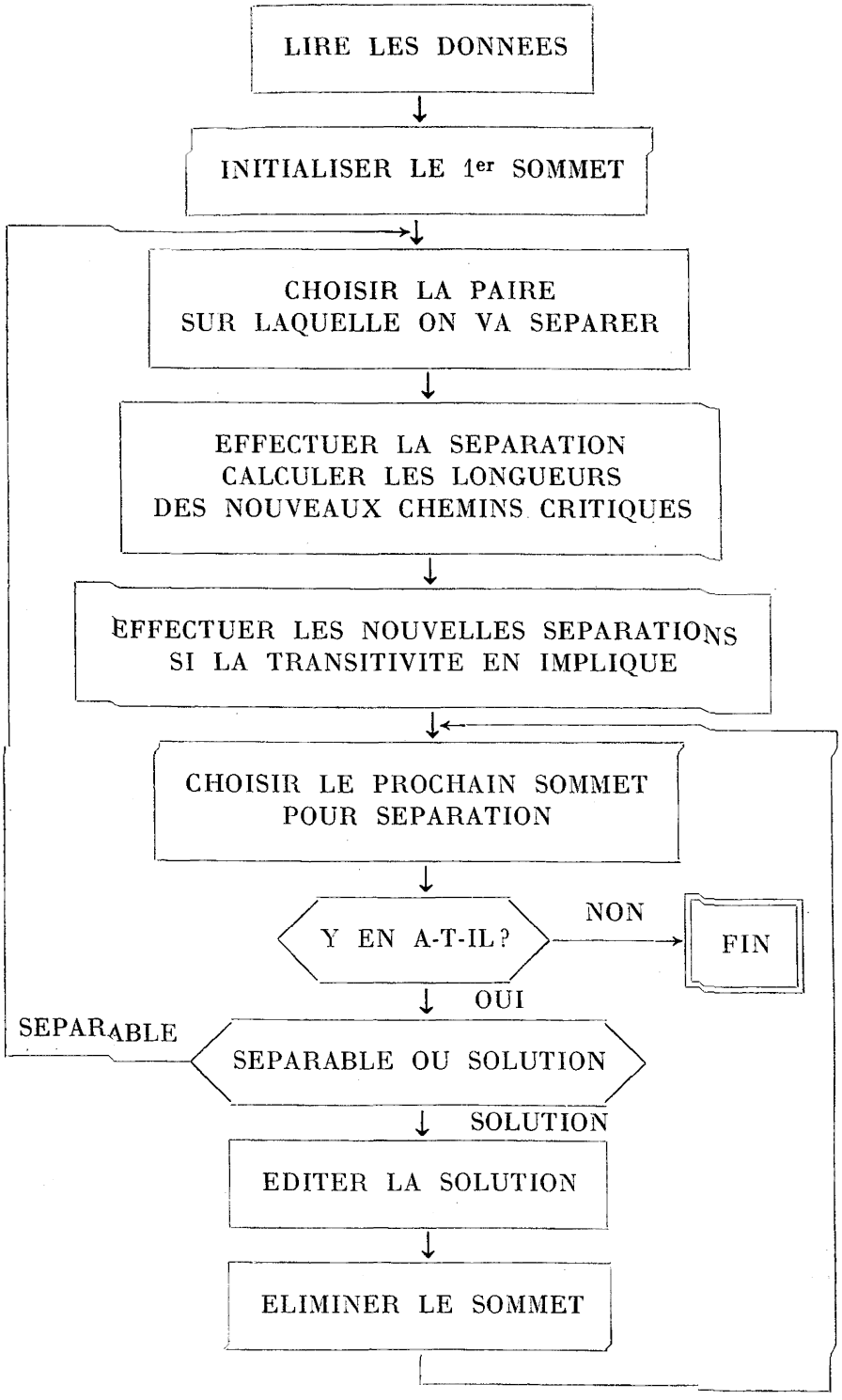
1. PARTICULARITES

Les délais sont supposés égaux à la durée des tâches ($a_{ij} = d_i$), ce qui est moins général. D'autre part, on ne se contente pas d'une seule solution optimale ; lorsque le problème est purement disjonctif et symétrique ($a_{ij} = d_i$), il y en a souvent un grand nombre, et différentes raisons concrètes nous poussaient à les vouloir *toutes*. Le procédé est le suivant : dès que l'on a trouvé la première solution optimale, de valeur V, on sait que maintenant tout sommet susceptible de fournir une autre variante optimale à une évaluation égale à V, car V est un minorant des évaluations des sommets pendants, et un majorant des évaluations des sommets capables de fournir l'optimal. On continue donc la procédure de séparation pour les sommets d'évaluation V ; lorsqu'il n'y en a plus, toutes les solutions optimales ont été trouvées : l'algorithme est terminé. Nous avons aussi inclus le calcul de deux critères secondaires dans le programme d'édition des solutions : ces critères sont destinés à sélectionner des solutions favorables à la synchronisation des carrefours le long de l'axe routier.

Enfin, le programme tient compte de ce que, lorsque $a_{ij} = d_i$, toute solution est réversible. Le premier sommet n'est donc pas la famille d'arbitrages vide, mais contient un arbitrage quelconque sur la paire de disjonction de plus grande durée totale.

2. STRUCTURE DE TADSEP

La structure indiquée ci-contre concerne le cas où l'on veut toutes les variantes optimales.


LIRE LES DONNEES
INITIALISER LE 1er SOMMET
CHOISIR LA PAIRE
SUR LAQUELLE ON VA SEPARER
EFFECTUER LA SEPARATION
CALCULER LES LONGUEURS
DES NOUVEAUX CHEMINS CRITIQUES
EFFECTUER LES NOUVELLES SEPARATIONS
SI LA TRANSITIVITE EN IMPLIQUE
CHOISIR LE PROCHAIN SOMMET
POUR SEPARATION
Y EN A-T-IL?
NON
FIN
OUI
SEPARABLE
SEPARABLE OU SOLUTION
SOLUTION
EDITER LA SOLUTION
ELIMINER LE SOMMET

3. QUELQUES RESULTATS

Nous avons essayé trois règles-S heuristiques pour un exemple avec :

10 tâches

20 contraintes disjonctives,

qui possédait 241 solutions optimales doubles (doubles, car le problème était symétrique). La taille de l'arborescence était limitée à 2 000 sommets.

Heuristique A : fondée sur les cliques maximales du graphe d'incompatibilité.

Le programme s'est arrêté sur la limite des 2 000 sommets. On a obtenu seulement 188 ($\times$ 2) solutions optimales ; le programme a demandé 21 secondes d'Unité Centrale de CDC 6600, y compris le calcul des sous-critères.

Heuristique B : mélange de *A* et *C*. Toutes les solutions obtenues en 9 secondes d'Unité Centrale, 918 sommets construits.

Heuristique C (« Δ_k »)

Toutes les solutions obtenues en 7 secondes d'Unité Centrale, 743 sommets construits.

CONCLUSION ET REMARQUES

L'outil tel qu'il est reste expérimental. Son originalité par rapport aux autres algorithmes « Branch and Bound » existants (par exemple [1], [2], [3]) est de progresser de solutions très incomplètes vers une solution optimale complète, au lieu de procéder à une énumération implicite de solutions complètes. Il nous paraît maintenant souhaitable de clore cet article par deux remarques importantes.

1) Une caractéristique du problème semble être le grand nombre de solutions optimales. A un certain niveau de complexité, on perd beaucoup de temps à les chercher toutes, sans savoir lesquelles seront meilleures vis-à-vis de certains critères secondaires. Dans certains cas favorables, il est possible de calculer pour eux des *évaluations secondaires*, qui jouent un rôle dans la sélection du sommet à séparer.

2) Le problème général, qui contient à la fois des contraintes conjonctives et disjonctives, peut être traité de la même façon, les contraintes conjonctives pouvant être interprétées comme des contraintes disjonctives sur lesquelles un choix a été fait. D'après quelques essais, il semble que l'heuristique Δ_k reste intéressante pour ce problème. Par contre, il apparaît très utile de ne pas arbitrer toutes les paires de disjonctions, mais d'introduire une notion de *conflit* est, due à B. Roy : un conflit est, pour un sommet donné de l'arborescence, une paire de disjonction

non arbitrée, telle que les débuts au plus tôt des deux tâches ne satisfont aucune des deux inégalités de la contrainte disjonctive. Il faut dire d'autre part que le problème général, très peu symétrique, possède un bien plus petit nombre de solutions optimales, et souvent une seule.

BIBLIOGRAPHIE

[1] E. Balas, « Finding a minimaximal path in a disjunctive PERT network », *Théorie des graphes*. Journées nationales d'études, Rome, juillet 1966.

[2] E. Balas. « Machine sequencing via disjunctive graphs : an implicit enumeration algorithm », *Management Science research report n° 125*, Carnegie-Mellon University, Pittsburgh, February 1968 ; revised on december 1968.

[3] J. F. Raimond. « An algorithm for the exact solution of the machine scheduling problem », IBM, New York Scientific Center report n° 320-2930, February 1968.

[4] B. Roy et B. Sussmann. « Les problèmes d'ordonnancement avec contraintes disjonctives », SEMA, Rapport de recherches n° 9, 1964.

[5] B. Roy, Nghiem P. T. et P. Bertier. « Programmes linéaires en nombres entiers et procédure SEP ». METRA, vol. IV, n° 3, 1965.

[6] B. Roy. « Prise en compte de contraintes disjonctives dans les méthodes de chemin critique », *Revue française de recherche opérationnelle*, n° 38, 10e année, 1er trimestre 1966.

[7] K. E. Stoffers. « Scheduling of traffic lights : a new approach ». *Transportation Research*, vol. II, n° 3, september 1968.

[8] H. Wagner. « An integer programming model for machine scheduling », *Naval Research Logistics quarterly*, 6, 1959, n° 2.