Le Problème du voyageur de commerce à coûts variables et quelques applications

Mauriceima Queyranne

THESE

présentée à

Université Scientifique et Médicale de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
en Mathématiques Appliquées
option : Recherche Opérationnelle

par

Maurice QUEYRANNE

LE PROBLEME DU VOYAGEUR DE COMMERCE A COUTS VARIABLES ET QUELQUES APPLICATIONS.

Thèse soutenue le 27 juin 1977 devant la Commission d'Examen :

Président : C. BENZAKEN

Examinateurs : H. RAYNAUD
M. SAKAROVITCH

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

MEMBRES DU CORPS ENSEIGNant DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BELORIZKY Elie	Physique
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZEZ Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONTVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMTIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM.	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	GAGNAIRE Didier	Chimie Physique
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie Pharmaceutique
	LAURENT Pierre	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (IUT I)
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences Nucléaires
	LONGEQUEUE Jean-Pierre	Physique Nucléaire
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALINAS Yves	Clinique Obstétricale
	MARTIN-NOEL Pierre	Clinique Cardiologique
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique Maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie Nucléaire
	NOZIERES Philippe	Spectrometrie Physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Semeiologie Médicale (Neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (IUT I)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale

MM.	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	DEMBICKI Eugéniuz	Mécanique
	JOHNSON Thomas	Mathématiques appliquées
	PENNEY Thomas	Physique

PROFESSEURS SANS CHAIRE

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT I)
	BUISSON René	Physique (IUT I)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques Pures
	JULLIEN Pierre	Mathématiques Appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques Appliquées
	KUHN Gérard	Physique (IUT I)
	LUU DUC Cuong	Chimie organique
	MAYNARD Roger	Physique du solide
Mme	MINIER Colette	Physique (IUT I)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mle	PIERY Yvette	Physiologie animale

MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (IUT I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B) (Personne étrangère habilitée à être directeur de thèse)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculair.
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JUNIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail

MM.	MARECHAL Jean	Mécanique (IUT I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (IUT I)
	NEGRE Robert	Mécanique (IUT I)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (IUT I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B) (Personnalité étrangère habilitée à être directeur de thèse)
	PEFFEN René	Métallurgie (IUT I)
	PERRIER Guy	Géophysique-Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine Interne
	RACINET Claude	Gynécologie et Obstétrique
	RAMBAUD André	Hygiène et Hydrologie (Pharmacie)
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme	RENAUDET JAcqueline	Bactériologie (Pharmacie)
MM	ROBERT Jean-Bernard	Chimie Physique
	ROMIER Guy	Mathématiques (IUT B) (Personnalité étrangère habilité à être directeur de thèse)
	SCHAERER René	Cancérologie
	SHOM Jean-Claude	Chimie Générale
	STOEBNER Pierre	Anatomie Pathologie
	VROUSOS Constantin	Radiologie

MAITRESDE CONFERENCESASSOCIES

MM.	DEVINE Roderick	Spectro Physique
	HODGES Christopher	Transition de Phases

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1976

TABLE DES MATIERES

REMERCIEMENTS

Je voudrais d'abord d'abord remercier Monsieur Claude BENZAKEN d'avoir accepté de diriger ce jury. Je remercie Monsieur Michel SAKAROVITCH qui a dirigé ce travail, malgré la distance et ses nombreuses autres charges, et fait également partie de ce jury ainsi que Monsieur Hervé RAYNAUD. Ce dernier a lu avec attention une version préliminaire de cette thèse, et je le remercie pour ses nombreux commentaires et suggestions.

Je voudrais également remercier l'ensemble du Département de Génie Industriel de l'Ecole Polytechnique de Montréal, ainsi que l'administration de cette école, pour m'avoir procuré les conditions pour réaliser ce travail. Ces recherches ont été en partie subventionnées par le Conseil National de la Recherche du Canada, subventions RD-804 et A-8528.

Monsieur Jean-Claude PICARD a droit à une place particulière dans l'expression de ma gratitude. Sans son aide , ses encouragements et aussi son expérience et son ingéniosité, ce travail n'aurait sans doute jamais vu le jour; qu'il en soit ici remercié. Je remercie aussi Monsieur Jean-Claude NADEAU pour son aide dans la mise au point de certains des programmes testés ici.

Enfin, je dédie ce travail à Marie-Pascale, qui mérite pourtant bien mieux (mais ceci, c'est une autre histoire...).

INTRODUCTION

Si le coût du trajet entre deux villes peut dépendre du jour où a lieu ce trajet, un "voyageur de commerce" qui désire, en n jours, visiter au moindre coût n villes données doit, pour déterminer sa tournée, résoudre un "problème du voyageur de commerce à coûts variables"; dans la suite, nous désignerons ce problème par ses initiales anglaises TDTSP. Le classique problème du voyageur de commerce et le problème d'affectation (linéaire) sont deux cas particuliers du TDTSP. Ce modèle peut être appliqué à la formalisation de problèmes de tournées, mais aussi de problèmes d'ordonnancement et de localisation et, plus généralement, peut fournir une relaxation pour une large classe de problèmes de permutation optimale. Deux de ces applications, l'une au problème du voyageur de commerce et l'autre à un problème d'ordonnancement sur une machine seront développées.

Les coûts de trajet associés au TDTSP définissent un réseau multiparti à n périodes (représentant les jours), chaque période contenant une image de chacune des n villes; ce réseau est complété par une origine (période 0) et une extrèmité (période n+1) et par les arcs représentant les divers trajets. Dans ce réseau, un plus court chemin de l'origine à l'extrèmité peut être obtenu par des algorithmes classiques dont l'application aux réseaux multipartis est discutée en détail; ces algorithmes joueront un rôle essentiel dans la suite. En général, un tel plus court chemin visite plusieurs images de certaines villes et oublie d'autres villes. Il est possible d'interdire de retourner à une ville visitée pendant les k périodes

suivantes et des algorithmes sont décrits pour cela; cependant la complexité de ces algorithmes est une fonction exponentielle de k,ce qui rend cette approche du TDTSP impraticable même pour des petites valeurs de k. Un algorithme par programmation dynamique est également décrit pour le TDTSP, mais il nécessite un tel volume de mémoire qu'il devient inapplicable à partir de 15 villes. Le TDTSP peut être également résolu par une méthode heuristique, réalisant des améliorations par des échanges ou des insertions, mais cette méthode a l'inconvénient de ne pas garantir l'optimalité de la solution obtenue.

Le TDTSP peut être formulé comme un programme en nombres entiers; trois telles formulations sont décrites. Des relaxations de ces formulations sont également introduites et les relations entre ces relaxations sont étudiées; ces relaxations joueront un rôle important dans la méthode de solution proposée pour le TDTSP. L'introduction de pénalités dans le réseau multiparti permet de résoudre les plus fines de ces relaxations; la méthode de solution consiste à trouver un plus court chemin en tenant compte des pénalités, puis à modifier les pénalités d'après le chemin obtenu. Cette modification des pénalités peut être réalisée par l'algorithme du simplexe, ou, de façon plus efficace, par une méthode d'optimisation par sous-gradient. Il peut être nécessaire d'incorporer cette approche dans un algorithme d'énumération implicite (Branch-and-Bound) lorsqu'on ne peut obtenir de solution réalisable, à cause du "saut primal-dual" causé par la non-convexité du problème.

Cette méthode de solution peut être appliquée au classique problème du voyageur de commerce; ceci rejoint d'ailleurs diverses propositions antérieures pour ce problème. Il est montré que la relaxation utilisée est plus fine que la relaxation classique par le problème d'affectation linéaire. Toutefois le temps passé dans les calculs de plus courts chemins est trop important et des résul-

tats d'expériences numériques montrent que cette méthode échoue pour des problèmes à 40 villes, que d'autres méthodes peuvent résoudre convenablement.

La méthode de solution proposée pour le TDTSP s'avère beaucoup plus efficace lorsqu'elle est appliquée à un problème d'ordonnancement sur une machine. Dans ce problème, n tâches doivent être exécutées, une à la fois, de façon à minimiser la somme des coûts de retard encourrus par ces n tâches. Le TDTSP est alors introduit comme une relaxation pour ce problème d'ordonnancement, et certaines réductions permettent de simplifier le réseau multiparti associé. Cette relaxation est intégrée dans un algorithme par Branch-and-Bound pour ce problème d'ordonnancement, qui, testé sur 84 ensembles de données, s'est révélé satisfaisant: il permet de résoudre en un temps de calcul raisonnable des cas que ne pouvaient résoudre des algorithmes précédemment publiés.

Cette approche utilisant le TDTSP comme une relaxation peut être étendue à d'autres problèmes d'optimisation combinatoire incluant le problème d'affectation quadratique, le problème de rangement linéaire et d'autres problèmes de tournées et d'ordonnancement.

C H A P I T R E I : Problèmes de plus courts chemins dans les réseaux multipartis; le problème du voyageur de commerce à coûts variables.

-1 Réseau multiparti

Un réseau multiparti est un réseau $R = (N,A,c)$ orienté dont l'ensemble N des noeuds est partitionné en k sous-ensembles $N_0, N_1, \ldots, N_{k-1}$ (mutuellement disjoints et exhaustifs), appelés périodes, tels que tout arc dans A ayant son origine dans une période N_t ait son extrèmité dans N_{t+1}. Cette définition suppose implicitement que $k \geq 2$; si $k = 1$, l'ensemble A est vide. Un réseau multiparti tel que $k = 2$ s'appelle biparti ; lorsque $k = |N|$ et qu'il est connexe, un tel réseau est réduit à un chemin élémentaire. Nous appellerons transition t l'ensemble des arcs joignant les périodes N_t et N_{t+1}.

Nous allons nous intéresser aux réseaux multipartis du point de vue du cheminement. Pour cela nous poserons $m = k-2$ et nous considèrerons que N_0 et N_{m+1} sont réduits chacun à un seul noeud, que nous appellerons respectivement origine n_0 et extrèmité n_{m+1} du réseau. Sans perdre de généralité, nous supposerons en outre que chaque période N_t peut être identifiée à un sous-ensemble d'un ensemble fini X. En particulier, on pourra considérer qu'un élément x_i de X peut être représenté par différents noeuds dans des périodes distinctes. Ainsi les noeuds intermédiaires (c'est-à-dire distincts de n_0 et de n_{m+1}) pourront être notés n_{it}, le deuxième indice t indiquant le numéro de la période. Puisque trois indices i,j,t suffisent alors à définir un arc $(n_{i\,t}, n_{j\,t+1})$ nous noterons la longueur d'un tel arc $c^t_{i\,j}$.

Pour la plupart des applications développées ici (sauf au § 1-4), on considérera que $N_t = X$ pour tout $t = 1,2,\ldots,m$, avec $|X| = m$. Donc $|N| = m^2 + 2$. Nous appelerons réseau multiparti complet un réseau de ce type dans lequel tous les arcs possibles existent, sauf les arcs "diagonaux" $(n_{i\,t}, n_{i\,t+1})$; dans ce cas, $|A| = 2m + m(m-1)^2$.

Remarque: en associant à un réseau multiparti R quelconque un nombre

$$m = \text{Max}\left\{|N_1|, |N_2|, \ldots, |N_{k-2}|, k-2\right\}$$

il est clair qu'un réseau multiparti complet est celui qui contient le plus de noeuds et le plus d'arcs (avec $|N_0| = |N_{k-1}| = 1$) pour m fixé.

2 Problème 1: Plus court chemin de n_0 à n_{m+1} .

Un réseau multiparti ne comportant pas de circuit, on peut appliquer un algorithme dû à Bellman [1958] pour trouver un plus court chemin de n_0 à n_{m+1} . Avant de décrire l'application d'un tel algorithme aux réseaux multipartis, nous introduirons quelques conventions destinées à alléger les notations.

Pour éviter des tests décidant si un noeud a déjà été atteint depuis l'origine, ou bien des manipulations de listes de noeuds déjà marqués, nous introduirons des marques $+\infty$, qui en pratique pourront être remplacées par un nombre M assez grand (par exemple $M \geq (m+1)\text{Max}\{c_{ij}^t\}$). Dans ces conditions, on pourra tirer parti du fait que tous les sommets d'une période sont (définitivement) marqués avant qu'on ne commence le marquage pour la période suivante.

En représentant par f(j,t) la distance de n_0 à $n_{j\ t}$ (la longueur d'un plus court chemin de n_0 à $n_{j\ t}$), le "principe d'optimalité " de Bellman peut s'énoncer

$$f(j,t) = \text{Min}\{f(i,t-1)+c_{i\ j}^{t-1}\} = f(m(j,t)\ ,\ t)+c_{m(j,t)\ j}^{t-1}$$

où $m(j,t)$ indique un noeud ayant permis de définir $f(j,t)$ et $f(0,0) = 0$ par définition. En "pseudo-algol" (cf. Aho et al. [1974]), cet algorithme peut s'énoncer comme suit :

Algorithme PCC:

```
(1)  pour j=1 jusqu'à |N1| faire
        si (n0 ,nj 1) A alors f(j,1) ← c0 j sinon f(j,1) ← +∞ ;
(2)  pour t=2 jusqu'à m faire
        début pour j=1 jusqu'à Nt faire f(j,t) ← +∞
              pour tous les i tels que (ni t-1 ,nj t)∈A faire
              début si f(i,t-1)+c(t-1) i j < f(j,t) alors
                    début f(j,t) ← f(i,t-1) + c(t-1) i j
                          m(j,t) ← i
        fin   fin   fin ;
```

(3) $\ell \leftarrow +\infty$;

pour tous les i tels que $(n_{i\,m}, n_{m+1}) \in A$ faire
début si $f(i,m)+c^m_{i\,m+1} < \ell$ alors
début $\ell \leftarrow f(i,m)+c^m_{i\,m+1}$
$ch(m) \leftarrow i$
fin fin.

Dans cet algorithme, l'ordre dams lequel les arcs de la transition t_{-1} sont utilisés est non-spécifié. Si les arcs sont regroupés par leur extrèmité on obtient un algorithme du type "marquage permanent" (cf[*]). Si, au contraire, les arcs sont regroupés suivant leur origine, on obtient un algorithme du type "correction de marquage" (ibid.), Pour ces différentes versions de l'algorithme, le nombre d'opérations élémentaires est essentiellement constant; la différence, en général négligeable, dépend surtout de la structure de données utilisée pour représenter le réseau et de la version en langage-machine de l'algorithme.

Le nombre d'opérations élémentaires requises pour l'exécution de cet algorithme est essentiellement proportionnel au nombre d'arcs, à moins que celui-ci ne soit inférieur au nombre de noeuds. Dans ce cas certains noeuds ne sont pas accessibles depuis l'origine.

Dans le cas particulier, très important dans la suite , où $|X_t| = m$ pour tout $t = 1,2,\ldots,m$ et où tous les arcs possibles existent, alors la complexité de l'algorithme est $O(m^3)$.

Dans le cas où l'on sait a priori qu'un certain nombre de noeuds ne sont pas accessibles depuis l'origine, il peut être plus efficace d'utiliser une liste des noeuds effectivement marqués, au lieu des marques fictives $+\infty$. (cf. chap. 4 pour une application).

[*] Gilsinn et Witzgall, [1973] .

Voir également l'annexe A1-1 pour la description d'un autre algorithme pour le problème 1, l'algorithme PCCH.

1-3 Problème 2: Plus court chemin sans oscillation de n_0 à n_{m+1}

Soit un chemin $n_0 - n_{j_1} - n_{j_2} - \cdots - n_{j_t} - \cdots - n_{j_m} - n_{m+1}$,que l'on notera simplement $0-j_1-j_2-\cdots-j_t-\cdots-j_m-m+1$ dans toute la suite. On dira qu'un tel chemin est <u>sans oscillation</u> si, pour tout $t=3,4,\ldots,m$

$$j_t \neq j_{t-1} \quad \text{et} \quad j_t \neq j_{t-2}$$

(et bien sûr $j_1 \neq j_2$).

Remarquons que la première contrainte peut être satisfaite en supprimant simplement les arcs "diagonaux" $(n_{j\,t}, n_{j\,t+1})$.

Désignons par $C_{1,m}$ l'ensemble des chemins de l'origine à l'exrèmité, et par $C_{2,m}$ l'ensemble des chemins sans oscillation de l'origine à l'extrèmité. Evidemment $C_{2,m} \subset C_{1,m}$.Si le réseau est complet,

$$|C_{1,m}| = m(m-1)^{m-1} \qquad |C_{2,m}| = m(m-1)(m-2)^{m-2}$$

$$\text{et} \quad \frac{|C_{1,m}|}{|C_{2,m}|} \xrightarrow[m\to\infty]{} e \quad (=2.718\ldots)$$

<u>Lemme 1:</u> Si $0-j_1-j_2-\cdots-j_{t-2}-j_{t-1}-j_t-\cdots-j_m-m+1$ est un plus court chemin sans oscillation de n_0 à n_{m+1} ,alors, pour tout $t=3,4,\ldots m$,le chemin $0-j_1-j_2-\cdots-j_{t-2}-j_{t-1}$ est un chemin sans oscillation de n_0 à $n_{j_{t-1}}$ de longueur minimale parmi ceux dont le $(t-2)^{\text{ème}}$ noeud est distinct de j_t .

<u>Preuve :</u> supposons que $0-j'_1-j'_2-\cdots-j'_{t-2}-j_{t-1}$ soit un chemin sans oscillation de n_0 à $n_{j_{t-1}}$ de longueur strictement plus petite que le précédent, et avec $j'_{t-2} = j_t$;alors le chemin $0-j'_1-j'_2-\cdots-j'_{t-2}-j_{t-1}-j_t-\cdots-j_m-m+1$ est un chemin sans oscillation de n_0 à n_{m+1} de longueur strictement plus petite que celui supposé optimal, d'où la contradiction. #

Le lemmme 1 permet de déduire une extension de l'algorithme période-par-période pour trouver un plus court chemin sans oscillation de l'origine à tout noeud du réseau. Dans cet algorithme, on conserve pour chaque noeud (j,t) la longueur du p.c.c.s.o. (plus court chemin sans oscillation) de l'origine à ce noeud ainsi qu'une marque m((j,t) indiquant le noeud (m(j,t),t-1) précédent sur ce chemin, comme dans l'algorithme PCC. Mais on conserve en plus la longueur f'(j,t) du p.c.c.s.o. de l'origine à (j,t) n'utilisant pas le noeud (m(j,t),t-1), et la marque m'(j,t) correspondante.

Si l'on définit

$$T(i,j) = \begin{cases} f(i,t-1) + c_{i\,j}^{t-1} & \text{si } m(i,t-1) \neq j \\ f'(i,t-1) + c_{i\,j}^{t-1} & \text{sinon,} \end{cases}$$

alors les relations de récurrence déduites du lemme 1 sont

$$f(j,t) = \min\{T(i,j); i \neq j\} = T(m(j,t),j)$$

$$\text{et } f'(j,t) = \min\{T(i,j); i \neq j \text{ et } i \neq m(j,t)\} = T(m'(j,t),j).$$

Nous allons maintenant donner une description de l'algorithme PCC2 pour trouver un plus court chemin sans oscillation de l'origine à l'extrèmité dans un réseau multiparti. Pour simplifier cette description, on suppose que le réseau est complet, au sens qui a été défini plus haut.

Algorithme PCC2 :

(1) pour i = 1 jusqu'à m faire

début $f(i,2) \leftarrow \min\left\{c^0_{0j} + c^1_{ji} \ ; \ j \neq i\right\}$

$m(i,2) \leftarrow k$ tel que $c^0_{0k} + c^1_{ki} = f(i,2)$

$f'(i,2) \leftarrow \min\left\{c^0_{0j} + c^1_{ji} \ ; \ j \neq i \text{ et } j \neq m(i,2)\right\}$

$m'(i,2) \leftarrow k$ tel que $c^0_{0k} + c^1_{ki} = f'(i,2)$ et $k \neq m(i,2)$

fin

(2) pour t = 3 jusqu'à m-1 faire

début pour j = 1 jusqu'à m faire

début pour i = 1 jusqu'à m faire $T(i,j) \leftarrow f(j,t-1) + c^{t-1}_{j\,i}$

$T(m(j,t-1),j) \leftarrow f'(j,t-1) + c^{t-1}_{j\,m(j,t-1)}$

fin

pour i = 1 jusqu'à m faire

début $f(i,t) \leftarrow \min\left\{T(i,j) \ ; j \neq i\right\}$

$m(i,t) \leftarrow k$ tel que $T(i,k) = f(i,t)$

$f'(i,t) \leftarrow \min\left\{T(i,j) \ ; j \neq i \text{ et } j \neq m(i,t)\right\}$

$m'(i,t) \leftarrow k$ tel que $T(i,k) = f'(i,t)$ et $k \neq m(i,t)$

fin fin

(3) pour j = 1 jusqu'à m faire

début pour i = 1 jusqu'à m faire $T(i,j) \leftarrow f(j,m-1) + c^{m-1}_{j\,i}$

$T(m(j,m-1),j) \leftarrow f'(j,m-1) + c^{m-1}_{j\,m(j,m-1)}$

fin

pour i = 1 jusqu'à m faire

début $b(i) \leftarrow c^m_{i\,m+1} + \min\left\{T(i,j) \ ; j \neq i\right\}$

$m(i,m) \leftarrow k$ tel que $T(i,k) = \min\left\{T(i,j) \ ; \ j \neq i\right\}$

fin

(4) $\ell \leftarrow \min\left\{b(i)\right\}$

$ch(m) \leftarrow k$ tel que $b(k) =$

terminer

Après exécution des algorithmes PCC ou PCCH , on pouvait retrouver un plus court chemin par "retour arrière" en utilisant simplement les marques m ; ici, il faut utiliser les marques m et m' :

Retour arrière : $ch(m-1) \leftarrow m(ch(m))$

pour $t = m-2$ pas -1 jusqu'à 1 faire

si $m(ch(t+1),t+1) = ch(t+2)$ alors $ch(t) \leftarrow m'(ch(t+1),t+1)$

sinon $ch(t) \leftarrow m(ch(t+1),t+1)$

terminer.

Avant d'analyser en détail le nombre d'opérations élémentaires effectuées durant l'execution de l'algorithme PCC2, nous allons préciser les raisons pour lesquelles certaines solutions ont été adoptées dans sa conception.

Afin d'éviter les tests j=i ? on utilise les arcs "diagonaux" auxquels un longueur $c_{ii}^{t} \leftarrow +\infty$ est affectée.

On aurait pu se passer du tableau T en utilisant des instructions du type :

$$f(i,t) \leftarrow \min\left\{f(j,t-1) + c_{j\,i}^{t-1} \;;\; j=i \text{ et } m(j,t-1)=i\right\} \cup \left\{f'(j,t-1) + c_{j\,i}^{t-1} \;;\; j=i \text{ et } m(j,t-1)=i\right\}$$

mais on devrait, dans ce cas, effectuer des tests $m(j,t-1)=i$? pour chaque couple (i,j) et chaque t. Dans la solution adoptée, on définit d'abord le tableau T ligne par ligne, puis on corrige les valeurs $T(m(j,t-1),j)$. On introduit ainsi, pour chaque période t, m additions supplémentaires, mais on économise m^2 comparaisons.

Une discussion détaillée de la méthode (algorithme MIN2) utilisée pour trouver les deux plus petits éléments de chaque colonne du tableau T est donnée dans l'annexe A1-2. Il y est montré que le nombre de comparaisons nécessaires pour trouver les deux plus petits éléments d'un ensemble à m éléments est, en moyenne (sous l'hypothèse habituelle que l'ordre dans lequel sont rangés ces m éléments est aléatoire) $m + 1.387 \log_2 m - 2$ avec cet algorithme MIN2 . Nous allons utiliser ce résultat pour analyser l'algorithme PCC2.

Dans le cas où R est un réseau multiparti complet de taille m , nous allons effectuer le compte des opérations (additions , comparaisons) pour chaque pas de l'algorithme PCC2 :

(1) (t = 2) pour chaque i — m additions

$m + 1.387 \log_2 m - 2$ comparaisons

(2) pour t = 3 jusqu'à m - 1

$m (m + 1)$ additions

pour chaque i , $m + 1.387 \log_2 m - 2$ comparaisons

(3) (t = m) — $m (m + 1)$ additions

pour chaque i, — m comparaisons

et enfin ,

(4) $(t = m+1)$ m comparaisons

En tout il faut donc $m^3 + m$ additions (exactement)

et $m^2(m-1)+m+m(m-1)(1.387 \log_2 m - 2)$ comparaisons en moyenne (au pire ce dernier facteur est égal à $(m-3)$). Il est intéressant de comparer ces nombres à ceux que l'on peut obtenir pour l'algorithme PCC dans le cas complet, soit $m^3 - m^2$ additions

et $m^2(m-1)+m$ comparaisons.

On voit donc que le travail supplémentaire pour éviter les oscillations consiste en m^2+m additions et $m(m-1)(1.387 \cdot \log_2 m - 2)$ comparaisons en moyenne. Le tableau II donne la proportion d'opérations supplémentaires requises pour PCC2 par rapport à PCC, pour les additions (nombre exact) et les comparaisons (en moyenne) dans le cas complet.

m	10	20	30	40	50
additions	10.9 %	5.2 %	3.4 %	2.6 %	2.0 %
comparaisons	25.8 %	19.9 %	16.0 %	13.4 %	11.6 %

TABLEAU II : Proportion d'opérations supplémentaires dans PCC2 par rapport à PCC.

On voit donc sur ce tableau que le travail supplémentaire pour éviter les oscillations est environ 10 à 20 % du travail initial, pour des

valeurs de m comprises entre 10 et 50, ce qui représente le domaine des applications actuellement étudiées.

On peut également généraliser l'algorithme PCCH pour éviter les oscillations, en utilisant les mêmes relations de récurrence. Nous ne donnerons pas ici la description de cet algorithme, car elle se déduit assez facilement de la précédente. Les conditions d'application d'un tel algorithme sont soumises aux mêmes contraintes que pour PCCH.

-4 Problème k : Plus courts chemins sans k-circuits.

On peut généraliser l'idée du paragraphe précédent de la manière suivante: un chemin $0-j_1-j_2-\ldots-j_m-(m+1)$ est sans k-circuit si, pour tous s,t ($s,t = 1,\ldots,n$ et $s \neq t$)

$$|s - t| \leq k \quad \text{implique} \quad j_s \neq j_t \ .$$

Evidemment un chemin sans 2-circuit est un chemin sans oscillation et réciproquement. Si l'on désigne par $C_{k,m}$ l'ensemble des chemins sans k-circuit dans un réseau multiparti complet de taille m, alors

$$C_{m-1,m} \subset C_{m-2,m} \subset \ldots \subset C_{3,m} \subset C_{2,m} \subset C_{1,m}$$

$$\text{et} \quad |C_{k,m}| = m(m-1)(m-2)\ldots(m-k+1)(m-k)^{m-k}$$

Alors

$$\frac{|C_{k,m}|}{|C_{k+1,m}|} \xrightarrow[m \to \infty]{} e = 2.718\ldots$$

$$\frac{|C_{1,m}|}{|C_{k,m}|} \xrightarrow[m \to \infty]{} e^k$$

Pour trouver un p.c.c. sans k-circuit, l'algorithme PCC2 ne semble pas pouvoir s'étendre d'une manière simple pour $k \geq 3$. Nous proposons ici une autre approche. En posant :

$$d = \left\lceil \frac{m}{k} \right\rceil \qquad \text{et} \qquad r = m - k(d-1)$$

on peut associer à R un autre réseau multiparti R_k à d+2 périodes. La période 0 contient l'origine, la période 1 contient $m(m-1)\ldots(m-r+1)$ noeuds, représentant chacune des séquences de r indices, parmi $\{1,2,\ldots,m\}$, distincts; les d-1 périodes suivantes contiennent chacune $m(m-1)\ldots(m-k+1)$ séquences de k indices distincts et la dernière contient l'extrèmité, notée encore n_{m+1}. Dans ce réseau on définit les arcs :

$(0,(j_1,j_2,\ldots,j_r)_1)$ ayant pour longueur $c^0_{0j_1} + c^1_{j_1j_2} + \ldots + c^{r-1}_{j_{r-1}j_r}$

$((j_1,j_2,\ldots,j_r)_1,(j'_1,j'_2,\ldots,j'_k)_2)$ — - ——— $c^r_{j_rj'_1} + c^{r+1}_{j'_1j'_2} + \ldots + c^{r+k-1}_{j'_{k-1}j'_k}$

si $j_b \neq j'_c$ ∀ b, ∀ c ≤ b

. . . . . .

$((j_1,j_2,\ldots,j_k)_a,(j'_1,j'_2,\ldots,j'_k)_{a+1})$ — - — $c^{r+(a-1)k}_{j_kj'_1} + c^{r+(a-1)k+1}_{j'_1j'_2} + \ldots + c^{r+ak-1}_{j'_{k-1}j'_k}$

si $j_b \neq j'_c$ ∀ b, ∀ c ≤ b

. . . . . .

$((j_1,j_2,\ldots,j_k)_d,m+1)$ — — ——— $c^m_{j_k\,m+1}$

Lemme 1-1 :

Il existe une correspondance bijective entre chemins (de l'origine à l'extrèmité) dans R_k et chemins sans k-circuits dans R; de plus, deux chemins correspondants ont même longueur.

preuve : immédiate, par construction. #

D'après ce lemme, on peut trouver un p.c.c. sans k-circuit en appliquant un algorithme de p.c.c. dans le réseau multiparti R_k. La complexité d'un tel algorithme sera $O(m^{2k+1})$. Remarquons que, pour k=2 le nombre

d'opérations élémentaires s'exprime par un polynome dont le terme de plus haut degré est $\frac{1}{2}m^5$, contre $2m^3$ (au pire) pour l'algorithme PCC2.

5 Le problème du voyageur de commerce à coûts variables :

Dans un réseau multiparti R tel que $|X_1| = |X_2| = \ldots = |X_m| = m$, c'est résoudre le problème du voyageur de commerce à coûts variables que de trouver un plus court chemin sans (m-1)-circuit. En d'autres termes, il s'agit de trouver une permutation w de $\{1, 2, \ldots, m\}$ qui minimise le coût total :

$$C(w) = C^0_{0,w(1)} + C^1_{w(1),w(2)} + \ldots + C^{m-1}_{w(m-1),w(m)} + C^m_{w(m),m+1}$$

On peut illustrer ce problème de plusieurs façons :

(i) considérons un voyageur de commerce (ou bien un candidat à une élection) qui doit arriver chaque jour dans une des m villes qu'il doit visiter; la fonction économique (baptisée ici "coût" , mais ce peut être aussi la durée de voyage, la fatigue,...) associée à l'ensemble de ses déplacements s'exprime comme la somme des coûts associés à chaque trajet; ces coûts dépendent du jour où le trajet a été effectué : par exemple, certains vols n'ont lieu que certains jours, ou bien un long trajet est plus fatiguant à la fin d'une campagne électorale,etc... ;

(ii) une fabrique de peintures produit cinq couleurs différentes, une par jour de la semaine; la machine doit être nettoyée chaque soir et cette opération est effectuée par une équipe de nuit qui travaille d'abord à d'autres tâches dans l'usine; ces autres tâches ont un horaire hebdomadaire fixe, ce qui laisse à l'équipe de nuit un certain temps disponible qui peut être différent pour différentes nuits de la semaine; à cause de ces temps disponibles différents, des facteurs de production tels la

main-d'oeuvre (heures supplémentaires ou main d'oeuvre additionnelle) ou des produits chimiques (solvents, détergents) peuvent être utilisés à différents niveaux; ainsi le coût de nettoyage dépend non seulement de la couleur enlevée et de celle à installer (problème du voyageur de commerce, voir Conway et al. [1967]), mais aussi de la nuit pendant laquelle chaque transition a eu lieu.

Ainsi, et bien que son appellation suggère un problème de tournées, le problème du voyageur de commerce à coûts variables (que nous désignerons par TDTSP, "Time-Dependent Traveling Salesman Problem" dans la thèse de K.Fox [1973]) peut donc aussi bien servir comme modèle pour certains problèmes d'ordonnancement.

Deux cas particuliers du TDTSP sont importants car ils ont déjà beaucoup été étudiés :

1. lorsque les coûts ne sont pas variables, c'est-à-dire

$$C^t_{i\,j} = C_{i\,j} \qquad \text{pour tout } t \text{ (et tous } i \text{ et } j)$$

on obtient le problème du voyageur de commerce, auquel sera consacré le Chapitre III.

2. lorsque les coûts ne dépendent pas de l'extrèmité de l'arc, soit

$$C^t_{i\,j} = C^t_i \qquad \text{pour tous } i,j,t,$$

on obtient le <u>problème d'affectation</u> (linéaire); rappelons que ce problème, dont une formulation sera donnée au paragraphe 2-2, consiste à chercher une affectation des m villes aux m périodes, exactement une ville i par période t, de façon à minimiser la somme des coûts d'affectation C^t_i . Remarquons en outre que, à cause de la symétrie dans la définition du TDTSP, on obtient également un problème d'affectation lorsque les coûts ne dépendent pas de l'origine de l'arc.

Il est possible d'incorporer certaines contraintes supplémentaires dans la formulation même du problème. Ainsi, et bien que celle-ci suppose un noeud origine et un noeud extrèmité, on peut modéliser des problèmes pour lesquels l'un ou l'autre n'existe pas, en plaçant un coût nul sur les arcs correspondants. De même, on peut interdire de placer deux éléments i et j dans deux périodes successives en définissant $C^t_{i\,j} = +\infty$ (et/ou $C^t_{j\,i} = +\infty$) pour tous les t considérés. On a déjà vu que l'on pouvait également interdire la présence d'un élément i dans certaines périodes $t \in T_i$ en supprimant (ou, ce qui revient au même, en leur plaçant un coût infini) les arcs incidents aux noeuds $n_{i\,t}$, $t \in T_i$.

Désignons par z_k la longueur d'un p.c.c. sans k-circuit dans R; alors z_{m-1} désigne la valeur (longueur) d'un solution optimale du TDTSP sur R. Grace aux inclusions successives des ensembles C_k , on a

$$z_1 \leqslant z_2 \leqslant z_3 \leqslant \cdots \leqslant z_k \leqslant \cdots \leqslant z_{m-2} \leqslant z_{m-1}$$

Les problèmes de plus courts chemins sans k-circuit forment donc une famille de relaxations de plus en plus fines pour le TDTSP. En contrepartie le coût de calcul pour leur solution est en $O(m^{k+2})$ et croît avec k. Remarquons que ,pour k=m-1, on trouve approximativement l'ordre de complexité de la "méthode brutale" qui consiste à évaluer chacune des m! permutations possibles. D'autre part, il semble difficile de prouver que les algorithmes proposés pour les problèmes de p.c.c. sans k-circuit sont, au moins en ordre de complexité, optimaux; d'ailleurs ce n'est certainement pas le cas pour k = 2 puisque PCC2 est meilleur. Mais cette complexité apparemment croissante semble fournir de bonnes raisons de croire que la réponse à la question "P = NP ?" (voir Karp [1972]) doit être négative.

On est donc amené à rechercher des méthodes de solution pour le TDTSP dont les performances ne sont pas garanties, mais qui pourraient néammoins être utilisables. A coté de la méthode brutale d'énumération exhaustive, on peut déjà imaginer une classe d'algorithmes par "Branch-and-Bound" (Séparation et Evaluation Progressives,cf Little et al. [1963], ou Enumération Implicite),définis de la manière suivante:

-évaluation: chercher un p.c.c. sans k-circuit dans R (k est un paramètre qui définit l'algorithme dans cette classe);

-séparation: si certains indices i apparaissent plusieurs fois dans le plus court chemin, essayer d'imposer un de ces indices à certaine période tout en l'interdisant aux autres périodes.

On obtient ainsi plusieurs TDTSP de taille m-1 que l'on peut essayer de résoudre par la même approche récursive, et la solution optimale sera la meilleure des solutions de ces sous-problèmes.

Remarque: si les problèmes de p.c.c. doivent être résolus pour fournir une borne pour un TDTSP, on peut simplifier le réseau R_k correspondant (défini au paragraphe précédent); ainsi, pour $k \geq 4$, des noeuds de R_k associés à un même ensemble $\{j_1, j_2, j_3, \ldots, j_{s-1}, j_s\}$ (où s peut être k ou r[§]), avec le même premier noeud j_1 et le même dernier noeud j_s, il suffit de ne retenir que celui correspondant à une séquence $j_1, j'_2, j'_3, \ldots, j'_{s-1}, j_s$ de longueur $c^t_{j_1 j'_2} + c^{t+1}_{j'_2 j'_3} + \ldots + c^{t+s-2}_{j'_{s-1} j_s}$ minimale; d'autre part, on peut également supprimer les arcs joignant deux séquences dès qu'elles ont un indice au moins en commun; ces simplifications réduisent le nombre d'éléments du réseau R_k, donc accélèrent l'exécution des algorithmes, et permettent d'obtenir une meilleure borne; on pourrait également obtenir une meilleure borne en appliquant à R_k l'algorithme PCC2 ou même un algorithme de p.c.c. sans ℓ-circuit ($\ell < d$).

A cette approche, nous préfererons une approche plus élaborée par laquelle les contraintes qui ne peuvent être représentées dans le réseau ("chaque indice doit apparaître exactement une fois") seront utilisées de manière indirecte pour tenter de limiter, ou au contraire de favoriser, la présence de certains indices dans le chemin que l'on obtiendra. Dans ce cas, l'évaluation sera obtenue après plusieurs itérations de p.c.c. dans un réseau identique à R, mais avec des coûts modifiés. Cette approche sera l'objet du chapitre 2. Auparavant, nous allons décrire rapidement deux autres approches, l'une exacte et l'autre approchée, pour le TDTSP.

[§] Rappelons que $r = m - k(\lceil \frac{m}{k} \rceil - 1)$ (cf. paragraphe 1-4).

1-6 Solution du TDTSP par la programmation dynamique

Une méthode de solution du TSP par la programmation dynamique a été proposée par Bellman, Gonzales et Held et Karp (voir Bellmore et Nemhauser [1968]) et peut être généralisée de manière immédiate au TDTSP. Soit $S \subseteq \{1,2,\ldots,m\}$ et $i \notin S$, désignons par $f(S,i)$ la valeur d'une solution optimale du TDTSP dans un réseau $R_{S,i}$ qui est déduit de R de la manière suivante: $R_{S,i}$ a $s+2$ périodes ($s=|S|$); les périodes $1,2,\ldots,s$ contiennent les noeuds (j,t) avec $j \in S$ et $t=1,2,\ldots,s$; la période 0 est identique à celle de R et la dernière période est réduite au noeud $(i,s+1)$; les arcs sont les mêmes que les arcs correspondants de R et ont même longueur. L'algorithme de résolution du TDTSP par la programmation dynamique est fondé sur la relation de récurrence :

$$f(S,i) = \min\{f(S-\{j\},j) + C^{s}_{ji} \quad : j \in S\}$$

avec les conditions initiales $f(\{j\},i) = C^{0}_{0j} + C^{1}_{ji}$ $(j \neq i)$. La valeur d'une solution optimale est donnée par $f(\{1,2,\ldots,m\},m+1)$ et la solution peut être obtenue par "retour arrière". On peut considérer cette approche comme la recherche d'un p.c.c. dans un réseau mutiparti à $m+2$ périodes, mais dans lequel chaque noeud (i,t) est "éclaté" en $\binom{m-1}{t-1}$ noeuds (S,i).

Pour l'application au TDTSP , la seule différence avec le TSP est que le coût C^{s}_{ji} apparaissant dans la relation de récurrence est variable avec S, ce qui n'entraîne qu'un accroissement négligeable du temps de calcul. Les limites de l'application de cette méthode de solution seront donc essentiellement les mêmes que pour le TSP: mémoire et temps croissent en $O(2^m)$;

ainsi une capacité de mémoire de 32K en limite l'application à $m \leq 15$ et, même si l'on utilise une mémoire auxiliaire, des problèmes seulement à peine plus grands pourront être résolus en un temps raisonnable.

Deux récentes contributions pourraient conduire à réviser ce jugement. L'emploi de méthodes de Branch-and-Bound au sein même d'un algorithme de programmation dynamique (Marsten et Morin [1976]) permet de réduire dans une proportion considérable le nombre de solutions à évaluer. L'utilisation de certaines relations de précédence dans la structure de la solution optimale permet aussi d'accélerer remarquablement l'énumération ("méthode des chaînes de K.R.Baker [1974], [1975]), voir le chapître 4. Enfin, pour certains problèmes très difficiles,ce peut être la seule méthode disponible pour une solution exacte.

-7 Echanges et insertions pour le TDTSP

La méthode heuristique que nous allons décrire brièvement peut être, elle aussi, considérée comme l'extension d'une méthode déjà appliquée au TSP (Reiter et Sherman [1965]). C'est une méthode du type "recherche dans un voisinage" ("neighborhood search", Garfinkel et Nemhauser [1972]), et elle peut être décrite comme suit:

(1) prendre une solution réalisable de départ X,

(2) chercher une meilleure solution dans un voisinage de X,

(3) s'il en existe , remplacer X par une meilleure solution et retourner en (2),sinon retourner en (1) choisir une autre solution de départ si l'on désire un nouvel essai.

Une méthode particulière, dans cette classe, est définie par la manière dont sont effectués les différents pas de l'algorithme.

Au pas (1) on peut choisir X de façon aléatoire (voir chapitre 3) ou bien par certains procédés constructifs (voir chapitre 4).

Au pas (2), il faut définir le voisinage d'une solution X. Nous allons considérer trois procédés qui permettent d'engendrer des voisinages qui sont à chaque fois plus vastes:

(i) échange d'éléments adjacents: si la solution considérée est représentée par la séquence

$$S = 0\text{-}j_1\text{-}j_2\text{-}\cdots\text{-}j_{k-1}\text{-}j_k\text{-}j_{k+1}\text{-}j_{k+2}\text{-}\cdots\text{-}j_m\text{-}(m+1)$$

le voisinage de celle-ci est défini par l'ensemble des séquences qui peuvent s'en déduire par l'echange de deux éléments adjacents j_k et j_{k+1}, c'est-à-dire les séquences du type $S_k = 0\text{-}j_1\text{-}j_2\text{-}\cdots\text{-}j_{k-1}\text{-}j_{k+1}\text{-}j_k\text{-}j_{k+2}\text{-}\cdots\text{-}j_m\text{-}(m+1)$; alors

$$C(S_k) = C(S) - (C^{k-1}_{j_{k-1}j_k} + C^{k}_{j_k j_{k+1}} + C^{k+1}_{j_{k+1}j_{k+2}}) + (C^{k-1}_{j_{k-1}j_{k+1}} + C^{k}_{j_{k+1}j_k} + C^{k+1}_{j_k j_{k+2}}).$$

(ii) insertion d'un élément: le voisinage est constitué par l'ensemble des séquences qui peuvent se déduire de S par l'insertion d'un élément j_k entre deux autres éléments j_h et j_{h+1}, c'est-à-dire les séquences du type:

$$S_{k,h} = 0\text{-}j_1\text{-}\cdots\text{-}j_{k-1}\text{-}j_{k+1}\text{-}\cdots\text{-}j_h\text{-}j_k\text{-}j_{h+1}\text{-}\cdots\text{-}j_m\text{-}(m+1)$$

$(k,h = 1,2,\ldots,m$ et $k \neq h)$; la différence entre $C(S_{k,h})$ et $C(S)$ peut être calculée en remarquant que $S_{k,h}$ peut être obtenue à partir de S par des échanges successifs d'éléments adjacents.

(iii) insertion d'un segment: dans ce cas on insère entre j_h et $j_{h\ 1}$ un segment constitué de r éléments consécutifs dans S, soit $j_k\text{-}j_{k+1}\text{-}\cdots\text{-}j_{k+r-1}$ $(r = 1,\ldots,m-1$; $k \leq m-r+1$; $h < k-1$ ou $h > k\ r)$; une telle solution, que l'on notera $S_{k,h,r}$ peut s'obtenir à partir de S par des insertions successives

des éléments du segment à leur place ultérieure (r insertions).

Remarque: dans le cas du TSP symétrique, Reiter et Sherman définissent un voisinage un peu plus vaste dans le cas (iii) en considérant l'insertion du segment dans l'un ou l'autre sens, permettant d'obtenir des séquences du type:

$$S'_{k,h,r}=0\text{-}j_1\text{-}\cdots\text{-}j_{k-1}\text{-}j_{k+r}\text{-}\cdots\text{-}j_h\text{-}j_{k+r-1}\text{-}\cdots\text{-}j_k\text{-}j_{h+1}\text{-}\cdots\text{-}j_m\text{-}(m+1$$

bien que ce type de séquences puisse être également obtenu en r insertions d'un seul élément, nous ne le considèrerons pas ici car il n'y a en général, pour les problèmes que nous allons considérer, aucune symétrie dans les coûts ($C^t_{i\,j}$ et $C^t_{j\,i}$ sont indépendants).

Nous dirons qu'une solution est r-localement optimale si $C(S)\leqslant C(S_{k,h,r'})$ pour tout $r'\leqslant r$ et tous les k,h correspondants. Remarquons qu'une solution même m-localement optimale n'est pas nécessairement optimale.

Lemme 1-2 Si S est $\lfloor \frac{1}{2}m \rfloor$-localement optimale, alors elle est m-localement optimale.

preuve: toute insertion d'un segment à r éléments est équivalente à l'insertion d'un autre segment à s éléments, tel que r+s= m. En effet, si k<h, on a

$$S_{k,h,r} = S_{h+1,k+r-1,k-h-1}$$

et si k>h,

$$S_{k,h,r} = S_{k+r,h-1,h-k-r+1}.$$

On peut donc se restreindre à $r\leqslant \frac{1}{2}m$.

Enfin, au pas (3), la décision de terminer peut être prise suivant l'un, ou plusieurs des critères suivants:

(i') on peut prouver que la solution obtenue est optimale, par exemple en comparant avec une borne inférieure sur la valeur de la solution optimale;

(ii') le nombre d'essais est fixé à l'avance;

(iii') un argument statistique "a priori" permet d'inférer que la probabilité d'améliorer sur la solution obtenue est trop faible, ou bien que le coût de calcul est trop grand par rapport à l'amélioration possible (Reiter et Sherman [1965].

Nous verrons au chapitre 4 qu'une méthode heuristique construite sur ces principes peut être très satisfaisante, alors qu'au chapitre 3, une telle méthode est beaucoup moins efficace.

A N N E X E S A U C H A P I T R E 1.

A1-1 L'algorithme PCCH :

On peut considérer que, puisque l'on cherche seulement un plus court chemin de l'origine à l'extrèmité, on a effectué un travail excessif en marquant tous les noeuds intermédiaires : l'idéal serait de n'avoir marqué que les noeuds sur un plus court chemin de l'origine à l'extrèmité. Nous allons discuter de l'application d'un algorithme par correction de marquage dans lequel une marque provisoire peut être rendue définitive avant que tous les arcs ayant ce noeud pour extrèmité n'aient été utilisés.

Le prototype de ce genre d'algorithmes est celui de Dijkstra [1959]. La remarque suivante permet son application dans le cas d'un réseau multiparti : tout chemin de l'origine à l'extrèmité contient exactement m + 1 arcs, un dans chaque transition. On ne changera donc pas les longueurs relatives de ces chemins en ajoutant une même constante K_t à tous les $c^t_{i\,j}$ et ceci pour tous les $t = 0,1,\ldots,m$. En posant

$$K_t = -\min\left\{c^t_{i\,j} ; (n_{i\,t-1}, n_{j\,t}) \in A\right\}$$

on obtient un réseau à longueurs nonnégatives pour lequel l'algorithme de Dijkstra est valide. Remarquons que chaque période contient au moins un arc de longueur nulle, ce qui permet de prévoir que certains arcs ne seront sans doute pas utilisés. Remarquons aussi que tous les arcs devront être examinés pour définir les différents K_t ,à moins que d'autres informations ne soient disponibles permettant d'éviter l'examen de tous les arcs.

Une généralisation de l'idée précédente consiste à utiliser une fonction compatible, suivant une proposition de G.Nemhauser [1972]. Une fonction $h : N \to \mathbb{R}$ sera dite <u>compatible avec c</u> si et seulement si

$$h(i,t) \leq h(j,t+1) + c^t_{i\,j} \quad \text{pour tous } i,j,t.$$

Puisqu'un réseau multiparti ne contient pas de circuit, il existe des fonctions compatibles avec c : par exemple la fonction h(j,t) définie par la longueur

d'un plus court chemin de (j,t) à l'extrèmité. Un autre exemple est

$$h(j,t) = -\sum_{\tau \geq t} K_\tau$$

qui correspond à l'algorithme de Dijkstra.

Algorithme PCCH :

(1) $L \leftarrow \emptyset$; $D \leftarrow \emptyset$;

pour j = 1 jusqu'à N_1 faire

si $(n_0, n_{j\,1}) \in A$ alors

début $f(j,1) \leftarrow c^0_{0\,j}$

$L \leftarrow L \cup \{(j,1)\}$

fin ;

(2) si $L = \emptyset$ alors terminer sinon

début soit $(j^*, t^*) \in L$ tel que

$$f(j^*,t^*) + h(j^*,t^*) = \min\{f(j,t) + h(j,t); (j,t) \in L\}$$

fin ;

(3) si $t^* = m + 1$ alors aller en (4) sinon

début $D \leftarrow D \cup \{(j^*,t^*)\}$

pour tout j tel que

$(j, t^*+1) \in D$ et $(n_{j^*\,t^*}, n_{j\,t^*+1}) \in A$

faire début si $((j,t^*+1) \notin L)$

ou $(f(j,t^*+1) > f(j^*,t^*) + c^{t^*}_{j^*\,j})$ alors

début $f(j,t^*+1) \leftarrow f(j^*,t^*) + c^{t^*}_{j^*\,j}$

$m(j,t^*+1) \leftarrow j^*$

fin ; fin ;

$L \leftarrow L - \{(j^*,t^*)\}$; aller en (2)

fin ;

(4) $\ell \leftarrow f(m+1,m+1)$;

$ch(m) \leftarrow m(m+1,m+1)$;

terminer.

Cet algorithme est la simple application de l'algorithme de Nemhauser au réseau multiparti. S'il termine au pas (2), il n'existe pas de chemin de l'origine à l'extrèmité.

Le nombre d'opérations requises dans l'exécution de cet algorithme dépend essentiellement de deux facteurs :

(i) le nombre de noeuds dans D à la fin de l'exécution

(ii) la structure de données utilisée pour conserver L.

A chaque passage à l'étape (3), D acquiert exactement un élément supplémentaire. Soit donc d le nombre d'éléments dans D à la fin de l'exécution : évidemment $m \leq d \leq |X|$. De plus à tout instant, lors de l'exécution de l'algorithme, on a $|L| \leq |X| - |D|$. Dans le cas régulier on a $m \leq d \leq m^2$ et $|L| \leq m^2$.

Pour simplifier la discussion de la structure de données à utiliser pour L, nous considérerons seulement le cas d'un réseau multiparti complet.

Si l'on ne tire pas parti de la structure multipartie du réseau, chaque exécution de la boucle constituée par les pas (2) et (3) peut requérir $O(|L|)$ ou bien $O(m \log L)$ opérations, soit pour calculer le minimum des marques dans L, soit pour modifier ou introduire m marques dans L au pas (3). Supposons alors que l'on connaisse pour chaque période X_t le minimum des marques provisoires. Une exécution du pas (2) requiert $O(m)$ comparaisons pour déterminer le "minimum minimorum", et $O(m)$ opérations pour recalculer la marque provisoire minimum dans la période t^*. L'exécution du pas (3) nécessite alors $O(m)$ additions, comparaisons et $O(m)$ comparaisons pour recalculer le minimum des marques provisoires dans X_{t^*+1}. Une boucle, formée des pas (2) et (3) nécessitera donc $O(m)$ opérations. En fait il est possible d'utiliser des structures de données plus efficaces afin d'éviter de répéter certaines comparaisons; par exemple si l'ensemble des minima des différentes périodes est conservé sous forme de pile (cf. Aho,... [1974]) le pas (3) nécessitera seulement $O(\log m)$ comparaisons.

En tirant ainsi parti de la structure du réseau, on obtient un algorithme consistant en d boucles comportant chacune O(m) opérations, soit une complexité de O(dm), et dans le cas extrème $O(m^3)$. Remarquons que le coefficient de proportionnalité est beaucoup plus grand que pour l'algorithme PCC. Un tel algorithme pourra donc être interessant aux conditions suivantes :

(i) d est"suffisamment petit" par rapport à m^2 et

(ii) une fonction compatible h est "facilement" accessible et satisfait (i).

A1-2 L' algorithme MIN2 :

La tâche à laquelle l'algorithme PCC2 consacre le plus de temps est la recherche des deux plus petits éléments de chaque colonne du tableau T. Nous allons discuter du choix d'une méthode pour cette phase de l'algorithme et nous serons amenés à préférer, pour cette application, une méthode très simple à une méthode "optimale".

Le nombre minimum de comparaisons nécessaires pour trouver les deux plus petits éléments d'un ensemble A à m éléments est $m + \lceil \log_2 m \rceil - 2$ (voir Knuth, [1973], pp. 209-212). Une classe d'algorithmes pour ce problème peut être définie ainsi: déterminer d'abord un plus petit élément en m-1 comparaisons, puis le second élément parmi ceux qui ont été éliminés par le vainqueur. Des algorithmes de cette classe sont optimaux pour le critère minimax (c'est-à-dire requièrent au plus $m + \log_2 m - 2$ comparaisons) si le plus petit élément peut être déterminé après au plus $\lceil \log_2 m \rceil$ victoires. Si, de plus, ce nombre de victoires remportées par le plus petit élément est exactement $\lceil \log_2 m \rceil$ ou bien $\lfloor \log_2 m \rfloor$, un tel algorithme minimise également le nombre moyen de comparaisons pour les algorithmes de cette classe, sous l'hypothese habituelle que les m! permutations dans lesquelles l'ensemble A peut être données sont équiprobables. Nous conjecturons qu'un tel algorithme, d'ailleurs assez facile à construire, est également optimal pour le critère nombre moyen de comparaisons parmi tous les algorithmes pour résoudre ce problème par comparaisons d'éléments de A deux à deux.

Toutefois, l'inconvénient de tels algorithmes pour la pratique est qu'ils nécessitent soit des opérations arithmétiques supplémentaires (par exemple des divisions par deux), soit des déplacements d'enregistrements , et que ce travail supplémentaire peut en dégrader notablement les performances. De plus leur codage est sans doute un peu long, ce qui les rend moins interessants s'il faut les incorporer à d'autres algorithmes.

L'algorithme suivant est très simple, et le nombre moyen de comparaisons qu'il requiert est très proche des valeurs minimales précédentes. Cet algorithme examine les éléments de A un par un, en conservant dans minl et min2 les deux plus petits éléments déjà examinés. L'élément suivant est alors comparé d'abord à min2, puis éventuellement à minl. Notons que si on le comparait d'abord à minl, le nombre moyen de comparaisons serait presque doublé.

Algorithme MIN2 :

```
(1)  si a(1)≤ a(2) alors
          début  m1←1
                 m2←2
          fin

     sinon début  m1←2
                  m2←1
          fin
(2)  pour i = 3 jusqu'à m faire si a(i)< m2 alors
          si a(i)< m1 alors début  m2←m1
                                   m1←i
                               fin
                          sinon  m2←i .
```

Lemme : Si l'ordre dans lequel les éléments de A sont rangés est aléatoire, alors l'algorithme MIN2 requiert en moyenne moins de

$$m + 1.387 \log_2 m - 2 \quad \text{comparaisons} \quad \text{pour } m \geq 3.$$

Preuve : le pas 1 requiert exactement 1 comparaison; au pas 2, la probabilité que $a(i) < m2$ est égale à la probabilité que $a(i)$ soit parmi les deux plus petits des i premiers éléments, soit $\frac{2}{i}$; on effectue donc en moyenne $1 + \sum_{i=3}^{m} (1 + \frac{2}{i})$ d'où le résultat. #

Le tableau suivant permet de comparer le nombre moyen de comparaisons effectuées par l'algorithme MIN2 aux valeurs minimales données auparavant.

nombre de comparaisons	m									
	5	10	15	20	25	30	35	40	45	50
max. opt.	6.	12.	17.	23.	28.	33.	39.	44.	49.	54.
moy. opt.	5.4	11.4	16.93	22.4	27.72	32.93	38.17	43.4	48.58	53.72
moy. MIN2	5.67	11.86	17.64	23.19	28.63	33.99	39.29	44.56	49.79	54.99

TABLEAU I

C H A P I T R E I I : Programmes en nombres entiers, relaxations, pénalités et autres artifices pour résoudre le TDTSP.

1 Trois formulations du TDTSP par la programmation en nombres entiers:

Dans sa thèse, Ken Fox donne cinq formulations du TDTSP comme programmes linéaires en nombres entiers, et une formulation comme un problème de flots avec gains§ en nombres entiers. En fait aucune de ces formulations n'a permis d'en déduire une méthode efficace de solution. L'algorithme de Branch-and-Bound décrit par K.Fox fut en fait incapable de résoudre un problème à dix périodes en douze minutes de temps de calcul. La méthode que nous présentons permet de résoudre des problèmes de cette taille en moins d'une seconde, en général, sur un calculateur de performances moyennes (IBM 360/50 ou 360/75).

L'approche que nous proposons repose sur trois formulations du TDTSP par des programmes (linéaires ou quadratiques) en variables 0-1. Plus précisément, on s'efforce de résoudre des relaxations d'un ou de plusieurs de ces programmes et une méthode d'énumération implicite sera éventuellement utilisée pour assurer la satisfaction des contraintes relaxées.

La première formulation que nous allons considérer est une généralisation d'une formulation du TSP donnée par Hadley 1964 dans laquelle interviennent des variables associées aux différentes périodes ("time-dependent variables"):

$$(ILP1)\quad \min \sum_{j=1}^{m} C_{0j}^{0}\, x_{0j}^{0} + \sum_{t=1}^{m-1} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} C_{ij}^{t}\, x_{ij}^{t} + \sum_{j=1}^{m} C_{j,m-1}^{m}\, x_{j,m}^{m}$$

$$\text{s.c.} \quad \sum_{j=1}^{m} x_{0j}^{0} = 1 \qquad (2\text{-}1\text{-}1)$$

$$\sum_{\substack{k=1 \\ k\neq j}}^{m} x_{jk}^{1} = x_{0j}^{0} \quad \forall\, j=1,2,\ldots,m \qquad (2\text{-}1\text{-}2)$$

$$\sum_{\substack{k=1 \\ k\neq j}}^{m} x_{jk}^{t+1} = \sum_{\substack{i=1 \\ i\neq j}}^{m} x_{ij}^{t} \quad \forall\, j=1,2,\ldots,m \quad \forall\, t=1,2,\ldots,m-2 \qquad (2\text{-}1\text{-}3)$$

§ cf. Roy [1970] .

$$x^{m}_{j,m+1} = \sum_{\substack{i=1 \\ i/j}}^{m} x^{m-1}_{ij} \qquad \forall\ j=1,2,\ldots,m \qquad (2\text{-}1\text{-}4)$$

$$x^{0}_{0j} + \sum_{\substack{i=1 \\ i\neq j}}^{m} \sum_{t=1}^{m-1} x^{t}_{ij} = 1 \qquad \forall j=1,2,\ldots,m \qquad (2\text{-}1\text{-}5)$$

$$\sum_{j=1}^{m} x^{m}_{j,m+1} = 1 \qquad (2\text{-}1\text{-}6)$$

$$\sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} x^{t}_{ij} = 1 \qquad \forall\ t=1,2,\ldots,m-1 \qquad (2\text{-}1\text{-}7)$$

$$x \geqslant 0 \qquad (2\text{-}1\text{-}8)$$

$$x \text{ entier} \qquad (2\text{-}1\text{-}9).$$

Dans cette formulation $x^{t}_{ij}=1$ si le voyageur de commerce va de la ville i à la ville j pendant la période t et 0 sinon (remarquons que chaque x^{t}_{ij} figure dans une somme qui doit être égale à 1, donc les contraintes impliquent que x ne peut prendre que des valeurs 0 ou 1). La contrainte (2-1-1) signifie que le voyageur doit partir le premier jour, les contraintes (2-1-2) qu'il doit quitter, le second jour, la ville qu'il vient de visiter; les contraintes (2-1-3),(2-1-4) et (2-1-6) expriment la même relation pour les jours suivants, jusqu'au dernier jour; puis la contrainte (2-1-5) exprime que chaque ville doit être visitée exactement une fois; enfin la contrainte (2-1-7) impose que chaque jour ait lieu exactement un trajet.

Lemme 2-1 (Houck et Vemuganti [1976])

Les contraintes (2-1-6) et (2-1-7) sont redondantes avec les contraintes (2-1-1)-(2-1-4).

preuve:

En additionnant sur j les contraintes (2-1-2) et (2-1-3) on obtient:

$$\sum_{j=1}^{m} \sum_{\substack{k=1 \\ k\neq j}}^{m} x_{jk}^{1} = \sum_{j=1}^{m} x_{0j}^{0} = 1 \quad \text{d'après (2-1-1),}$$

$$\sum_{j=1}^{m} \sum_{\substack{k=1 \\ k\neq j}}^{m} x_{jk}^{t+1} = \sum_{j=1}^{m} \sum_{\substack{k=1 \\ k\neq j}}^{m} x_{jk}^{t} = 1 \quad \text{par induction,}$$
$$\forall\ t=1,2,\ldots,m-1$$

donc les contraintes (2-1-7) sont vérifiées. Enfin, en additionnant les contraintes (2-1-4) sur j , on obtient:

$$\sum_{j=1}^{m} x_{j,m+1}^{m} = \sum_{j=1}^{m} \sum_{\substack{i=1 \\ i\neq j}}^{m} x_{ij}^{m-1} = 1 \quad \text{d'après (2-1-7)}$$

donc (2-1-6) est aussi vérifiée. □

Par conséquent les contraintes (2-1-6) et (2-1-7) sont redondantes aussi bien dans (ILP1) que dans sa relaxation continue (obtenue en relaxant la contrainte (2-1-9), cf. paragraphe suivant). Dans toute la suite, lorsque nous réfererons à (ILP1) il sera entendu que les contraintes (2-1-6) et (2-1-7) ont été supprimées.

La seconde formulation du TDTSP que nous allons considérer est sous forme d'un problème d'<u>affectation quadratique:</u>

$$(QAP)\quad \text{min} \quad \sum_{i\ 1}^{m} \sum_{j\ 1}^{m} \sum_{t\ 1}^{m} \sum_{s\ 1}^{m} a_{ijts}\, x_{it}\, x_{js} \qquad (2\text{-}1\text{-}10)$$

$$\text{s.c.} \quad \sum_{t\ 1}^{m} x_{it} = 1 \qquad \forall\ i \ \ 1,2,\ldots,m \qquad (2\text{-}1\text{-}11)$$

$$\sum_{i\ 1}^{m} x_{it} = 1 \qquad \forall\ t \ \ 1,2,\ldots,m \qquad (2\text{-}1\text{-}12)$$

$$x_{it} = 0 \text{ ou } 1 \qquad \forall\ i,t \ \ 1,2,\ldots,m \qquad (2\text{-}1\text{-}13).$$

Dans cette formulation, $x_{it}=1$ signifie que la ville i est visitée le jour t. La contrainte (2-1-11) exprime le fait que chaque ville doit être visitée exactement une fois, et la contrainte (2-1-12) qu'à chaque jour on visite exactement une ville. Les coûts associés aux trajets sont représentés dans la fonction-objectif (2-1-10) en posant:

$$a_{iill} = c^0_{0i} \qquad \text{pour tout } i=1,2,\ldots,m$$

$$a_{ijtt\,1} = c^t_{ij} \qquad \text{pour tous } i,j,t=1,2,\ldots,m \text{ et } i \neq j$$

$$a_{iimm} = c^m_{i,m\,1} \qquad \text{pour tout } i=1,2,\ldots,m$$

et tous les autres coefficients $a_{ijts}=0$.

La troisième formulation fait appel aux chemins (de l'origine à l'extrèmité) dans le réseau multiparti R du chapitre 1. Désignons par p un tel chemin et par P l'ensemble de tous ces chemins dans R. A chaque chemin $p \in P$ nous allons associer un <u>vecteur associé</u> $a^p \in R^m$, dont la i-ème composante représente le nombre de fois que le chemin p traverse la ville i. Puisqu'un chemin contient exactement m villes (pas nécessairement distinctes) alors

$$\sum_{i\ 1}^{m} a^p_i = m \qquad \text{pour tout } p \in P \qquad (2\text{-}1\text{-}14)$$

Dans la formulation suivante, à chaque chemin $p \in P$ est associée une variable x_p :

(ILP2)

$$\min \quad \sum_{p\ P} \ell_p x_p \qquad (2\text{-}1\text{-}15)$$

$$\text{s.c.} \quad \sum_{p\ P} a^p_i x_p \qquad \forall\ i\ \ 1,2,\ldots,m \qquad (2\text{-}1\text{-}16)$$

$$x_p \geqslant 0 \qquad \forall\ p \in P \qquad (2\text{-}1\text{-}17)$$

$$x \text{ entier} \qquad (2\text{-}1\text{-}18)$$

où ℓ_p représente la longueur du chemin p dans le réseau R.
Si l'on additionne sur i les contraintes (2-1-16), on obtient en utilisant (2-1-14) et en divisant par m :

$$\sum_{p \in P} x_p = 1 \qquad (2\text{-}1\text{-}19)$$

Il découle de (2-1-17)-(2-1-19) que, dans une solution à (ILP2), exactement une variable x_{p*} est égale à 1, et toutes les autres sont nulles. Les contraintes (2-1-16) deviennent alors:

$$a_i^{p*} = 1 \qquad \text{pour tout } i = 1,2,\ldots,m$$

c'est-à-dire que p* est un chemin qui visite exactement une fois chaque ville, et d'après (2-1-15) une solution optimale du TDTSP.

On peut également introduire une quatrième formulation du TDTSP , que l'on notera (ILP3), en remplaçant dans (ILP2) l'ensemble P par l'ensemble P' des chemins sans oscillation (voir chapitre 1, où cet ensemble était désigné par $C_{2,m}$).

·2 Relaxations:

Considérons un problème de programmation mathématique:

$$(MP)\begin{cases}\min & f(x)\\ \text{s.c.} & x\in X\end{cases}$$

Nous dirons, d'après Geoffrion [1974], qu'un problème

$$(MP')\begin{cases}\min & f'(x)\\ \text{s.c.} & x\in X'\end{cases}$$

est une relaxation de (MP) si et seulement si $X\subseteq X'$, et $f'(x)\leqslant f(x)$ pour tout $x\in X$.

On peut distinguer trois types de relaxations :

- type 1: $X=X'$ (T1)
- type 2: $f'(x)=f(x)$ pour tout $x\in X$ (T2)
- type 3: ni l'un ni l'autre. (T3)

Dans les chapitres suivants nous allons rencontrer des relaxations de ces trois types. La distinction est importante pour les règles d'élimination utilisées dans des algorithmes par Branch-and-Bound. Pour le type 1, chaque résolution d'une relaxation à un noeud du Branch-and-Bound fournit une solution réalisable, dont le coût exact a été sous-évalué; il est facile d'en calculer le coût exact, qui peut, d'une part améliorer la borne supérieure "actuelle" utilisée pour l'élimination, et, d'autre part permettre de "remonter" si les deux coûts sont égaux (ou si $\lceil f'(x)\rceil = f(x)$ lorsqu'on sait que f prend des valeurs entières). D'autre part, une relaxation de type 2 est toujours plus forte (par définition§) qu'une relaxation de type 3 et par conséquent permettra de résoudre le problème en faisant moins d'énumération. On est donc amené à considérer

§ Rappelons qu'une relaxation (MP") de (MP) est plus forte que (MP') lorsque

$$\min_{X''} f''(x) \geqslant \min_{X'} f'(x)$$

que des relaxations de type 1 ou 2 sont plus interessantes que de type 3, avec, lorsque le choix est possible, une certaine préference pour le type 1. Pourtant nous verrons au chapitre 4 qu'une relaxation de type 3 (utilisant le TDTSP) peut être nettement plus efficace qu'une relaxation de type 1.

Les relaxations continues (LP1),(LP2) et (LP3) se déduisent des formulations (ILP1),(ILP2) et (ILP3) en supprimant les contraintes d'intègrité ((2-1-9) et (2-1-18)). Désignons par z_{LP1} ,z_{LP2} et z_{LP3} les valeurs optimales des fonctions-objectif de ces programmes linéaires. Remarquons que ces trois relaxations sont de type 2.

Pour le problème d'affectation quadratique (QAP), considèrons la relaxation définie par le problème d'affectation linéaire suivant (Lawler [1963]) :

$$(LAR) \quad \min \quad \sum_{j=1}^{m} \sum_{s=1}^{m} b_{js}\, x_{js} \qquad (2\text{-}2\text{-}1)$$

$$\text{s.c.} \quad \sum_{s=1}^{m} x_{js} = 1 \qquad \forall \; j = 1,2,\ldots,m \qquad (2\text{-}2\text{-}2)$$

$$\sum_{j=1}^{m} x_{js} = 1 \qquad \forall \; s = 1,2,\ldots,m \qquad (2\text{-}2\text{-}3)$$

$$x_{js} \geq 0 \qquad \forall \; j,\, s = 1,2,\ldots,m \qquad (2\text{-}2\text{-}4)$$

où $b_{js} = a_{jjss} + U_{js}$, et U_{js} est la valeur optimale de la fonction -objectif du problème d'affectation linéaire

$$(LAR_{j,s}) \quad \min \quad \sum_{\substack{i=1 \\ i/j}}^{m} \sum_{\substack{t=1 \\ t/s}}^{m} a_{ijts}\, v_{it} \qquad (2\text{-}2\text{-}5)$$

$$\text{s.c.} \quad \sum_{\substack{t=1 \\ t \neq s}}^{m} v_{it} = 1 \qquad \forall \; i = 1,2,\ldots,m \text{ et } i \neq j \qquad (2\text{-}2\text{-}6)$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{m} v_{it} = 1 \qquad \forall\ t=1,2,\ldots,m \text{ et } t \neq s \quad (2\text{-}2\text{-}7)$$

$$v_{it} \geq 0 \qquad \forall\ i,t=1,2,\ldots,m\ , i\neq j \text{ et } t\neq s \quad (2\text{-}2\text{-}8)$$

Puisque les coefficients a_{ijts} sont nuls pour $t \neq s$ et s-1 dans la fomulation du TDTSP comme un problème d'affectation quadratique, le calcul de U_{js} est réduit dans ce cas à la sélection d'un indice $i \neq j$ tel que le noeud (i,s-1) soit le plus proche de (j,s) (pour les distances C) dans la période s-1. Pour s=1, $U_{j1} = 0$.

Désignons par z_{LAR} la valeur de l'affectation optimale dans le problème (LAR). Remarquons que (LAR) est une relaxation de type 1 pour le TDTSP. Le théorème suivant va montrer que (LAR) est en fait la moins forte des quatre relaxations du TDTSP.

THEOREME 2-1: Soit z^* le coût d'une séquence optimale dans le le TDTSP. Alors :

$$z_{LAR} \leq z_{LP1} = z_{LP2} \leq z_{LP3} \leq z^*$$

preuve: (i) $z_{LAR} \leq z_{LP1}$: nous allons montrer que toute solution realisable x de (LP1) induit une solution réalisable u de (LAR) de coût inférieur ou égal au coût de x dans (LP1).

Définissons $u_{j1} = x^0_{0j}$ pour tout j

$$u_{js} = \sum_{\substack{i=1 \\ i\neq j}}^{m} x^{s-1}_{ij} \qquad \text{pour tout } j \text{ et tout } s \geq 2$$

u satisfait la contrainte (2-2-2) d'après (2-1-5), et (2-2-3) d'après (2-1-7); de plus u est nonnégatif d'après (2-1-8) donc c'est une solution réalisable de (LAR).

D'après la définition des coefficients b_{js} on a:

$$b_{j1} = a_{jj11} = c^0_{0j} \qquad \text{pour tout } j$$

$$b_{js} = \min\left\{a_{i,j,s-1,s} : i \neq j\right\} = \min\left\{c^{s-1}_{ij} : i \neq j\right\}$$

pour tout j et tout s, $2 \leqslant s \leqslant m-$

$$b_{jm} = a_{jjmm} + \min\left\{a_{i,j,m-1,m} : i \neq j\right\}$$

$$= c^m_{j,m\ 1} + \min\left\{c^{m-1}_{i,j} : i \neq j\right\} \qquad \text{pour tout } j$$

d'où :

$$\sum_{j=1}^{m} \sum_{s=1}^{m} b_{js} u_{js} \leqslant \sum_{j=1}^{m} c^0_{0j} x^0_{0j} + \sum_{s=1}^{m} \sum_{j=1}^{m} \sum_{\substack{i=1 \\ i \neq j}}^{m} c^s_{ij} x^s_{ij} + \sum_{j=1}^{m} c^m_{j,m+1} x^m_{j,m+1}$$

puisque u est réalisable pour (LAR), on a

$$z_{LAR} \leqslant \sum_{j=1}^{m} \sum_{s=1}^{m} b_{js}\ u_{js}$$

et l'inégalité énoncée est prouvée en prenant pour x une solution optimale de (LP1).

(ii) $z_{LP1} = z_{LP2}$:ce résultat est une extension au TDTSP d'un résultat prouvé pour le TSP par Houck et Vemuganti [1976]. La démonstration que nous allons en donner est beaucoup plus simple. Considérons les contraintes (2-1-1)-(2-1-4) et (2-1-8) : elles caractérisent un flot réalisable d'une unité circulant dans le réseau multiparti R de l'origine à l'extrèmité. Les points extrèmes du polyèdre qu'elles définissent représentent donc les chemins (de l'origine à l'extrèmité)dans R. Si l'on applique le principe de décomposition de Dantzig et Wolfe au programme linéaire (LP1) (après en avoir supprimé les contraintes redondantes (2-1-6) et (2-1-7)), on obtient, en remarquant que le vecteur associé a^p s'introduit pour représenter les contraintes (2-1-5) :

$$(DW1)\quad \min \sum_{p\in P} \ell_p x_p$$

$$\text{s.c.} \sum_{p\in P} x_p = 1 \qquad (2\text{-}2\text{-}9)$$

$$\sum_{p\in P} a_i^p x_p = 1 \qquad \forall\, i = 1,2,\ldots,m \qquad (2\text{-}2\text{-}10)$$

$$x_p \geq 0 \qquad \forall\, p \in P \qquad (2\text{-}2\text{-}11).$$

Or, nous avons dèjà remarqué que, d'après (2-1-14), les contraintes (2-2-10) impliquent (2-2-9). Si l'on supprime donc la contrainte de convexité (2-2-9) qui est redondante, (DW1) se ramène exactement à (LP2).

(iii) $z_{LP2} \leq z_{LP3}$ vient de l'inclusion $P' \subseteq P$.

(iv) $z_{LP3} \leq z^*$ vient du fait que (LP3) est une relaxation du (TDTSP) (puisqu'une séquence est un chemin sans oscillation). □

Ce théorème a plusieurs conséquences:

- on peut obtenir la borne z_{LP1} en résolvant (LP2), et ce sera l'objet du reste de ce chapitre;
- la borne obtenue en résolvant une relaxation par un problème d'affectation (ce qui est très fréquent dans les applications) est moins forte que celle que l'on peut obtenir par le TDTSP ;
- enfin la borne obtenue en ne considérant que des chemins sans oscillation est la meilleure parmi celles considérées (en fait, on pourrait obtenir des bornes encore meilleures en ne considérant que des chemins sans k-circuit, mais à un coût de calcul qui semble prohibitif).

Pénalités:

Le problème (LP2) (ou (LP3) -en fait, tout ce qui sera dit dans le reste de ce chapitre peut s'appliquer aussi bien à (LP3)) a seulement m contraintes, mais un très grand nombre de variables, soit $m(m-1)^{m-1}$ dans le cas complet. Il n'est donc pas question d'écrire et de résoudre (LP2) de manière explicite, mais peut le résoudre par génération de colonnes.

Nous appelerons pénalités les variables duales $(u_i)_{i=1,\ldots,m}$ associées à une base du programme linéaire (LP2). La recherche d'une variable candidate, suivant le critère habituel du simplexe, revient à la recherche d'un chemin p qui minimise la quantité $\mathcal{C}^p - \sum_{i\ 1}^{m} u_i\ a_i^p$ .Ceci revient à pénaliser d'une quantité u_i la longueur d'un chemin chaque fois qu'il traverse un noeud représentant la ville i, et à chercher un plus court chemin pour ces longueurs pénalisées. Les algorithmes donnés dans le premier chapitre peuvent s'adapter très facilement pour tenir compte de pénalités associées aux noeuds. Ainsi, dans l'algorithme PCC, il suffit d'ajouter $-u_i$ aux longueurs $f(i,t)$[§] après que le minimum définissant ces f(i,t) ait été calculé . Ceci introduit environ m^2 additions supplémentaires, ce qui est négligeable pour un algorithme en $O(m^3)$. Nous ne développerons pas davantage ce point qui est assez immédiat.

Un certain nombre de problèmes de taille $m \leqslant 10$ ont été traités de cette manière. Pour des problèmes dans lesquels les coûts C_{ij}^t étaient des nombres pseudo-aléatoires, on obtenait assez souvent une séquence comme solution optimale; les autres problèmes étaient résolus en très peu de séparations -voir le paragraphe Branch-and-Bound. Par contre des problèmes plus

[§]-Rappelons que $f(i,t) = \min_j \left\{ f(j,t-1) + C_{ji}^t \right\}$ (cf. § 1-2).

"réalistes", en ce sens qu'ils résultaient d'applications (par exemple le TSP) et présentaient une certaines structures (régularité dans les coûts), nécessitaient davantage d'itérations, entre 40 et 80 itérations pour $m=10$, pour fournir une solution fractionnaire; une pathologie particulière aux TSP symétriques sera exposée au chapitre suivant. A partir de $m=15$, les itérations devenaient interminables ("tailing off", cf. Held et Karp [1970]) sans fournir aucun résultat utilisable.

Bien que l'algorithme par génération de colonnes ne soit pas très efficace pour calculer z_{LP2} , on peut néammoins s'en servir pour démontrer quelques résultats intéressants sur la structure et les propriétés de (LP2).

Lemme 2-2: (i) si deux pénalités u et u' se déduisent l'une de l'autre par une translation parallèle à la direction définie par le vecteur $\underline{1}$ (dont toutes les composantes sont l'unité)

$$u = u' + q\underline{1} \qquad (q \in \mathbb{R})$$

alors, pour tout chemin $p \in P$, on a :

$$\ell^p - \sum_{i=1}^{m} u_i a_i^p = \ell^p - \sum_{i=1}^{m} u'_i a_i^p - mq \qquad (2\text{-}3\text{-}1);$$

(ii) si deux chemins p et p' ont même vecteur associé $a^p = a^{p'}$ et si $\ell^p < \ell^{p'}$, alors pour toute pénalité u on a :

$$\ell^p - \sum_{i\ 1}^{m} u_i a_i^p < \ell^{p'} - \sum_{i\ 1}^{m} u_i a_i^{p'} \qquad (2\text{-}3\text{-}2).$$

preuve: (i) et (ii) se déduisent immédiatement de (2-1-14). □

Le résultat (i) implique que les chemins optimaux pour u sont aussi optimaux pour u', et réciproquement. Dans le cas de la génération de colonne, et pour tout autre algorithme engendrant des chemins à partir des pénalités, on peut restreindre l'ensemble P en ne conservant, pour chaque vecteur a^p qu'un seul chemin de longueur ℓ^p minimale, d'après le résultat (ii).

Appelons <u>séquence</u> tout chemin sans (m-1)-circuit.

En particulier, au vecteur $a^s = \underline{1}$ ne sera associée que la séquence optimale. Le théorème suivant précise la structure du polyèdre convexe défini par les contraintes de (LP2), sous l'hypothèse que P a été réduit comme on vient de l'indiquer.

THEOREME 2-2:

S'il existe une séquence s dans le réseau R, le polyhèdre convexe Q défini par les contraintes (2-1-16) et (2-1-17) contient exactement un point entier, qui en est d'ailleurs un point extrème, et le diamètre de Q est égal à 2.

preuve: considérons le point entier défini par $x_s = 1$ et $x_p = 0$ pour tout $p \in P$, $p \neq s$. Puisque ce point appartient à Q, Q est non vide; de plus ce point entier en est un point extrème, puisque, pour tout $p \in P$, $0 \leq x_p \leq 1$.
D'autre part on a déjà vu qu'en additionnant les contraintes (2-1-16) on obtient (2-1-19)

$$\sum_{p\ P} x_p = 1$$

donc tout point entier de Q est défini par exactement une composante non nulle $x_p = 1$; les contraintes (2-1-16) impliquent alors que p doit être une séquence, donc est identique à s.

Considérons maintenant un point extrème quelconque de Q et une base $B = (p_1, p_2, \ldots, p_m)$ associée. Si l'un des p_i est s alors la solution $x_s = 1$, $x_p = 0$ pour tout $p \in P, p \neq s$ est une solution de base, donc le point considéré est le point entier. Considérons donc un point extrème non entier et une base associée B; nous allons montrer que l'on peut passer de ce point au point entier en un seul pivotage, ce qui prouvera que le diamètre de Q est égal à 2. Soit x^B la solution de base associée à B : on a

$$x^B = B^{-1}.\underline{1}$$

donc la représentation de la colonne $a_s = \underline{1}$ dans la base B est justement x^B. Puisque $\sum_{i\ 1}^{m} x_i^B = 1$ et que $x^B \geqslant 0$, il existe (au moins) un indice j tel que $x_{p_j} > 0$; on peut alors faire entrer a^s dans la base en faisant sortir n'importe quel chemin p_j tel que $x^B_{p_j} > 0$, et l'on obtient alors une base qui correspond au point entier.

Padberg et Rao [1974] ont décrit une classe de polyèdres convexes associés à des problèmes d'optimisation combinatoire réputés difficiles et ayant un diamètre égal à 2. Le polyèdre Q associé à cette formulation du TDTSP possède donc lui aussi cette propriété.

Le dernier théorème concerne l'existence de pénalités permettant d'obtenir la séquence optimale comme plus court chemin. Une séquence peut être identifiée facilement par un test en O(m) (par exemple $\|a^p - \underline{1}\|^2 = 0$?) et il n'y a alors aucun intérêt à effectuer d'autres itérations puisque le TDTSP est alors résolu. Mais comment peut-on espérer obtenir de telles pénalités ?

THEOREME 2-3: Les deux conditions suivantes sont équivalentes :

(i) il existe des pénalités u* telles qu'une séquence s* soit un plus court chemin par rapport à u* ;

(ii) $z_{LP1} = z^*$.

preuve : (i) implique (ii) : d'après (i) on a

$$z^* - \sum_{i\ 1}^{m} u_i^* \leqslant \ell^p - \sum_{i\ 1}^{m} u_i^* a_i^p \quad \text{pour tout } p \in P;$$

posons $U = \sum_{i\ 1}^{m} u_i$ et définissons des pénalités u' par

$$u'_i = u^*_i + (z^* - U)/m$$

alors $\sum_{i\ 1}^{m} u'_i = z^*$

d'où $0 \leqslant \ell^p - \sum_{i\ 1}^{m} u'_i\ a_i^p$ pour tout $p \in P$,

c'est-à-dire que les pénalités u' sont duales-réalisables pour (LP2). On a alors un couple de solutions x' primale-réalisable ($x'_p = 0$ sauf $x'_s = 1$) et u' duale-réalisable pour (LP2) avec

$$\sum_{i\ 1}^{m} u'_i = z^* = \sum_{p\ P} \ell^p\ x'_p$$

donc ces solutions sont optimales et $z_{LP2} = z^*$.

(ii) implique (i) : si z_{LP2} z^* ,soit alors u* des variables duales optimales pour (LP2); on a

$$\sum_{i\ 1}^{m} u^*_i = z^*$$

et $0 \leqslant \ell^p - \sum_{i\ 1}^{m} u^*_i\ a_i^p$ pour tout $p \in P$

donc $z^* - \sum_{i\ 1}^{m} u^*_i \leqslant \ell^p - \sum_{i\ 1}^{m} u^*_i\ a_i^p$ pour tout $p \in P$.

□

Ce résultat est assez décevant : quelque soit la manière dont on s'y prenne pour définir ou modifier des pénalités, celles-ci ne pourront fournir une séquence optimale que si c'est en fait une solution optimale de (LP2). C'est une indication qu'il est inutile de consacrer trop d'efforts à essayer de calculer la valeur exacte de z_{LP2} ,et qu'une borne inférieure assez serrée sur cette valeur, obtenue plus facilement peut être préférable.

Pénalités optimales:

Puique (LP2) est difficile à résoudre et que, de plus, z_{LP2} est seulement une borne inférieure pour z*, il faudra combiner à cette approche d'autres techniques pour garantir que la solution de (LP2) soit entière. Nous adopterons l'énumération implicite par "Branch-and-Bound", puisque les autres techniques disponibles (plans sécants, cône asymptotique) semblent difficilement applicables ici.

Il est nécessaire d'avoir une borne inférieure sur z* pour pouvoir appliquer cette technique. Or, la valeur de la fonction-objectif de (LP2) pour une solution primale-réalisable n'est pas une borne inférieure valide tant que (LP2) n'a pas été résolu de façon optimale. Toutefois, une borne inférieure est disponible au cours des itérations du simplexe (Dantzig et al. [1954]): soit z la valeur d'une solution primale-réalisable et u les variables duales correspondantes, alors

$$\underline{z} = z - \min\left\{ \ell^p - \sum_{i\,1}^{m} u_i\, a_i^p \;:\; p \in P \right\} \qquad (2\text{-}4\text{-}1)$$

est une borne inférieure pour z_{LP2} ,donc a fortiori pour z*. Mais $\underline{z}$ est en général d'un comportement très erratique au cours des itérations et ne fournit pas une très bonne borne (voir plus loin).

Pour n'importe quelles pénalités u,on peut obtenir une borne inférieure valide pour z* en cherchant un plus court chemin par rapport à ces pénalités; en effet

$$\min\left\{ \ell^p + u \cdot a^p \;:\; p \in P \right\} \leqslant z^* - u \cdot \underline{1}$$

d'où

$$\min\left\{ \ell^p + u(a^p - \underline{1}) \;:\; p \in P \right\} \leqslant z^* \qquad (2\text{-}4\text{-}2)$$

(en fait, ceci fournit une justification simple pour (2-4-1)).

Désignons alors par w(u) cette borne inférieure:

$$w(u) = \min\{\ell^p + u(a^p - \underline{1}) : p \in P\} \qquad (2\text{-}4\text{-}3)$$

On est alors conduit à considérer le problème de rechercher des pénalités u* qui fournissent la plus grande borne inférieure w(u*) possible, soit le problème

(POP) $\left[\max \{w(u) : u \in R^m\}\right.$ où w est défini comme en (2-4-3)

soit $\left[\max_{u \in R^m} \min_{p \in P} (\ell^p + u(a^p - \underline{1}))\right.$ (2-4-4).

Remarque : (POP) peut se formuler comme le programme linéaire suivant

$$\left[\begin{array}{l} \max \quad W \\ \text{s.c.} \quad W - u(a^p - \underline{1}) \leqslant \ell^p \qquad \forall\, p \in P \\ \qquad\quad W \text{ et } u \text{ non-astreints.} \end{array}\right.$$

Le dual de ce programme linéaire est

$$(\text{DOP})\left[\begin{array}{ll} \min \sum_{p \in P} \ell^p x_p & \\ \text{s.c.} \sum_{p \in P} x_p = 1 & (2\text{-}4\text{-}5) \\ \sum_{p \in P} (a^p - \underline{1}) x_p = 0 & (2\text{-}4\text{-}6) \\ x \geqslant 0. & \end{array}\right.$$

En utilisant (2-4-5), les contraintes (2-4-6) peuvent s'écrire comme (2-1-16). Comme (2-4-5) est redondant avec les contraintes (2-1-16) (on l'a déjà vu au paragraphe 2-1), (DOP) apparaît donc comme équivalent à (LP2), c'est-à-dire que (POP) est une formulation duale de (LP2).

Pour un problème du type de (2-4-4), une méthode approchée très efficace, l'optimisation par sous-gradient a été développée par Held et Karp [1971], et étudiée et généralisée par plusieurs autres. Voir en particulier Mathematical Study 3, notamment l'article de Fisher et al. [1975], pour de nombreuses références. Cette approche peut s'appliquer ici en observant que, pour toute

pénalité $\bar{u}$, le vecteur $a^{\bar{p}}-\underline{1}$ correspondant à un chemin $\bar{p}$ tel que

$$\ell^{\bar{p}} + \bar{u}\cdot(a^{\bar{p}}-\underline{1}) = \min_{p\in P}\{\ell^{p} + \bar{u}\ (a^{p}-\underline{1})\} = w(\bar{u})$$

est une direction de sous-gradient pour w au point $\bar{u}$.

Remarque: d'après le lemme 2-2 (i), ce minimum peut s'obtenir en cherchant un plus court chemin pour les pénalités $\bar{u}$. En fait, on peut se restreindre aux pénalités u telles que

$$\sum_{i\ 1}^{m} u_i = 0 \qquad (2\text{-}4\text{-}7)$$

d'après cette même observation et, dans ce cas w peut être défini plus simplement par

$$w(u) = \min\{\ell^{p} + ua^{p} \ : \ p\in P\}$$ pour tout $u\in\mathbb{R}^m$ vérifiant (2-4-7).

Par suite, (POP) peut se formuler aussi

$$\max\left\{ \min_{p\in P} (\ell^{p}+ua^{p}) \ : \ u\in\mathbb{R}^m \text{ et } \sum_{i\ 1}^{m} u_i = 0\right\}.$$

Nous allons donner le principe général de l'optimisation par sous-gradient:

optimisation de w par sous-gradient :

(0) choisir u^0 et poser $k=0$;

(1) chercher un plus court chemin p^k pour les pénalités u^k;

(2) si p^k est une séquence, terminer (le TDTSP est résolu);

(3) si l'on ne désire pas d'autres itérations, aller en (4)
sinon remplacer k par k+1
déterminer u^k à l'aide de $(a^{p^k}-\underline{1})$ et des pénalités précédentes $u^{k-1}, u^{k-2}, \ldots, u^0$
aller en (1);

(4) une borne inférieure pour z* est fournie par la quantité
$\max\ w(u^j) : j\ 0,1,\ldots,k$.

Un algorithme particulier est défini par la façon dont les instructions (0) et (3) sont effectuées.

Pour l'instruction (0), remarquons que n'importe quelles pénalités u^0 ,par exemple $u^0 = \underline{0}$, peuvent faire l'affaire. On peut aussi déduire les pénalités initiales u^0 des variables duales optimales pour le problème d'affectation (LAR) :

THEOREME 2-4 : Soient $(u_j^*)_{j\ 1,\ldots,m}$ et $(v_s^*)_{s\ 1,\ldots,m}$ des variables duales optimales pour (LAR). Si l'on définit

$$u_j^0 = -u_j^* - \bar{v}^* \quad \text{pour tout } i = 1,2,\ldots,m \qquad (2\text{-}4\text{-}8)$$

où $$\bar{v}^* = \left(\sum_{s=1}^{m} v_s^*\right)/m \qquad (2\text{-}4\text{-}9),$$

alors $w(u^0) = z_{LAR}$.

preuve : considérons un chemin quelconque $p = 0\text{-}i_1\text{-}i_2\text{-}\ldots\text{-}i_m\text{-}(m+1)$;

d'après la définition des b_{js} (cf. § 2-2), on a :

$$\begin{aligned}
C^0_{0i_1} + u^0_{i_1} &= b_{i_1 1} - u^*_{i_1} - \bar{v}^* \\
C^1_{i_1 i_2} + u^0_{i_2} &\geq b_{i_2 2} - u^*_{i_2} - \bar{v}^* \\
&\cdots\cdots\cdots\cdots\cdots \\
C^{s-1}_{i_{s-1} i_s} + u^0_{i_s} &\geq b_{i_s s} - u^*_{i_s} - \bar{v}^* \\
&\cdots\cdots\cdots\cdots\cdots \\
C^m_{i_m m\ 1} + C^{m-1}_{i_{m-1} i_m} + u^0_{i_m} &\geq b_{i_m m} - u^*_{1_m} - \bar{v}^*
\end{aligned}$$

d'où $$p - \sum_{i=1}^{m} u_i a_i^p \geq \sum_{s=1}^{m} (b_{i_s s} - u^*_{i_s} - \bar{v}^*) = \sum_{s=1}^{m} (b_{i_s s} - u^*_{i_s} - v^*_s)$$

et, puisque les u*,v* sont duales-réalisables pour (LAR),

$$p + \sum_{i=1}^{m} u_i a_i^p \geq 0 \qquad \text{pour tout } p \in P.$$

Comme $$-\sum_{i=1}^{m} u_i^0 = \sum_{i=1}^{m} u_i^* + \sum_{s=1}^{m} v_s^* = z_{LAR}$$

on a $$w(u^0) = \min_{p\ P} \left(\ell^p + \sum_{i\ 1}^{m} u_i^0 a_i^p\right) - \sum_{i\ 1}^{m} u_i^0 \geq z_{LAR} \ .$$

□

Remarques :

1. si l'on désire $\sum_{i\ 1}^{m} u_i^0 = 0$, il suffit de retrancher z_{LAR}/m à chaque u_i^0 ;
2. le théorème ci-dessus fournit une autre démonstration de l'inégalité $z_{LAR} \leq z_{LP2}$ du théorème 2-1; en fait cette démonstration peut être considérée comme duale de la démonstration donnée au paragraphe 2-2.

Le résultat suivant permet d'observer que la première inégalité du Théorème 2-1 est "presque toujours" stricte.

THEOREME 2-5 :

Si la solution optimale de (LAR) est unique, alors l'égalité $z_{LAR} = z_{LP1}$ implique que ces deux quantités sont égales à z^*, et toutes les inégalités du théorème 2-1 deviennent des égalités.

preuve :

soit $i_1\text{-}i_2\text{-}\ldots\text{-}i_n$ l'unique séquence optimale pour (LAR); d'après le théorème fort des écarts complémentaires, il existe des variables duales optimales u_i^* et v_t^* telles que

$$b_{i_t t} - u_{i_t}^* - v_t^* = 0 \quad \text{pour tout } t$$

$$b_{jt} - u_j^* - v_t^* > 0 \quad \text{pour tout } t \text{ et tout } j \neq i_t$$

et $$\sum_i u_i^* + \sum_t v_t^* = z_{LAR} \ .$$

Considérons alors le TDTSP défini par les coûts

$$\hat{c}_{ij}^t = c_{ij}^t - u_j^* - v_t^*$$

et désignons par $\hat{z}^*$, $\hat{z}_{LP1}$ et $\hat{z}_{LAR}$ les valeurs des solutions

optimales du PDFSP, de (LP1) et de (LAR) respectivement, pour ces coûts $\hat{c}^t_{ij}$. D'après (2-1-5) et (2-1-7), il vient:

$$\hat{z}_{LP1} = z_{LP1} - z_{LAR}$$

et $$\hat{z}^* = z^* - z_{LAR}$$

D'autre part, puisque les longueurs de tous les arcs entrant dans le noeud (j,t) sont toutes diminuées de $u^*_j + v^*_t$, les coefficients $\hat{b}_{jt}$ du problème (LAR) correspondant deviennent:

$$\hat{b}_{jt} = b_{jt} - u^*_j - v^*_t = 0 \quad \text{si } j = i_t$$

$$> 0 \quad \text{sinon}$$

et $$\hat{z}_{LAR} = 0.$$

Si l'on a alors l'égalité $z_{LAR} = z_{LP1}$, les seules variables x^t_{ij} positives dans une solution optimale de $(\hat{LP1})$ doivent avoir un coût $\hat{c}^t_{ij}$ nul, puisque $\hat{z}_{LP1} = 0$. Par conséquent, on doit avoir

$$x^0_{0i_1} = 1 \quad \text{et} \quad x^0_{0j} = 0 \quad \text{pour tout } j \neq i_1$$

puisque le seul $\hat{c}^0_{0j}$ nul correspond à $j = i_1$ (car $\hat{b}_{j0} = \hat{c}^0_{0j}$).

Supposons que l'on ait démontré que

$$x^{t-1}_{i_{t-1}i_t} = 1 \quad \text{et} \quad x^{t-1}_{ij} = 0 \quad \text{pour tous } (i,j) \neq (i_{t-1}, i_t);$$

alors, d'après (2-1-3),

$$x^t_{jk} = 0 \quad \text{pour tout } j \neq i_t \text{ et pour tout } k;$$

puisque $\hat{b}^{t+1}_k > 0$ pour tout $k \neq i_{t+1}$ on a $\hat{c}^t_{i_t k} > 0$, et donc :

$$x^t_{i_t k} = 0 \quad \text{pour tout } k \neq i_{t+1}, \text{et par suite}$$

$$x^t_{i_t i_{t+1}} = 1 .$$

On a donc démontré par induction que la solution optimale de $(\hat{LP1})$ est une séquence, donc $\hat{z}^* = 0$ et toutes les autres égalités s'en déduisent immédiatement.

Remarque : la condition d'unicité de la solution optimale de (LAR) est indispensable. Dans l'exemple ci-contre, la matrice des b_{jt} est identiquement nulle, donc les 6 affectations sont optimales; l'unique solution optimale de (LP1) est aussi de coût nul, et est définie par les chemins 0-1-2-1-0, 0-2-3-2-0 et 0-3-1-2-0 avec chacun un poids de 1/3. Pourtant l'unique séquence optimale 0-1-2-3-0 a un coût de

c^0_{0j}	1	2	3
0	0	0	0

c^1_{ij}	1	2	3
1		0	2
2	2		0
3	0	2	

c^2_{ij}	1	2	3
1		2	0
2	0		1
3	2	0	

c^3_{i4}	4
1	0
2	0
3	0

Coûts c^t_{ij}

Pour l'instruction (3) de la méthode d'optimisation par sous-gradient, le critère d'arrêt peut être soit un nombre d'itérations fixé à l'avance, soit le fait que la valeur max $w(u^j)$ n'ait pas été améliorée depuis un certain nombre d'itérations, soit encore d'autres critères possibles portant sur la valeur relative des dernières améliorations. Pour la définition de u^k, un type d'itérations assez général est :

$$u^k = u^{k-1} + t_k(a^{p^{k-1}} - \underline{1}) \qquad (2\text{-}4\text{-}10)$$

où $(t_k)_{k=1,2,\ldots}$ est une suite de scalaires. Pour les propriétés et la convergence d'itérations fondées sur (2-4-10), voir Goffin [1976]. Si $\bar{z}$ désigne une borne supérieure connue sur z^* (par exemple $\bar{z}$ est le coût d'une séquence obtenue par une méthode heuristique, voir le paragraphe 1-7), et si $(\lambda_k)_{k=1,2,\ldots}$ désigne une suite de scalaires, on peut utiliser en particulier des itérations définies par:

$$t_k = \lambda_k \frac{\bar{z} - w(u^{k-1})}{\| a^{p^{k-1}} - \underline{1} \|^2} \qquad (2\text{-}4\text{-}11)$$

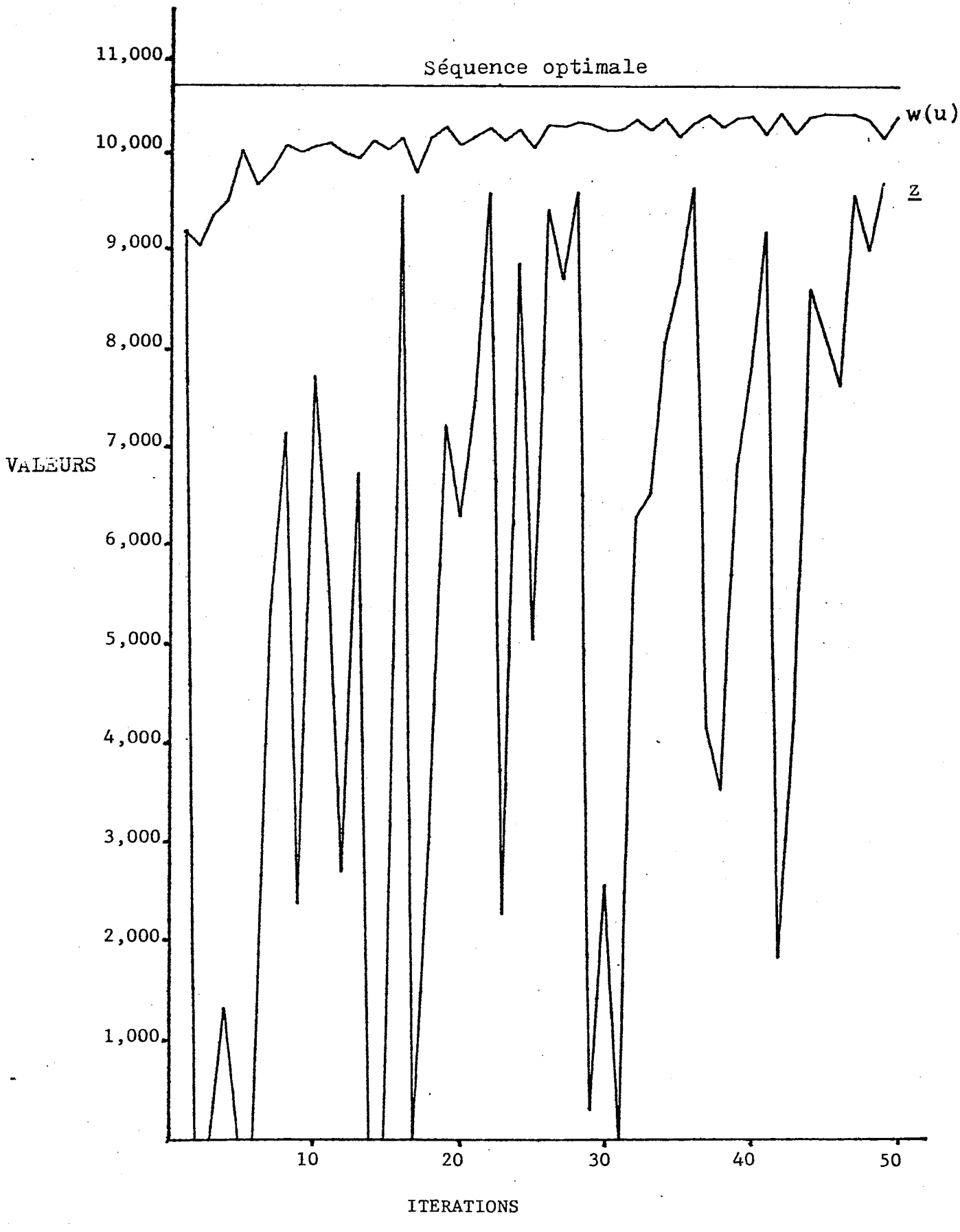
(Held et Karp [1971]). Pour $\lambda_k = 1$, ceci peut être interprété

comme une tentative pour fixer $w(u^k) = \bar{z}$ (plus précisément, on a $\ell^{p^{k-1}} + \sum_{i=1}^{m} u^k (a^{p^{k-1}} - \underline{1}) = \bar{z}$).

D'autres types d'itérations utilisant également les pénalités $u^{k-2}, u^{k-3}, \ldots, u^0$ sont décrits dans Camerini et al. [1975], Lemarechal [1975] et Wolfe [1975].

On peut comparer l'algorithme du simplexe par génération de colonnes et l'optimisation par sous-gradient. Ces deux méthodes requièrent la recherche d'un p.c.c. pour des pénalités données, puis modifient ces pénalités d'après le vecteur a^p correspondant au p.c.c. obtenu. La principale différence est dans la façon dont cette modification est effectuée. Comme conséquence, l'algorithme du simplexe fournit des bornes généralement plus mauvaises pour le même nombre d'itérations, mais sa convergence en un nombre fini d'itérations est garantie.

La figure suivante permet de comparer le comportement de ces bornes sur un exemple (TSP symétrique tiré de Dantzig et al. [1954]) avec $m = 32$; la méthode d'optimisation par sous-gradient utilise les itérations (2-4-10)-(2-4-11) avec $\lambda_k = 1$ pour tout k.


Séquence optimale
w(u)
z
11,000
10,000
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
VALEURS
10
20
30
40
50
ITERATIONS

2-5 Un algorithme par Branch-and-Bound pour le TDTSP :

L'ensemble des m! séquences dans le réseau R est partitionné en m sous-ensembles, définissant autant de sous-problèmes, selon la ville visitée en dernier. Après plusieurs séparations, un sous-problème est défini par un chemin partiel

$$i_{m-k+1}-i_{m-k+2}-\ldots-i_{m-1}-i_m-(m+1); \qquad (2\text{-}5\text{-}1)$$

soient $j_1, j_2, \ldots, j_s$ les indices des s villes qui ne figurent pas dans le chemin partiel et telles que les arcs associés aux coûts $C^{m-k}_{j_1 i_{m-k+1}}, C^{m-k}_{j_2 i_{m-k+1}}, \ldots, C^{m-k}_{j_s i_{m-k+1}}$ existent (dans le cas où R est complet, alors $s = m-k$); ce sous-problème pourra alors être séparé en s sous-problèmes, que l'on appelera sous-problèmes fils selon la ville $j_1, j_2, \ldots$ ou j_s que l'on fixera en $(m-k)^{\text{ème}}$ période. Un sous-problème tel que défini par (2-5-1) est un TDTSP de taille m-k, qui peut être représenté par un réseau dont les m-k premières périodes et transitions se déduisent de celles de R en supprimant les noeuds associés aux villes $i_{m-k+1}, i_{m-k+2} \ldots i_m$ et les arcs adjacents, tandis que la dernière transition contient les arcs de coûts $C^{m-k}_{j_1 i_{m-k+1}}, C^{m-k}_{j_2 i_{m-k+2}} \ldots C^{m-k}_{j_s i_m}$.

Certaines informations obtenues en traitant le problème père peuvent être utiles pour les sous-problèmes fils. Ainsi une borne supérieure $\bar{z}_{m-k}$ peut se déduire de la borne supérieure précédente $\bar{z}_{m-k+1}$ en posant

$$\bar{z}_{m-k} = \bar{z}_{m-k+1} - C^{m-k}_{j_r i_{m-k+1}}$$

pour le $r^{\text{ème}}$ sous-problème. De même, des "bonnes" pénalités initiales u^0 pour ce sous-problème peuvent se déduire des "meilleures" pénalités u* pour le problème père en posant:

$$u_j^0 = u_j^* + \frac{u_{j_r}^*}{m-k-1} \qquad \forall j = j_1,\ldots,j_s \quad j \neq j_r \qquad (2\text{-}5\text{-}2)$$

(le facteur constant est ajouté pour assurer que la somme des u_j^0 est nulle).

Le paragraphe suivant expose un procédé qui permet de réduire le nombre de sous-problèmes possibles au cours des itérations de plus court chemin dans le sous-problème père, c'est-à-dire avant même qu'on ne songe à le séparer.

-6 Bornes implicites:

L'algorithme de plus court chemin avec pénalités (PCC ou PCC2) fournit davantage d'informations que seulement un plus court chemin et sa longueur. En fait, cette longueur est calculée comme le minimum des $f(i,m)+C^m_{i,m\ 1}$ et ces quantités

$$b_i = f(i,m)+C^m_{i,m\ 1} \qquad (2\text{-}5\text{-}3)$$

représentent la longueur d'un plus court chemin de l'origine à l'extrèmité passant par le noeud (i,m). Clairement, la longueur d'une séquence se terminant par ...-i-(m+1) est bornée inférieurement par b_i . Si l'on utilise des pénalités dont la somme est nulle, cette propriété est vraie pour chacune des bornes $b_i(u)$ obtenues par (2-5-3) pour des pénalités u; par conséquent la quantité

$$B_i = \max\{b_i(u^j) : j = 0,1,\ldots,k\} \qquad (2\text{-}5\text{-}4)$$

obtenue au cours des k itérations de p.c.c. effectuées dans le problème père est une borne inférieure sur la valeur de la solution optimale du TDTSP dans le sous-problème fils défini en fixant la ville i en dernière position. Nous appelerons cette quantité B_i borne implicite pour ce sous-problème ("implicite", puisqu'elle peut être déterminée avant même que l'on ne s'interesse à ce sous-problème).

Les bornes implicites, définies lors des itérations de p.c.c. dans chaque sous-problème lors de l'énumération, auront deux usages. Premièrement, elles permettrons d'éliminer les sous-problèmes correspondants dès que certaines bornes implicites dépasseront une borne supérieure. Ceci sera réalisé en pratique en conservant une liste des villes restant dans la dernière période, c'est-à-dire des villes telles que le sous-problème correspondant n'ait pas encore été éliminé; au début, cette liste contiendra toute les villes qui ne figurent pas dans le chemin partiel (en fait, on verra au paragraphe suivant qu'on ne retiendra qu'un sous-ensemble de cet ensemble initial). Par la suite le minimum n'aura besoin d'être calculé que pour les villes restant dans cette liste L; ce procédé permet déjà d'obtenir une borne meilleure que celle fournie par les simples itérations de plus court chemin, mais on verra que l'on peut en déduire davantage : en fait une borne inférieure sur le coût de la meilleure séquence dans le sous-problème actuel est donnée par $w' = \min \{ B_i : i \in L \}$ puisque la séquence optimale doit bien contenir une ville dans la dernière période.

Le problème d'obtenir la meilleure borne inférieure est donc, en fait, celui de maximiser la quantité w'. En particulier, il ne sert à rien de tenter d'augmenter la longueur du plus court chemin trouvé à l'itération précédente si celui-ci a une longueur nettement plus petite que w'. Il paraît préférable d'utiliser, à la place de la direction de sous-gradient associée à ce chemin, la direction de sous-gradient réorienté définie par le plus court chemin se terminant par la ville i telle que

$$B_i = \min \{ B_j \} = w' .$$

Alors, à partir des pénalités actuelles u, on cherche plutôt à maximiser la plus petite des bornes implicites. En fait, on travaille déjà un niveau plus bas dans l'énumération, c'est-à-dire dans le sous-problème qui a la plus petite borne implicite, celui qui sera probablement le plus difficile à éliminer.

Algorithme d'optimisation de w' par sous-gradient réorienté:

(0) choisir u^0 et poser $k=0$, $L=\{1,2,\ldots,m\}$ et $B_i=-\infty$ $\forall i \in L$;

(1) pour les pénalités u^k, calculer les quantités $b_i(u^k)$ par une itération de plus court chemin;
remplacer B_i par $\max\{b_i(u^k);B_i\}$ pour tout $i \in L$
remplacer L par $L - \{i: B_i > \bar{z} \text{ et } i \in L\}$;

(2) si $L=\emptyset$ terminer (remonter)
sinon soit i* tel que $B_{i*} = \min\{B_i : i \in L\}$
chercher le plus court chemin se terminant en (i*,m) (en utilisant les marques fournies par l'algorithme de p.c.c.): soit p^k ce chemin;
si p^k est une séquence, terminer (cette séquence est optimale pour ce problème);

(3) identique à l'algorithme d'optimisation par sous-gradient;

(4) une borne inférieure pour z* est fournie par
$\min\{B_i : i \in L\}$
et une séquence de longueur strictement plus petite que $\bar{z}$ ne peut se terminer que par une des villes de L.

Des expériences effectuées sur plusieurs problèmes lors de la mise au point des algorithmes ont montré que cette modification était très intéressante, aussi bien pour les éliminations précoces qu'elles permettait que pour les bornes meilleures qu'elles fournissait. En fait, cette modification a apporté plus d'améliorations que le choix d'une stratégie "optimale" pour les itérations de sous-gradient; de plus, il n'est pas

du tout clair que les résultats théoriques ou expérimentaux concernant le comportement de différentes stratégies d'optimisation par sous-gradient soient encore valides dans ce cas.

2-7 Test de dominance:

Le test de dominance que nous allons maintenant introduire permet de réduire la liste L avant même de commencer les itérations de sous-gradient. L'idée essentielle est la suivante: considérons un sous-problème dans lequel au moins une ville a déjà été fixée; soit i_{m-k+1}-i_{m-k+2}-...-i_m-(m+1) le chemin fixé ($k \geqslant 1$) et soit i l'une des villes non fixées; si l'on décidait de fixer i en $(m-k)^{ème}$ position, le chemin fixé deviendrait i-i_{m-k+1}-i_{m-k+2}-...-i_m-(m+1). Or s'il existe une permutation j_{m-k+1}-j_{m-k+2}-...-j_m de l'ensemble des villes déjà fixées telle que la longueur du chemin résultant i-j_{m-k+1}-j_{m-k+2}-...-j_m-(m+1) soit plus petite que celle de i-i_{m-k+1}-i_{m-k+2}-...-i_m-(m+1), alors le sous-problème correspondant ne peut pas contenir de séquence optimale; par conséquent, la ville i peut être éliminée de la liste L des villes candidates à la dernière position.

Le problème de trouver s'il existe un chemin constitué des mêmes villes, mais dans un ordre différent et plus court que le chemin fixé est en fait aussi difficile que le TDTSP (la seule différence est qu'on ne cherche pas une solution optimale mais qu'on peut s'arrêter dès qu'on a trouvé une séquence meilleure). Aussi nous emploierons une méthode heuristique similaire à celle décrite au paragraphe 1-7 pour tenter d'améliorer le chemin fixé; en effet, dès qu'une amélioration est trouvée, on peut s'arrêter et éliminer le noeud i de L.

Le test de dominance sera réalisé de la façon suivante:

pour chaque ville i non fixée telle que l'arc $((i,m-k),(i_{m-k+1},m-k+1))$ existe, et pour chaque valeur de $r = 1,2,\ldots,k-1$ (r désigne la longueur du segment commençant par i_{m-k+1} que l'on va chercher à insérer plus loin dans le chemin), essayer d'insérer le segment $i_{m-k+1}-i_{m-k+2}-\ldots-i_{m-k+r}$ entre $i_{m-k+r+1}$ et $i_{m-k+r+2}$, puis entre $i_{m-k+r+2}$ et $i_{m-k+r+3}$, ..., enfin entre i_m et (m+1); dès qu'une amélioration est trouvée, passer à la valeur de i suivante; si aucune amélioration n'a pu être trouvée pour certaine ville i, alors mettre i dans la liste L.

La valeur de ce test de dominance a été particulièrement démontrée pour l'application au problème d'ordonnancement étudié au chapitre 4. Ce test est également interessant pour le TSP, quoique à un degré moindre. Dans ces deux cas, la complexité du temps de calcul nécessaire à l'exécution de ce test est d'un ordre de grandeur plus petite que pour le TDTSP (c'est-à-dire en $O(mk^2)$ au lieu de $O(mk^3)$), voir les chapitres suivants.

Discussion :

De nombreux aspects de l'algorithme que l'on vient de décrire peuvent être discutés. Nous discuterons seulement ici la règle de séparation; d'autres discussions relatives aux spécialisations de l'algorithme décrites dans les chapitres suivants seront développées à ce moment.

La séparation sur la dernière ville à être fixée présente plusieurs avantages : elle est assez simple à programmer; par exemple, il suffit de $2m^2$ changements environ dans la matrice des C^t_{ij} pour descendre d'un sous-problème à l'un de ses fils, passer à un sous-problème "frère" ou remonter; les sous-problèmes sont des TDTSP de taille m-1 et les bornes implicites ainsi que le test de dominance sont bien adaptés à ce type de séparation.

Mais il se peut que l'on effectue de la sorte un travail inutile si la partie importante des séquences ne se trouve pas à la fin. Par exemple, si le meilleur chemin obtenu est 0-1-2-3-4-5-3-7-8-9, c'est probablement dans les périodes 3 à 6 qu'il faudra intervenir pour être le plus efficace, car on risque, si l'on sépare par la fin, de continuer à obtenir le 3-circuit 3-4-5-3 sur ces périodes. Une alternative serait alors de "brancher sur" la $3^{ème}$ ou la $6^{ème}$ période; on peut otenir des bornes implicites (pour éliminer plusieurs sous-problèmes) en utilisant des distances (f(i,3) si l'on "branche sur" la $3^{ème}$ période) déjà disponibles et en calculant les distances des noeuds sur lesquels on branche (dans ce cas (i,3)) à l'extrèmité par la même méthode, mais en partant de l'extrèmité. Une autre alternative serait de brancher simultanément sur les périodes 8,7 et 6 (voir figure 1.) dans ce cas bornes implicites et test de dominance s'appliquent normalement (remarquer que le chemin 3 -7-8-9 ne peut pas être amélioré) et l'on a "sauté" quelques-uns des sous-problèmes.

On peut aussi séparer sur la période à laquelle une ville donnée (la ville 3 dans l'exemple ci-dessus) est visitée; ou encore utiliser simplement une stratégie binaire portant sur un seul arc -dans ce cas, l'un des deux sous-problèmes n'est pas nécessairement de taille plus petite. Certaines de ces stratétégies de séparation sont peut-être plus efficaces que la stratégie utilisée, mais sont aussi sans doute plus délicates à programmer et à mettre au point. Déterminer la "meilleure" stratégie reste une question largement ouverte à l'investigation.

Nous ne présenterons pas de résultats numériques détaillés pour le cas du TDTSP général. La principale raison en est que

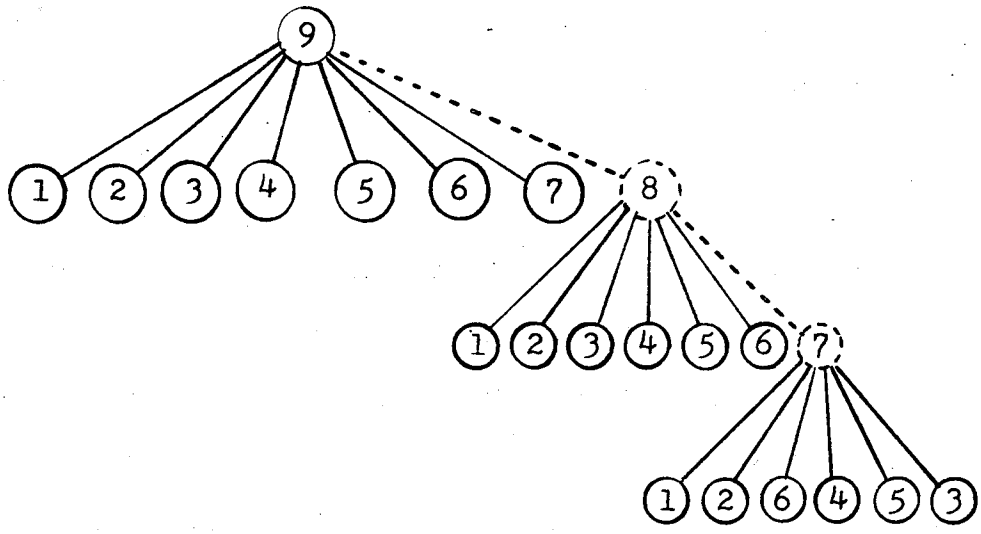


Figure 1.

le TDTSP a surtout été introduit comme modèle pour approcher des problèmes difficiles de permutations optimales ou des problèmes reliés à ceux-ci. De plus, définir des problèmes tests par des tirages aléatoires des valeurs C_{ij}^{t} a peu de sens car les problèmes spécialisés demandent en général, pour être résolus par cette méthode, plus d'efforts de calcul que des problèmes aléatoires.

Néammoins, quelques problèmes aléatoires à dix villes ont été facilement résolus par cette méthode, généralement sans séparation et en moins d'une seconde (IBM 360/50). Par contraste, on peut se rappeler que l'algorithme de K.Fox, basé sur des techniques de programmation en nombres entiers, était incapable de résoudre des problèmes de cette taille.

Les chapitres suivants concernent des applications de cette approche à des problèmes de tournées, le TSP, et à des problèmes d'ordonnancement sur une machine.

C H A P I T R E I I I : Application au problème du voyageur de commerce (TSP).

L'approche décrite dans les deux premiers chapitres peut, bien entendu, s'appliquer au cas particulier dans lequel les C_{ij}^t sont indépendants de t, c'est-à-dire au problème du voyageur de commerce (TSP). Pour cela, il convient d'abord de fixer une origine arbitraire, choisie parmi les villes du TSP et qui représentera le point de départ et d'arrivée de la tournée; dans le cas d'un problème de chemin hamiltonien (et non pas de circuit hamiltonien, comme pour le TSP) de longueur minimale, ceci n'est pas nécessaire. On peut alors construire le réseau multiparti correspondant, à n+1 périodes si le TSP comprend n villes à visiter, la première et la dernière période représentant l'origine, les autres contenant n-1 noeuds représentant les autres villes et les transitions, sauf la première et la dernière, contenant toutes les mêmes arcs de longueurs C_{ij}. Un chemin (de l'origine à l'extrémité) dans ce réseau peut être interprété comme un circuit contenant exactement n villes avec éventuellement des répétitions mais passant exactement une fois par l'origine.

Avant de discuter de la spécialisation au TSP de l'approche décrite dans les chapitres précédents, nous allons passer rapidement en revue quelques travaux antérieurs en relation étroite avec ce que nous allons développer.

Relation avec certains travaux antérieurs :

Nous avons déjà indiqué au début du chapitre 2 que le programme en nombres entiers (IP1) a été d'abord proposé par Hadley [1964] pour le TSP; l'extension aux coûts variables que nous avons considérée consiste simplement en une légére généralisation de la

fonction-objectif, ce qui ne change rien à la structure du polyèdre associé à sa relaxation continue (LP1), ni à la structure du polyèdre Q associé à la décomposition de Dantzig et Wolfe de ce programme linéaire. En fait notre approche tire explicitement parti de la structure très particulière des contraintes (2-1-1)-(2-1-4) et (2-1-8) (problème de plus court chemin) dans le cadre de cette décomposition de Dantzig et Wolfe. De plus, comme on l'a déjà indiqué, l'emploi des bornes implicites et de la méthode de sous-gradient réorienté permettent d'obtenir, au même coût de calcul, une borne inférieure meilleure que z_{LP1} .

Le problème de plus court chemin avec contrainte sur le nombre d'arcs (exactement n arcs) a été proposé comme relaxation pour le TSP par Saigal [1968] (voir également Rosseel [1968] pour certaines corrections à l'article de Saigal). L'algorithme proposé par Saigal pour trouver un tel plus court chemin est la version de l'algorithme PCC au cas des coûts indépendants de t; on peut noter que la même méthode a été également suggérée par Wagner [1970] comme un simple exercice d'application de la programmation dynamique. Les pénalités que nous introduisons apparaissent alors comme un procédé permettant d'améliorer, et de façon assez sensible, la borne inféreure fournie par la longueur d'un plus court chemin à n arcs. L'algorithme PCC2 peut être, lui aussi, considéré comme un raffinement destiné à éviter un comportement pathologique de l'algorithme PCC dans le cas du TSP, particulièrement pour des distances symétriques. On peut également mentionner la proposition de Hardgrave et Nemhauser [1962] : modifier les distances pour qu'un plus long circuit élémentaire soit un tour optimal; on peut alors considérer notre approche comme une tentative partielle d'assurer qu'un circuit obtenu comprendra

exactement n arcs, mais sans pouvoir garantir qu'il sera élémentaire; c'est d'ailleurs pour tenter de satisfaire cette contrainte de circuit élémentaire que sont introduites les pénalités.

Bien que, à première vue, elle puisse apparaître différente, la suggestion de Bernard Roy [1970,p.549] pour le TSP est en fait très proche de la notre. B.Roy considère un réseau qui est l'équivalent de notre réseau multiparti; dans ce réseau, on cherche un flot entier d'une unité qui satisfasse des"contraintes de faisceau", lesquelles sont l'équivalent exact, en termes de flots, des contraintes (2-1-5). B.Roy précise:"Or, la méthode proposée par Matthys pour rechercher un flot optimum satisfaisant à des contraintes de faisceau conduit en règle générale à une solution non entière" et plus loin :"Compte tenu de ce que l'on sait par ailleurs sur la difficulté des problèmes de parcours hamiltonien, la formulation présentée ci-dessus apparaît moins comme une contribution à l'étude de ces problèmes que comme une marque de la difficulté de l'étude des flots entiers avec contraintes de faisceaux." Notre étude peut alors être considérée comme une évaluation de la "contribution à l'étude de ces problèmes"; en effet la formulation de B.Roy est exactement équivalente à celle de Hadley et le coût d'un flot optimal (en général non entier) satisfaisant à ces contraintes de faisceaux est exactement z_{LP1} , aussi peut-on tenter de comparer les algorithmes susceptibles de fournir cette solution. Nous n'avons aucune indication permettant d'évaluer le comportement de la méthode de Matthys [1961] ,ni même si elle a jamais été programmée et testée;Klingman et Russell [1975] et Chen et Saigal [1977] ont développé des méthodes primales (la méthode de Matthys est primale-duale) permettant de

résoudre des problèmes de flots avec contraintes, mais il est difficile de prévoir le comportement de ces méthodes dans ce cas à cause de la dégénérescence massive qui apparaît dans les contraintes (2-1-2)-(2-1-4); la méthode proposée par Cunningham [1976] pour éviter les inconvénients de la dégénérescence dans les problèmes de flots est ici essentiellement équivalente à la recherche de tous les plus courts chemins depuis l'origine, ce qui est justement obtenu de façon très efficace par l'algorithme PCC. De plus, pour de tels problèmes exigeant sans doute un très grand nombre de pivotages, une méthode par sous-gradient est probablement (et dans l'état actuel de nos connaissances) la seule qui soit viable. Notre méthode apparaît donc comme une méthode spécialisée pour tirer parti de la structure très particulière du problème et devrait dominer ces autres méthodes plus générales (on pourrait aussi rappeler les avantages additionnels dûs aux bornes implicites et à l'algorithme PCC2).

A coté de ces propositions très proches les unes des autres, et dont notre étude fait la synthèse, il convient de rappeler quelques autres approches utilisées pour le TSP et qui ont certaines relations avec celle qui est étudiée ici. Les méthodes par Branch-and-Bound pour le TSP utilisant une relaxation fournie par un problème d'affectation linéaire (nous étudierons plus loin les relations entre cette relaxation classique et LAR) due à Dantzig et al. [1954] et Little et al. [1963], ont été développées par de nombreux autres auteurs; la contribution la plus remarquable, due à Srinivasan et Thompson [1972], a été l'introduction d'opérateurs de coûts pour la post-optimisation de ce problème d'affectation, voir Smith et al. [1975]. Pour les problèmes

symétriques, cette approche souffre d'une "pathologie" (voir Bellmore et Malone [1971]) dont notre approche souffre également et que nous discuterons dans le prochain paragraphe. La meilleure méthode pour les problèmes symétriques, à ce jour (Smith et Thompson, [1975]), est fondée sur les travaux de Held et Karp [1970] , [1971] et Christofides [1970] utilisant une relaxation par l'arbre de poids minimum proposée par Kruskal [1956] .

La méthode d'optimisation par sous-gradient que nous utilisons a été introduite par Held et Karp [1971] , justement dans leurs travaux sur le TSP. Cette méthode a été ensuite utilisée dans un nombre considérable de travaux ultérieurs, autant pour le TSP qu'en programmation linéaire, non-linéaire, en nombres entiers, pour des problèmes de flot, de localisation, d'ordonnancement,etc... Cette méthode est maintenant généralement préférée aux méthodes du genre "génération de colonnes" dans les algorithmes de Branch-and-Bound, pour les mêmes raisons que celles indiquées au paragraphe 2-4 et déjà discutées par Held et Karp. A ce titre, et à coté du fait qu'elle s'adresse au même problème, le TSP, notre approche doit aux travaux de Held et Karp, tout autant que beaucoup d'autres.

Pour ce qui est du TSP, la méthode que nous présentons ici diffère de celles que nous avons mentionné plus haut sur le principe de séparation; en effet, ces autres méthodes "branchent sur" les arcs (élimination des sous-tours dans l'approche basée sur l'affectation optimale, échange d' une arête dans l'arbre de poids minimal) alors que notre méthode "branche sur" les noeuds, en formant un chemin partiel se terminant au point d'arrivée. Nous connaissons en fait deux autres méthodes utilisant cette règle de

séparation pour le TSP : celle de Camerini et al. [1973] et celle de Bazaraa et Goode [1976] , toutes deux dans le cadre de la relaxation par l'arbre de poids minimal. Il est difficile de décider a priori laquelle de ces règles est préférable; si l'on compare des résultats expérimentaux, il faut savoir distinguer la part qui est due à une supériorité de la méthode elle-même plutôt qu'à une programmation plus raffinée de l'algorithme. La discussion sur la règle de séparation à adopter, que nous avions amorcée au § 2-8 reste donc encore largement ouverte.

3-2 Spécialisation au TSP : simplifications et difficultés

L'application au TSP de l'approche décrite au chapitre précédent présente étonnamment peu de simplifications, mais soulève plutôt un certain nombre de difficultés.

Comme on l'a décrit plus haut, l'algorithme PCC (et PCC2 également) s'adapte au cas du TSP de façon immédiate; il devient dans ce cas identique à l'algorithme proposé par Saigal. En fait, la seule différence dans les programmes en FORTRAN entre le cas des coûts variables C^t_{ij} et coûts constants C_{ij} est dans le nombre de dimensions de ce tableau C, le reste du programme étant absolument identique dans les deux cas.

Lorsque les longueurs C_{ij} sont symétriques, il est possible de réduire environ de moitié le temps d'exécution de ces algorithmes de plus court chemin. En effet, et à condition d'appliquer les pénalités u_i de manière symétrique, c'est-à-dire avec les longueurs pénalisées $C_{ij}-\frac{1}{2}u_i-\frac{1}{2}u_j$ (avec $u_0 = 0$, où 0 désigne l'origine choisie), on observe une symétrie parfaite dans le

réseau, soit par rapport à la période n/2 lorsque n est pair, soit par rapport à la transition (n-1)/2 lorsque n est impair. On peut exploiter cette symétrie de la façon suivante :

- pour n pair, appliquer un algorithme de plus court chemin (PCC) pour calculer toutes les distances f(i,n/2); il suffit alors de choisir une ville i qui minimise cette distance pour obtenir un plus court chemin symétrique de longueur 2f(i,n/2).
- pour n impair, calculer les distances f(i,(n-1)/2) et f(i,(n 1)/2) et choisir un noeud i qui minimise la somme de ces deux quantités; on obtient un plus court chemin, ayant cette longueur en mettant bout-à-bout les deux chemins correspondants.

Les mêmes idées s'appliquent, avec les changements correspondants, à l'algorithme PCC2.

Cette possibilité qui peut sembler avantageuse dévoile en fait un comportement "pathologique"de notre approche dans le cas symétrique. En effet, pour les problèmes avec un nombre pair de villes, on obtient toujours ainsi un chemin symétrique, qui ne pourra donc jamais être hamiltonien; en outre, pour qu'un chemin hamiltonien soit un plus court chemin, il faut que ses deux"moitiés" soient exactement de même longueur (en tenant compte des pénalités), ce qui restreint la dimension de l'ensemble des pénalités optimales dans le cas où LP2 a une solution optimale entière.

D'autre part, et toujours dans le cas symétrique, l'algorithme PCC a tendance à produire des oscillations (voir chapitre 1.), même lorsque l'on utilise des pénalités; c'est d'ailleurs ce qui a motivé le développement de l'algorithme PCC2. Cet algorithme PCC2 permet d'éviter les oscillations, mais peut encore produire des triangles (3-circuits), ou des circuits ayant quelques noeuds, quoique, en général, dans une moindre mesure que PCC. On constate qu'un plus court chemin construit par ces algorithmes a tendance,

à cause de la symétrie, à revenir à peu près par les mêmes noeuds que ceux qui lui ont permis de partir de l'origine, ces noeuds étant favorisés par les longueurs et les pénalités. Au cours des itérations de sous-gradient, le comportement observé consiste en une alternance de deux tels chemins, à peu près complémentaires, avec quelques modifications au fil des itérations, ce qui correspond à la convergence des pénalités vers les pénalités optimales.

Dans le cas où les longueurs ne sont pas symétriques, le comportement est moins défavorable, quoique l'on continue à obtenir des petits circuits, surtout dans les premières itérations.

3-3 Plus proches voisins et affectation optimale :

La relaxation (LAR) définie au chapitre 2 s'applique au TSP de la manière suivante :

-calcul des a_i^t :

pour $t=1$ poser $a_i^1 = d_{0i}$

pour $t=2,\ldots,m-1$ poser $a_i^t = a_i^2 = \min\{d_{ji} : j \neq 0 \text{ et } \neq i\}$

pour $t=m$ poser $a_i^m = \min\{d_{ji} : j \neq 0 \text{ et } \neq i\} + d_{i0} = a_i^2 + d_{i0}$

donc, à part pour la première période, a_i^t a été défini en associant à i un de ses plus proches voisins (dans le cas non symétrique: l'origine d'un plus court arc d'extrèmité i).

-solution de (LAR) :

ce problème d'affectation a toutes ses colonnes identiques, sauf la première et la dernière; le coût d'une affectation quelconque commençant par la ville i en première période et se terminant par j en dernière période est donc

$$a_i^1 + \sum_{k\neq i,j} a_k^2 + a_j^m = (a_i^1 - a_i^2) + (a_j^m - a_j^2) + \sum_{k=1}^{m} a_k^2$$

par conséquent, on peut résoudre (LAR) par simple inspection : choisir i et j distincts qui minimisent la quantité

$$(a_i^1 - a_i^2) + (a_j^m - a_j^2) \qquad (3\text{-}3\text{-}1)$$

On peut donc obtenir la borne z_{LAR} en $O(m^2)$ opérations. Cette borne z_{LAR} est en général très mauvaise et l'on peut l'améliorer en utilisant des pénalités. Supposons en effet que des pénalités u_i soient ajoutées aux longueurs des arcs ayant i pour origine (des pénalités pour les extrèmités sont sans intérêt puisqu'elles sont déjà assumées par les variables duales correspondant aux lignes du problème d'affectation (LAR)), alors la longueur de chaque tour (ou séquence) est augmentée de $\sum_i u_i$ mais la solution

de (LAR) peut être modifiée de façon différente. Soit donc $(LAR)_u$ le problème correspondant et $z_{LAR}(u)$ la valeur optimale de la fonction-objectif de ce problème. Un problème tout-à-fait comparable à celui traité dans le chapitre précédent est alors de trouver des pénalités u* qui maximisent $z_{LAR}(u) - \sum_i u_i$.

Définissons $$z_{PPV} = z_{LAR}(u^*) - \sum_i u_i^* = \max_u \left\{ z_{LAR}(u) - \sum_i u_i \right\} . \quad (3\text{-}3\text{-}2)$$

Nous allons montrer que la recherche de z_{PPV} est très proche de la résolution d'une relaxation classique du TSP.

Le problème d'affectation suivant :

$$(AP) \quad \min \sum_{i=0}^{m} \sum_{j=0}^{m} d_{ij} y_{ij} \qquad (3\text{-}3\text{-}3)$$

$$\text{s.c.} \quad \sum_{j=0}^{m} y_{ij} = 1 \qquad \text{pour tout } i = 0,1,\ldots,m \qquad (3\text{-}3\text{-}4)$$

$$\sum_{i=0}^{n} y_{ij} = 1 \qquad \text{pour tout } j = 0,1,\ldots,m \qquad (3\text{-}3\text{-}5)$$

$$y_{ij} = 0 \text{ ou } 1 \qquad \text{pour tous } i,j \quad i \neq j. \qquad (3\text{-}3\text{-}6)$$

constitue une relaxation pour le TSP, voir par exemple Dantzig et al. [1954], Bellmore et Nemhauser [1968], Bellmore et Malone [1971], Smith et al. [1974]. Il est bien connu que la solution optimale de (AP) présente en général des sous-tours ; on peut alors introduire des contraintes supplémentaires pour éviter ce phénomène. Pour le but que nous poursuivons ici, relation avec les plus proches voisins, nous n'introduirons qu'un sous-ensemble très modeste de ces contraintes : nous interdirons seulement les sous-tours passant par 0 et par un seul autre noeud; ces contraintes correspondent au choix de i et j distincts que l'on fait en résolvant (LAR). Les contraintes suivantes expriment cette interdiction:

$$y_{0i} + y_{i0} \leq 1 \qquad \text{pour tout } i = 1,2,\ldots,m \qquad (3\text{-}3\text{-}7)$$

Nous désignerons par (APC) le problème (AP) avec ces m contraintes

supplémentaires (3-3-7), et par z_{AP} et z_{APC} les valeurs optimales des fonctions-objectif de ces problèmes. Evidemment, on a l'inégalité : $z_{AP} \leqslant z_{APC}$ et, de plus, ces valeurs sont en général peu différentes. En effet, si la solution optimale de (AP) ne contient pas de sous-tour de longueur 2 passant par l'origine 0, on a l'égalité; sinon, soit i la ville ainsi associée à l'origine, on peut partitionner l'ensemble des affectations en deux sous-ensembles (exhaustifs et deux à deux disjoints); le premier est défini par $y_{0i} = 1$ et $y_{i0} = 0$, et une affectation optimale dans ce sous-ensemble est une solution réalisable pour (APC) et peut être obtenue par une post-optimisation à partir de la solution optimale de (AP); l'autre sous-ensemble est défini par $y_{0i} = 0$ et ce problème peut fournir, après également une post-optimisation, une solution qui ne soit toujours pas réalisable pour (APC) (c'est-à-dire avec une ville $j \neq i$ telle que $y_{0j} = y_{j0} = 1$) et qui oblige à séparer de nouveau. Après au plus m séparations les contraintes (3-3-7) sont satisfaites et il suffit alors de prendre des solutions réalisables ainsi obtenues. Chacune de ces post-optimisations peut être réalisée par un algorithme en $O(m^2)$ (Edmonds et Karp [1971] , Tomizawa [1971]) donc z_{APC} peut être obtenu en au plus $O(m^3)$ opérations.

THEOREME 3-1 :

$$z_{AP} \leqslant z_{PPV} \leqslant z_{APC} \leqslant C^*$$

preuve: pour démontrer ces inégalités, nous allons introduire un problème "dual lagrangien" (DAPC) associé à (APC), tel que $z_{AP} \leqslant z_{DAPC} \leqslant z_{APC}$ et nous démontrerons que $z_{PPV} = z_{DAPC}$.

La dernière inégalité du théorème vient du fait qu'un tour optimal est une affectation "sans sous-tours", c'est-à-dire

une solution réalisable pour (APC).

Etant donnés des nombres u_i $(i=1,2,\ldots,m)$ que nous appelerons ici des "multiplicateurs de Lagrange", nous définirons la relaxation lagrangienne $(RAPC)_u$ de (APC) en "introduisant" les contraintes (3-3-4), sauf celle correspondant à $i=0$, dans la fonction-objectif de (APC); on obtient ainsi le programme linéaire en nombres entiers

$$(RAPC)_u \quad \min \sum_{i=0}^{m}\sum_{j=0}^{m} d_{ij}y_{ij} + \sum_{i=1}^{m} u_i \left(\sum_{j=0}^{m} y_{ij} - 1 \right) \qquad (3\text{-}3\text{-}8)$$

$$\text{s.c.} \sum_{i=0}^{m} y_{ij} = 1 \quad \text{pour tout } j = 0,1,\ldots,m \qquad (3\text{-}3\text{-}5)$$

$$\sum_{j=1}^{m} y_{0j} = 1 \qquad (3\text{-}3\text{-}4')$$

$$y_{0i} + y_{i0} = 1 \quad \text{pour tout } i = 1,\ldots,m \qquad (3\text{-}3\text{-}7)$$

$$y_{ij} = 0 \text{ ou } 1 \quad \text{pour tous } i,j \quad i \neq j \qquad (3\text{-}3\text{-}6)$$

Considérons alors le problème "dual lagrangien"

$$(DAPC) \quad \max z_{RAPC}(u) \quad \text{pour} \quad u \in \mathbb{R}^m \qquad (3\text{-}3\text{-}9)$$

où $z_{RAPC}(u)$ désigne la valeur optimale de (3-3-8). Puisque chaque $(RAPC)_u$ est une relaxation pour (APC), on a

$$z_{RAPC}(u) \leqslant z_{APC}$$

d'où, en désignant par z_{DAPC} la valeur optimale de (3-3-9)

$$z_{DAPC} \leqslant z_{APC} \qquad (3\text{-}3\text{-}10).$$

On peut effectuer la même construction à partir de (AP), en définissant les relaxations lagrangiennes $(RAP)_u$, identiques à $(RAPC)_u$ sauf les contraintes (3-3-7), d'où

$$z_{RAP}(u) \leqslant z_{RAPC}(u) \qquad (3\text{-}3\text{-}11),$$

et le problème dual lagrangien

$$(DAP) \quad \max z_{RAP}(u) \quad \text{pour } u \in \mathbb{R}^m \qquad (3\text{-}3\text{-}12).$$

On déduit alors de (3-3-11)

$$z_{DAP} \leqslant z_{DAPC} \qquad (3\text{-}3\text{-}13).$$

Mais d'après la "propriété d'intégrité" (unimodularité) du problème (AP), on a l'égalité

$$z_{DAP} = z_{AP} \qquad (3\text{-}3\text{-}14)$$

voir, par exemple Geoffrion [1974].

Il nous reste maintenant à démontrer l'égalité $z_{PPV} = z_{DAPC}$. Pour cela, il suffira de démontrer que, pour tout u, les problèmes $(LAR)_u$ et $(DAPC)_u$ sont équivalents.

Soit une affectation optimale pour $(LAR)_u$; une telle solution est définie par deux villes i et j distinctes, i étant la "première" ville de la solution et j la "dernière". A cette solution associons une solution y définie de la façon suivante:

(i) $y_{0i} = 1$; $y_{0k} = 0$ pour tout $k \neq i$

$y_{ki} = 0$ pour tout $k \neq 0$

(ii) pour chaque $k \neq 0$ et $\neq i$, choisir une ville $h \neq 0$ telle que $d_{hk}+u_h = \min\{d_{gk}+u_g : g \neq 0\}$

poser $y_{hk} = 1$ et $y_{gk} = 0$ pour tout $g \neq h$

(iii) $y_{j0} = 1$ et $y_{g0} = 0$ pour tout $g \neq j$.

D'après cette définition, on a clairement $y_{ij} = 0$ ou 1 pour tous les i et j; les différentes contraintes de $(RAPC)_u$ sont satisfaites comme on peut le vérifier aisément :

(3-3-5) d'après (i) pour i, et d'après (ii) pour les $k \neq i$

(3-3-4') d'après (i)

et (3-3-7) d'après (i), (iii) et le fait que i et j soient distincts.

Enfin on peut vérifier l'égalité des fonctions-objectif en remarquant que (3-3-8) peut s'écrire:

$$\sum_{j=1}^{m} d_{0j}y_{0j} \quad \sum_{i=1}^{m}\sum_{j=0}^{m}(d_{ij}+u_i)y_{ij} \quad - \sum_{i=1}^{m} u_i \qquad (3\text{-}3\text{-}13)$$

Réciproquement, pour résoudre $(RAPC)_u$, il suffit, après avoir décidé quel serait l'indice i tel que $y_{0i}=1$, unique d'après (3-3-4')et l'indice j tel que $y_{j0}=1$, unique d'après $(3\text{-}3\text{-}5)_{j=0}$,de choisir pour chaque $k \neq 0$ et $\neq i$, la ville h la plus proche, c'est-à-dire telle que

$$d_{hk}+u_h = \min\{d_{gk}+u_g \ : \ g \neq 0\}.$$

Il suffit alors de vérifier que le choix de i et j est identique pour $(LAP)_u$ et $(RAPC)_u$, ce qui découle simplement de la formulation (3-3-13) et de (3-3-0).

On a donc $z_{LAP}(u) = z_{RAPC}(u)$

d'où $z_{PPV} = z_{DAPC}$ ce qui achève la démonstration. □

Par conséquent, et bien que les problèmes de"plus proches voisins" $(LAR)_u$ soient faciles à résoudre, ils ne peuvent pas fournir de borne meilleure que z_{APC} ,c'est-à-dire à peine meilleure que la borne classique z_{AP} obtenue en résolvant le problème d'affectation (AP). De plus le problème d'affectation (AP) (et aussi (APC)) est facile à résoudre et possède une structure qui se prête bien aux post-optimisations requises par un algorithme de Branch-and-Bound, voir Srinivasan et Thompson [1973] .

En conclusion, on peut considérer que l'approche fondée sur les problèmes de plus proches voisins $(LAR)_u$ avec pénalités n'apparaît pas intéressante pour le TSP.

3-4 TDTSP et affectation optimale :

Le théorème 3-1 nous permet de comparer la borne z_{LP2} que l'on peut obtenir par les techniques décrites au chapitre 2 avec la borne classique z_{AP} :

Corollaire 3-1:

$$z_{AP} \leqslant z_{LP2} \qquad (3\text{-}4\text{-}1)$$

preuve: pour tout u, on a $z_{LAR}(u) \leqslant z_{LP2}$,d'après le théorème (2-1) et le fait que z_{LP2} est invariant pour des pénalités dont la somme est nulle; (3-4-1) découle alors immédiatement du théorème (3-1) et de la définition de z_{PPV} .

Comme on l'a vu plus haut, (LAR) et (AP) ne sont pas équivalents et ce corollaire est un résultat plus fort que la première inégalité du théorème (3-1). On peut même démontrer que la borne z_{LP2} est "presque toujours" strictement meilleure que z_{AP} , dans le sens suivant:

THEOREME 3-2 :

Si l'on a l'égalité $z_{AP} = z_{LP2}$

alors l'une au moins des deux conditions suivantes est vérifiée:

(i) le tour optimal est solution optimale à la fois pour (AP) et (LP2);

(ii) la solution optimale de (AP) n'est pas unique.

preuve:

pour démontrer ce théorème, nous utiliserons une autre démonstration du corollaire (3-1); nous verrons dans un paragraphe ultérieur une troisième démonstration, "duale" de celle-ci, pour ce même corollaire.

Considérons une solution optimale $\bar{x}$ pour (LP1) et associons à cette solution une solution $\bar{y}$ pour (AP) définie de la façon suivante:

(i) $\bar{y}_{0j} = \bar{x}^0_{0j}$ pour tout j (et bien sûr, comme toujours, $\bar{y}_{00} = 0$)

(ii) $\bar{y}_{ij} = \sum_{t=1}^{m-1} \bar{x}^t_{ij}$ pour tous i et j ($i \neq j$) différents de 0 (et $y_{ii} = 0$ pour tout i)

(iii) $\bar{y}_{i0} = \bar{x}^m_{i,m+1}$ pour tout i.

Cette solution $\bar{y}$ satisfait les contraintes (3-3-5) d'après les contraintes (2-1-6) pour $j = 0$, et (2-1-5) pour les autres j. Elle satisfait les contraintes (3-3-4) de (AP) d'après les contraintes (2-1-1) pour $i = 0$, et d'après (3-3-5) (que l'on vient d'établir)et par les relations (2-1-4),(2-1-3) et (2-1-2) pour les autres valeurs de i.

Cette solution y est donc une solution continue de (AP); mais puisque la valeur optimale z_{AP} ne change pas si l'on relâche les contraintes (3-3-6) en :

$$y_{ij} \geq 0 \quad \text{pour tous i et j } (i \neq j)$$

on en déduit que la valeur de (3-3-3) pour $\bar{y}$ est supérieure ou égale à sa valeur optimale z_{AP} ; on peut vérifier de façon immédiate que la valeur de (3-3-3) pour $\bar{y}$ est égale à z_{LP1} puisque l'on a choisi $\bar{x}$ solution optimale de (LP1), et donc égale à z_{LP2} d'après le théorème (2-1).

A ce point, nous avons obtenu une seconde démonstration du corollaire (3-1). Supposons alors que $z_{AP} = z_{LP1}$ et que la solution optimale de (AP)soit unique. Considérons le graphe orienté $G = (X, A)$ où $X = \{0, 1, \ldots, m\}$ et $A = \{(i,j) : y_{ij} > 0\}$. D'après (LP1), nous savons qu'il existe, pour tout $i \neq 0$, un chemin partant de 0, visitant i au moins une fois et retournant à 0; G est donc fortement connexe. Mais comme G représente l'unique solution optimale de (AP), il ne peut contenir de sous-tours (puisque fortement connexe) et doit donc être un tour.

□

Remarque :

Si la solution optimale de (AP) n'est pas unique, l'égalité $z_{AP} = z_{LP1}$ n'implique pas nécessairement que le tour optimal soit solution optimale de (AP) ou (LP1).

Exemple : pour les longueurs données dans le tableau ci-contre, il y a deux affectations optimales de coût nul, l'une représentée par les sous-tours 0-1-2-0 et 3-4-3, et l'autre par les sous-tours 0-3-4-0 et 1-2-1; la solution optimale, unique, de (LP2) est également de coût nul, et peut être exprimée comme la demi-somme des chemins 0-1-2-1-2-0 et 0-3-4-3-4-0.

d_{ij}	j=0	1	2	3	4
0		0	1	0	1
1	1		0	1	1
i 2	0	0		1	1
3	1	1	1		0
4	0	1	1	0	

longueurs d_{ij}

Pourtant le tour optimal 0-1-2-3-4-0 est de longueur 1.

La borne z_{LP1} apparaît donc meilleure que la borne classique z_{LAR} pour le TSP. Rappelons que, pour obtenir z_{LP1} ,ou au moins une borne inférieure assez serrée sur cette valeur, il est nécessaire d'appliquer <u>plusieurs</u> itérations de plus court chemin, qui nécessitent chacune $O(m^3)$ opérations, alors que z_{AP} peut être obtenu en résolvant <u>un seul</u> problème d'affectation en $O(m^3)$ opérations. On ne peut donc conclure a priori sur la valeur de la relaxation TDTSP par rapport au problème d'affectation (AP) pour le problème du voyageur de commerce; cette valeur dépend du nombre d'itérations nécessaires, du temps de calcul pour une itération par rapport au temps de solution de (AP), des possibilités de post-optimisation pour chacune de ces relaxations, et l'évaluation finale de ces méthodes devra être fournie par les résultats d'expériences numériques.

3-5 Expériences numériques :

Nous avons écrit un programme FORTRAN pour l'algorithme de Branch-and-Bound décrit au chapitre 2, appliqué au cas particulier du TSP. Les seules particularités spécifiques à cette application sont
- dans l'algorithme PCC2 utilisé, suppression du troisième indice t dans la matrice des longueurs $d_{ij} = c_{ij}^t$ (sauf pour $t=0$ et $t=m$, où $d_{0j} = c_{0j}^0$ et $d_{i0} = c_{i,m+1}^m$);
- dans le Branch-and-Bound, manipulation de cette matrice des longueurs à seulement deux dimensions, ce qui est plus simple que le cas de trois dimensions.

Le programme est précédé par une méthode heuristique, semblable à celle de Reiter et Sherman [1965] et qui est décrite au chapitre 1, pour obtenir une bonne solution réalisable. La seule différence avec la méthode de Reiter et Sherman provient du fait que nous nous adressons à des problèmes qui ne sont pas nécessairement symétriques; par conséquent nous ne considérons pas les inversions. Les paramètres pour cet algorithme ont été choisis ainsi: le nombre de solutions aléatoires de départ est égal au nombre de villes, et la longueur maximale d'un segment que l'on cherche à insérer est fixée à $r=5$ (remarquons que, en permettant les inversions et en fixant $r=\lfloor n/3 \rfloor$ on obtient la méthode 3-opt de Lin [1965] dans le cas symétrique).

L'algorithme de Branch-and-Bound utilise PCC2 pour calculer les bornes, et les paramètres utilisés dans l'optimisation par sous-gradient réorienté sont désignés par p1,p2,p3,p4 et leurs valeurs figurent dans la table I pour les différents exemples; ces paramètres sont définis ainsi:
- dans le sous-problème initial, le nombre maximum d'itérations

de sous-gradient réorienté avant branchement est pl; ces itérations utilisent les formules de transformation (2-4-10)-(2-4-11) avec $\lambda = 1$; s'il se produit p2 itérations successives sans amélioration de w(u), alors λ est multiplié par p3 (p3 <1); s'il se produit de nouveau p2 itérations sans amélioration de w(u), λ est encore multilplié par p3, etc... jusqu'à un maximum de pl itérations;

- après la première séparation et pour chaque sous-problème, λ est de nouveau fixé à la valeur 1, et les modifications de λ sont définies par les mêmes paramètres p2 et p3; toutefois le nombre maximum d'itérations est p4, avec p4 <pl.

La raison pour laquelle on effectue moins d'itérations dans les sous-problèmes est que l'on dispose pour ceux-ci de bonnes pénalités initiales déduites des"meilleures" pénalités trouvées pour le problème-père, par les relations (2-5-2), alors que pour le problème initial, on part des pénalités arbitraires $u^0 = 0$.

Enfin l'origine est fixée arbitrairement comme étant la ville d'indice n; les villes intermédiaires sont donc 1,2,...,n-1 (=m).

Nous avons testé cet algorithme sur 7 problèmes non symétriques. Les matrices d_{ij} ont été tirées comme des nombres entiers pseudo-aléatoires indépendants, uniformément distribués entre 0 et 99 (les termes diagonaux d_{ii} sont évidemments fixés à $+\infty$). La table I ci-dessous indique les valeurs des paramètres utilisées pour ces différents exemples et les temps de solution (CDC Cyber 173, Université de Montréal).

Les résultats sont assez décevants et montrent que l'approche par le TDTSP ne semble pas intéressante pour le TSP, à moins qu'il ne soit possible de trouver des améliorations importantes pour accélérer certaines des phases de calcul (comme par exemple les "opérateurs de coût" de Srinivasan et Thompson pour la relaxation AP).

Problème	nombre de villes	paramètres p1	p2	p3	p4	temps de solution CDC Cyber 173 (sec.)
1	20	40	10	.5	5	17".170
2	20	40	10	.5	10	1".801
3	20	40	7	.5	5	8".583
4	30	60	10	.5	15	107".741
5	30	45	10	.5	5	145".916
6	30	60	5	.7	15	32".304
7	40	40	7	.5	5	>240" non-résolu

TABLE I Expériences avec l'algorithme appliqué au TSP.

C H A P I T R E I V : Application à un problème d'ordonnancement sur une machine.

1 Définitions:

Le problème d'ordonnancement suivant est considéré dans ce chapitre: étant donnés une machine, disponible de manière continue et sur laquelle on ne peut effectuer qu'une seule tâche à la fois, et un ensemble de tâches $J_1, J_2, \ldots, J_n$ à effectuer sur cette machine, déterminer l'ordre dans lequel ces tâches devront être effectuées de façon à minimiser le coût total de retard. Toutes les tâches peuvent commencer à l'instant $t = 0$, doivent être effectuées sans interruptions et ont une durée d'execution $p_i \geqslant 0$, indépendante de l'intervalle de temps pendant lequel la tâche J_i est exécutée; soit t_i l'instant auquel la tâche J_i est terminée. A chaque tâche est associée une <u>fonction de retard</u> non-décroissante $C_i(t)$ qui représente le coût encouru lorsque cette tâche J_i est prête à l'instant t. Le problème est donc de déterminer une permutation w des indices 1,2,...,n qui minimise le coût de retard total

$$C(w) = C_{w(1)}(t_{w(1)}) + C_{w(2)}(t_{w(2)}) + \ldots + C_{w(n)}(t_{w(n)}) \qquad (4\text{-}1\text{-}1)$$

où $t_{w(1)} = p_{w(1)}$

$t_{w(2)} = t_{w(1)} + p_{w(2)}$

.....

$t_{w(n)} = t_{w(n-1)} + p_{w(n)}$ (4-1-2).

Nous allons rappeler quelques exemples de fonctions C_i qui donnent des problèmes d'ordonnancement sur une machine qui ont déjà été étudiés dans la littérature. Dans tous ces exemples, une <u>date prévue</u> d_i est associée à chacune des tâches (d_i peut être aussi bien négative, ce qui exprime simplement que la tâche J_i est particulièrement urgente)

a) $C_i(t) = \begin{cases} 0 & \text{si } t \leqslant d_i \\ a_i & \text{sinon;} \end{cases}$

ceci définit un problème NP-complet (Karp [1972]); dans le cas où tous les a_i sont égaux, le problème est alors simplement de minimiser le nombre de tâches en retard, et il existe de"bonnes" méthodes de solution pour ce problème (voir Baker [1974]).

b) $C_i(t) = a_i(t - d_i)$;

ceci définit un problème ("weighted lateness problem") qui revient à minimiser la somme pondérée des temps passés dans le système (attente plus temps d'exécution); ce problème est résolu en faisant passer les tâches dans un ordre non-décroissant des rapports p_i/a_i (WSPT), voir Baker [1974] .

c) $C_i(t) = a_i \max\{0; t-d_i\} + b_i t$;

il s'agit ici de minimiser une somme pondérée des temps de retard $\max\{0; t-d_i\}$ et des temps passés dans le système; si tous les b_i sont nuls, le problème est de minimiser les retards pondérés ("weighted tardiness problem"), et c'est sur des problèmes de ce type que sera testée l'approche présentée ici; lorsque de plus tous les poids a_i sont égaux, il s'agit alors de minimiser le retard total; les problèmes dans cette classe c) sont considérés comme difficiles et ont déjà reçu beaucoup d'attention (voir Baker [1974] , [1976] , et Rinnooy Kan et al. [1975] pour d'autres références)

Dans toute la suite, les fonctions $C_i(t)$ seront des fonctions non-décroissantes arbitraires, à moins que le contraire ne soit spécifié (retards pondérés pour les expériences numériques).

-2 La relation de précédence:

Le principal résultat théorique disponible à propos des problèmes d'ordonnancement de ce type est que l'on peut déterminer certaines relations entre les tâches, appelées relations de précédence, telles qu'il existe une séquence optimale dans laquelle une tâche J_i dont on a déterminé qu'elle précédait une autre tâche J_j ,est exécutée avant cette tâche J_j. Outre la question de garantir que ces relations de précédence sont compatibles (c'est-à-dire ne présentent pas de circuit), il reste que ces relations ne sont pas en général suffisantes pour déterminer complètement une séquence optimale. Néammoins, ceci permet de réduire, souvent dans une proportion considérable, le nombre de solutions possibles (soit n! dans le problème initial).

Nous ne développerons pas ici les détails concernant la définition, la justification et le calcul de ces relations de précédence car ceci est très bien exposé par Emmons [1969] dans le cas du retard total et aussi de fonctions C_i convexes non-décroissantes, et par Rinnooy Kan et al. [1975] qui ont étendu les résultats d'Emmons à des fonctions non-décroissantes quelconques.

On peut considérer que ces relations sont regroupées en une seule relation R, où iRj signifie que J_i précède J_j dans une solution optimale; cette relation R sera considérée comme transitive et antisymétrique, c'est dire qu'elle constitue un ordre partiel strict (la réflexivité n'a pas grande importance ici); nous appelerons cette relation la relation de précédence et nous la supposerons donnée (c'est-à-dire déjà calculée). A chaque indice i on peut

associer l'ensemble B_i des ascendants de i, c'est -à-dire des j tels que jRi, et l'ensemble A_i des descendants de i, c'est-à-dire des k tels que iRk. Enfin nous dirons que i est un prédecesseur (immédiat) de j si iRj et si $A_i \cap B_j = \emptyset$, c'est-à-dire s'il n'existe pas d'indice k "entre" i et j dans R.

Une séquence w(1),w(2),...,w(n) sera dite satisfaire la relation R si, pour toutes les positions k et ℓ , w(k)Rw(ℓ) implique k<

Le problème qui nous intéresse est donc réduit à la recherche d'une séquence de coût (4-1-1) minimal parmi les séquences qui satisfont la relation R. Pour cela nous allons définir un TDTSP tel que les seules séquences réalisables satisferont la relation R, et dont la longueur sera une borne inférieure sur le coût de cette séquence; ce TDTSP est donc une relaxation (de type 1) pour ce problème d'ordonnancement; comme on utilise pour résoudre le TDTSP une relaxation de type 2, constituée par les plus courts chemins, cette approche fournit une relaxation de type 3 pour le problème d'ordonnancement considéré.

-3 Une relaxation par le TDTSP :

Pour définir une telle relaxation, nous associerons au problème d'ordonnancement un réseau à n+2 périodes (dont une origine et une extrèmité fictives) dans lequel chaque arc ((i,s),(j,s+1)) sera défini si et seulement si il existe une séquence satisfaisant R et dans laquelle les $s^{ème}$ et $(s+1)^{ème}$ tâches exécutées sont i et j respectivement (théorème 4-1) et dont la longueur C^s_{ij} est une borne inférieure sur le coût $C_j(t_j)$ pour toute séquence vérifiant ces propriétés (théorème 4-2).

Désignons par B_{ij} l'ensemble $\{i\} \cup B_i \cup B_j$ et par A_{ij} l'ensemble $\{j\} \cup A_i \cup A_j$, et soit $D_{ij} = \{1,2,...,n\} - (B_{ij} \cup A_{ij})$.

THEOREME 4-1 : Il existe au moins une séquence satisfaisant R et dans laquelle la $s^{ème}$ tâche exécutée est J_i et la $(s+1)^{ème}$ est J_j si et seulement si

(i) i est un prédécesseur ("immédiat") de j , ou bien $i \notin A_j \cup B_j$;

(ii) $s \geq |B_{ij}|$

et (iii) $n-s \geq |A_{ij}|$.

preuve:

conditions nécessaires:

(i) si $i \in A_j$ mais n'est pas un prédécesseur de j, il existe un indice $k \in A_i \cap B_j$;alors la tâche J_k devrait être exécutée après la $s^{ème}$ tache J_i ,mais avant la $(s+1)^{ème}$ tâche J_j , ce qui est impossible;

si $i \in B_j$ la contradiction est immédiate.

(ii) toutes les tâches J_k telles que kRi ou kRj, ainsi que J_i ,doivent être exécutées dans les s premières positions.

(iii)toutes les tâches J_k telles que iRk ou jRk, ainsi que J_j ,doivent être exécutées dans les n-s dernières positions.

conditions suffisantes: supposons que (i),(ii) et (iii) sont vérifiées et considérons l'ensemble D_{ij} des tâches qui n'ont aucune relation de précédence (dans un sens ou dans l'autre) avec i ni avec j .D'après l'hypothèse (i), les ensembles B_{ij} et A_{ij} sont disjoints. On a donc trois parties B_{ij} ,D_{ij} et A_{ij} disjointes qui recouvrent $\{1,2,\ldots,n\}$ (on serait tenté de dire qu'elles forment une partition de cet ensemble, mais il se peut que l'une, ou deux de ces parties, soit vide). Considérons alors les restrictions de R à $B_{ij}-\{i\}$, à D_{ij} et à $A_{ij}-\{j\}$ respectivement : ce sont des

ordres partiels stricts, donc il existe sur chacun de ces ensembles un ordre total qui prolonge ces ordres partiels (cf. Roy [1970]). On peut alors construire une séquence de la manière suivante :

- placer d'abord les éléments de $B_{ij}-\{i\}$ dans un ordre compatible avec R;
- placer ensuite les $s-|B_{ij}|$ premiers éléments de D_{ij} dans un ordre compatible avec R;
- en $s^{ème}$ et $(s+1)^{ème}$ positions, placer i puis j ;
- placer les éléments restants de D_{ij} , toujours dans un ordre compatible avec R
- et enfin placer les éléments de $A_{ij}-\{j\}$, également dans un ordre compatible avec R.

Ceci est possible grâce aux propriétés (ii) et (iii) (de nouveau, précisons que certaines de ces sous-séquences peuvent être vides). Il reste à vérifier que la séquence obtenue satisfait la relation R, ce qui découle immédiatement de (i) et des définitions de B_{ij}, A_{ij} et D_{ij} . □

Corollaire 4-1: il existe au moins une séquence satisfaisant R et dans laquelle la première tâche exécutée est J_i si et seulement si $B_i = \emptyset$.

Nous pouvons maintenant associer des longueurs aux arcs $((i,s),(j,s+1))$ définis par les conditions du théorème précédent pour construire un réseau R' qui permettra d'obtenir une borne inférieure sur le coût de retard minimum.

Lemme 4-1: (construction du réseau R')

Pour tous les i,j,s satisfaisant aux conditions du théorème 4-1, définissons

$C^0_{0j} = C_j(p_j)$

$C^s_{ij} = C_j(\sum_{k\ B_{ij}} p_k + \underline{t}^s_{ij})$ où $\underline{t}^s_{ij}$ désigne une borne inférieure sur la somme de temps d'exécution des $s-|B_{ij}|$ premiers éléments de D_{ij} dans toute séquence satisfaisant R ,

$C^n_{i,n+1} = 0$.

Alors toute séquence dans le réseau R' satisfait la relation R, et sa longueur dans ce réseau est inférieure ou égale à son coût total de retard dans le problème d'ordonnancement.

preuve: que toute séquence dans R' satisfasse la relation R découle immédiatement du théorème précédent. Soit donc w(1),w(2),...,w(n) une telle séquence; sa longueur est

$$\ell(w) = C^0_{0w(1)} + C^1_{w(1)w(2)} + \cdots + C^s_{w(s)w(s+1)} + \cdots + C^{n-1}_{w(n-1)w(n)}$$

et son coût est

$$C(w) = C_{w(1)}(p_{w(1)}) + C_{w(2)}(p_{w(1)}+p_{w(2)}) \cdots$$
$$+ C_{w(s+1)}(p_{w(1)}+p_{w(2)}+\cdots+p_{w(s)}+p_{w(s-1)}) + \cdots$$
$$+ C_{w(n)}(p_{w(1)}+p_{w(2)}\cdots+p_{w(n)}).$$

Or $C^0_{0w(1)} = C_{w(1)}(p_{w(1)})$

$C^1_{w(1)w(2)} = C_{w(2)}(p_{w(1)}+p_{w(2)})$

$C_{w(2)w(3)} = C_{w(3)}(\underline{t}^3_{w(2)w(3)}+p_{w(2)}+p_{w(3)}) \leq C_{w(3)}(p_{w(1)}+p_{w(2)}+p_{w(3)})$

d'après la définition de $\underline{t}^3_{w(2)w(3)}$ et le fait que $C_{w(3)}$ soit non-décroissante;

.....

$$C^{s}_{w(s)w(s+1)} = C_{w(s+1)}(\underline{t}^{s}_{w(s)w(s+1)} + p_{w(s)} + p_{w(s+1)})$$

$$\leqslant C_{w(s+1)}(p_{w(1)} + p_{w(2)} \cdots + p_{w(s-1)} + p_{w(s)} + p_{w(s+1)})$$

pour les mêmes raisons;

.....

$$C^{n-1}_{w(n-1)w(n)} = C_{w(n)}(p_{w(1)} + p_{w(2)} + \cdots + p_{w(n-2)} + p_{w(n-1)} + p_{w(n)})$$

puisque $\underline{t}^{n-1}_{w(n-1)w(n)}$ peut être calculé exactement.

d'où l'inégalité $\ell(w) \leqslant C(w)$. □

De ce lemme on déduit le théorème suivant qui caractérise les plus courts chemins associés à des pénalités u données, comme relaxations de type 3 pour le problème d'ordonnancement :

THEOREME 4-2 : Soient u des pénalités et p un plus court chemin dans le réseau R' relativement à u, et désignons par C* le coût d'une séquence optimale dans le problème d'ordonnancement, alors $\ell^{p} + u(a^{p} - \underline{1}) \leqslant C^{*}$.

preuve: ce théorème découle de l'inégalité du lemme précédent; en effet, soit s une séquence de coût C* minimal, on a

$$\ell^{p} + u(a^{p} - \underline{1}) \leqslant \ell^{s} \leqslant C^{*}$$

puisque $a^{s} = \underline{1}$. □

Corollaire 4-2: si, pour quelque pénalité u le plus court chemin obtenu est une séquence s et si $\ell(s) = C(s)$, alors s est une solution optimale du problème d'ordonnancement.

Remarque: la première condition n'est pas suffisante car cette relaxation est de type 3; le cas où l'on obtenait une séquence comme plus court chemin, mais qu'elle n'était pas une séquence optimale s'est rencontré plusieurs fois dans les expériences numériques reportées plus loin. Par conséquent, et au contraire du TDTSP, lorsque, dans un algorithme utili

sant cette relaxation pour le problème d'ordonnancement, le plus court chemin obtenu est une séquence, il est nécessaire de calculer le "vrai" coût de cette séquence; nous indiquerons dans le paragraphe consacré à la description d'un algorithme de Branch-and-Bound pour ce problème d'ordonnancement ce qu'il convient de faire lorsque ce coût et la longueur du chemin correspondant diffèrent.

La construction décrite dans le lemme 4-1 requiert les bornes inférieures $\underline{t}^s_{ij}$ mais ne spécifie pas de façon de les calculer. Pour cela, nous adopterons simplement la manière suivante: nous choisirons, parmi les éléments k appartenant à D_{ij} , les $s-|B_{ij}|$ qui ont les plus petits temps d'exécution p_k ; la somme de ces $s-|B_{ij}|$ plus petits temps d'exécutions fournit clairement une borne inférieure ,mais pas nécessairement une solution réalisable à cause de la relation R. Nous discuterons plus loin une autre façon possible permettant d'obtenir une meilleure borne inférieure en tenant compte de R.

-4 Réductions :

Nous allons présenter maintenant des réductions dans le réseau R' qui se sont révélées très efficaces : ces réductions ont permis de résoudre en moins de 50 secondes (CDC Cyber 74-18) des problèmes qui n'étaient pas résolus en 450 secondes.

Considérons une paire d'arcs ((i,s),(j,s+1)) et ((j,s),(i,s+1)) où i et j sont deux indices sans relation entre eux dans R (condition (i) du théorème 4-1). Nous allons donner des conditions suffisantes susceptibles de permettre l'élimination d'un de ces deux arcs (mais pas les deux), disons ((j,s),(i,s+1)), en montrant que toute séquence qui contiendrait j en $s^{ème}$ position et i en $(s+1)^{ème}$ pourrait être améliorée (au sens large) par l'échange des éléments adjacents i et j.

Désignons par $\overline{t}^s_{ij}$ une borne <u>supérieure</u> sur le temps total

d'exécution des $(s-|B_{ij}|)$ premières tâches de D_{ij} dans toute séquence ayant i en $s^{ème}$ position et j en $(s+1)^{ème}$ position. On peut calculer $\bar{t}^s_{ij}$ d'une manière comparable à $\underline{t}^s_{ij}$,c'est-à-dire en choisissant les $(s-|B_{ij}|)$ plus grands temps d'exécution dans D_{ij} . Remarquons que $\underline{t}^s_{ij} = \underline{t}^s_{ji}$

$$\text{et que } \bar{t}^s_{ij} = \bar{t}^s_{ji} \quad .$$

Désignons alors par $\underline{T}^s_{ij}$ et $\bar{T}^s_{ij}$ les quantités suivantes :

$$\underline{T}^s_{ij} = \sum_{k\in B_{ij}} p_k + \underline{t}^s_{ij} + p_j = \sum_{k\in B_{ji}} p_k + \underline{t}^s_{ji} + p_i$$

$$\bar{T}^s_{ij} = \sum_{k\in B_{ij}} p_k + \bar{t}^s_{ij} + p_j = \sum_{k\in B_{ji}} p_k + \underline{t}^s_{ji} + p_i$$

qui représentent respectivement des bornes inférieures et supérieures sur le temps d'exécution des (s+1) premières tâches dans toute séquence de ce type.

THEOREME 4-3:

si, pour tout $T \in \left[\underline{T}^s_{ij}, \bar{T}^s_{ij}\right]$ on a

$$C_j(T) - C_j(T-p_i) \leqslant C_i(T) - C_i(T-p_j) \qquad (4\text{-}4\text{-}1)$$

alors, pour toute séquence satisfaisant R et ayant j en $s^{ème}$ position et i en $(s+1)^{ème}$ position, on ne fait pas augmenter son coût en échangeant i et j.

preuve:

soit S une telle séquence; dans S, l'instant T auquel les (s+1) premières tâches sont terminées est dans l'intervalle $\left[\underline{T}^s_{ij}, \bar{T}^s_{ij}\right]$ par définition de cet intervalle (si s = 1, cet intervalle est réduit au seul point p_i+p_j); le gain réalisé dans l'échange de i et j est

$$(C_i(T-p_j) + C_j(T)) - (C_j(T-p_i) + C_i(T)) \leqslant 0 \ . \qquad \square$$

Evidemment, si la relation (4-4-1) est vérifiée avec égalité,

on n'éliminera qu'un seul des deux arcs; de plus on adoptera dans ce cas une règle de choix lexicographique, c'est-à-dire si $i \prec j$ on éliminera l'arc $((j,s),(i,s+1))$, afin d'assurer qu'il existe toujours au moins une séquence (optimale) dans le réseau R'.

Dans le cas du problème des retards pondérés ($C_i(t) = a_i \max\{0; t-d_i\}$) nous allons indiquer deux conditions suffisantes faciles à tester. Observons d'abord qu'une séquence, dans ce cas, peut être en général divisée en trois parties (dont une ou deux peuvent être éventuellement vides) : un "début" où toutes les tâches sont terminées sans retard, un "milieu" (de la première tâche en retard à la dernière tâche terminée à temps) et une "fin" constituée de tâches toutes en retard. Alors, si cette séquence est optimale, on peut ranger le début dans un ordre des dates prévues non-décroissantes(EDD "earliest due date", voir Baker [1974]) et la fin dans l'ordre WSPT ("weighted shortest processing time", ibid.). Les réductions que nous allons indiquer visent à réaliser ces rangements au début et à la fin du réseau R'.

Corollaire 4-3 :

si

(i) $d_i \leq d_j$ (4-4-2)

et (ii) $\bar{T}^s_{ij} \leq \min\{d_i; d_j - p_j\} + p_j$ (4-4-3)

alors on peut supprimer l'arc $((j,s),(i,s+1))$.

preuve: la relation (4-4-3) implique

$T - p_j \leq d_i$

et $T \leq d_j$ pour tout $T \leq \bar{T}^s_{ij}$. Donc ces deux tâches J_i et J_j sont exécutées sans retard si elles le sont selon l'ordre EDD spécifié par (4-4-2) □

Corollaire 4-4 :

si

(i) $p_i/a_i \leqslant p_j/a_j$ (4-4-4)

et (ii) $\underline{T}^s_{ij} \geqslant \max\{d_i + p_j \; ; \; d_j + p_i\}$ (4-4-5)

alors on peut supprimer l'arc $((j,s),(i,s+1))$.

preuve: la relation (4-4-5) implique que

$$T - p_j \geqslant d_i$$

et $T - p_i \geqslant d_j$ pour tout $T \geqslant \underline{T}^s_{ij}$ .Donc ces deux tâches J_i et J_j sont déjà en retard quand elles sont exécutées en $s^{ème}$ position; il convient donc de les ranger dans l'ordre WSPT spécifié par (4-4-4). □

On peut effectuer le calcul du réseau R' en incorporant ces réductions d'une façon efficace. La méthode adoptée consiste à trier au préalable les indices $k = 1,2,\ldots,n$ par ordre non-décroissant des temps d'exécution. On détermine ensuite, pour chacune des $n(n-1)/2$ paires (i,j) si l'un des éléments précéde l'autre (dans ce cas on ne construira qu'un seul arc pour chaque valeur de s permise, et l'on n'aura pas à appliquer de réduction) ou bien si ces éléments n'ont aucune relation dans R; on calcule ensuite $s = |B_{ij}|$ et $\sum_{k\ B_{ij}} p_k + p_j$ puis on introduit un à un les éléments k ($\in D_{ij}$) dans B_{ij} l'ordre des temps d'exécution non-décroissants (et aussi non-croissants au début, pour le test du corollaire 4-3), en effectuant les tests définis par les corollaires 4-3 puis 4-4. On peut vérifier que, en s'y prenant de la sorte, le nombre d'opérations élémentaires impliquées dans le calcul du réseau R' est en $O(n^3)$.

-5 Un algorithme de Branch-and-Bound pour le problème d'ordonnancement :

Même avec les réductions précédentes, il se peut que l'on ne puisse obtenir de séquence optimale (par exemple si la solution entière n'est pas optimale pour le problème (LP2) dans le réseau R'). On a alors recours à un algorithme de Branch-and-Bound essentiellement similaire à celui exposé au chapitre 2. Nous ne soulignerons donc ici que les différences essentielles.

Afin d'obtenir des bornes aussi bonnes que possible, le réseau R' est recalculé, selon la procédure indiquée au paragraphe précédent, à chaque noeud de l'énumération. On peut utiliser les meilleures pénalités du problème père pour les sous-problèmes fils grâce aux relations (2-5-2); on obtient ainsi dès le début des bornes implicites meilleures que la borne correspondante dans le problème père. Mais d'autre part, la relation de précédence n'est pas recalculée à chaque noeud de l'énumération mais simplement déduite de la relation R initiale; la raison en est que ce calcul est trop coûteux (d'une complexité qui peut excéder $O(n^3)$). Toutefois, et avant même de calculer le réseau R', on effectue à chaque noeud d'abord le test d'Elmaghraby (voir Baker [1974]) qui peut permettre de fixer éventuellement en dernière position une ou plusieurs tâches.

On a déjà souligné plus haut que, lorsqu'on trouve une séquence S comme plus court chemin, il convient de calculer son coût exact C(S) et de le comparer à sa longueur $\ell(S)$ dans R' : dans le cas où $\ell(S) < C(S)$, on sait qu'on ne peut pas obtenir de meilleure borne par le réseau R' (puisque $\ell(S) = z^*$) et il faut brancher à partir de là; si $\ell(S) = C(S)$ on peut remonter car le sous-problème (d'ordonnancement) est résolu. Auparavant, et indépendamment du résultat du test, on peut comparer C(S) à $\bar{z}$

et S peut éventuellement devenir la meilleure séquence connue.

Enfin, le test de dominance peut être réalisé de façon plus efficace pour ce problème d'ordonnancement que pour le TDTSP, en $O(nk^2)$ opérations au plus. En outre, et puisque l'algorithme s'adresse au problème d'ordonnancement et non à un TDTSP, on peut remarquer que le coût associé à l'exécution d'une tâche J_i en $(n-k)^{ème}$ position est indépendant de la tâche exécutée en $(n-k-1)^{ème}$ position, et l'on peut donc chercher à insérer des segments commençant par chacune des n-k tâches J_i non fixéees dans la partie fixée de la solution (positions n-k+1 à n).

5 Comparaison avec une autre approche :

Avant notre étude, la méthode la plus efficace pour résoudre les problèmes d'ordonnancement de ce type était celle de Rinnooy Kan et al. [1975] (que nous désignerons ci-après par RKLL). Cette méthode repose sur un algorithme de Branch-and-Bound essentiellement semblable au notre pour ce qui concerne le principe de séparation, mais qui en diffère pour la relaxation utilisée et aussi par le fait que notre algorithme utilise un test de dominance

L'importance du test de dominance dans notre algorithme a été bien démontrée lors de la mise au point de cet algorithme : certains problèmes qui requèraient environ 50 secondes pouvaient être résolus en 10 secondes par simple addition du test de dominance (nous n'avons pas effectué de comparaisons systématiques à cause du coût de calcul impliqué - dans les problèmes sur lesquels cette comparaison a pu être faite, l'avantage du test de dominance, toujours positif, croissait considérablement avec le nombre de sous-problèmes explorés). Il est possible d'incorporer

ce test de dominance à l'algorithme de RKLL, et c'est notre conviction qu'on doit réaliser par là une amélioration sensible sur l'algorithme original.

Il est possible de comparer la borne fournie par la relaxation utilisée par RKLL avec celle obtenue par le TDTSP. Cette borne inférieure, que nous désignerons par z_{RKLL} est la valeur optimale d'une solution d'un problème d'affectation dont les coefficients a_j^s (RKLL utilisent une autre notation) représentent un minorant sur la valeur $C_i(t_i)$ pour toute séquence compatible avec R et dans laquelle la tâche J_i est exécutée en $s^{ème}$ position; ce minorant est obtenu plaçant d'abord toutes les tâches J_j telles que jRi (c'est-à-dire dans B_i), puis en remplissant les positions restantes avant la $s^{ème}$ par des tâches sans relation avec i et de plus petits temps d'exécution. En comparant ce procédé avec la façon dont sont définis les C_{ij}^s ,on obtient immédiatement l'inégalité

$$a_i^s \leqslant \min_j C_{ij}^s \qquad \text{pour tous } i,s \qquad (4\text{-}5\text{-}1).$$

Si l'on considère alors la relaxation (LAR) du TDTSP défini sur le réseau R', on obtient une relaxation pour le problème d'ordonnancement, qui est également un problème d'affectation et l'on a

$$z_{RKLL} \leqslant z_{LAR} \leqslant z_{LP1} \leqslant C^* \qquad (4\text{-}5\text{-}2)$$

d'après (4-5-1) et le théorème 2-1. Il convient de remarquer que, en fait, la borne z_{LP1} n'est en général pas évaluée exactement et qu'on se contente d'une borne inférieure qui pourrait être plus mauvaise que z_{RKLL} ; mais d'autre part, d'après le théorème 2-4, des pénalités fournissant une borne au moins aussi bonne que z_{LAR} sont accessibles en résolvant d'abord z_{LAR}. En outre, la borne effectivement utilisée dans notre algorithme est $\min_i B_i$, la plus petite borne implicite et cette valeur peut bien être même meilleure que z_{LP1}.

Il ya d'autres raisons pour considérer que la borne inférieure que nous utilisons est plus précise. En effet, on utilise, pour le calcul des C_{ij}^{s} des séquences initiales (positions 1 à s) qui sont beaucoup plus diversifiées, et tiennent mieux compte de la relation de précédence, que celles utilisées pour le calcul des a_i^s ; en effet, d'une part, on s'efforce d'évaluer explicitement ce qu'il en coûte de placer certaines tâches J_j immédiatement avant certaines tâches J_i et les ensembles B_{ij} permettent de tenir compte de manière plus précise de la relation R que les B_i seuls. En d'autres termes, on dispose de plus de latitude pour tenter de bouleverser l'ordre SPT ("shortest processing times", temps d'exécution non-décroissants) qui tend à s'imposer dans le calcul des minorants.

Mais il faut se garder, on l'a vu au chapitre 3, de ne considérer que la valeur de la borne utilisée pour évaluer un algorithme de ce type; le temps de calcul, et plus précisément les possibilités de déduire facilement les solutions des sous-problèmes de la solution des problèmes précédents (post-optimisation) jouent un rôle également important. La plus convaincante validation de notre approche provient des résultats des expériences numériques en comparaison avec ceux rapportés par RKLL.

Expériences numériques:

L'algorithme décrit dans les paragraphes précédents a été programmé en FORTRAN IV et évalué, sur le CDC Cyber 74-18 de l'université de Montréal, sur un ensemble de 84 problèmes tests à 15 et 20 tâches pour le problème des retards pondérés.

L'algorithme est précédé par une méthode heuristique basée sur les insertions d'éléments, voir paragraphe 1-7 . Les solutions de départ utilisées étaient fournies par six méthodes de rangement: SPT, WSPT, EDD ("earliest due-date", dates prévues non-

décroissantes), poids a_i dans un ordre non-croissant, méthode du rapport de Montagne (MRM voir Baker [1974]) et une nouvelle méthode constructive que nous appelerons "méthode du taux de retard", qui consiste à ranger les tâches dans un ordre non-décroissant des coefficients

$$q_i = t(p_i/a_i) + (1-t)d_i$$

où t désigne le"taux de retard"(voir Baker [1974])

$$t = 1 - (\sum_{i=1}^{n} d_i)/n(\sum_{i=1}^{n} p_i)$$

(cette méthode peut être justifiée par les observations suivantes (voir RKLL p.922): si toutes les tâches sont en retard, donc $t = 1$, la séquence WSPT est optimale et si toutes les tâches peuvent être terminées à temps, d'où $t = 0$, la séquence EDD est optimale). Cette dernière méthode s'est avérée la plus efficace des méthodes constructives, mais nettement moins que celle qui consiste à améliorer par des insertions d'éléments à partir de l'une des six séquences construites. En fait, choisir la meilleure des six séquences améliorées fournit une méthode heuristique très efficace pour les données que nous avons testées. Le temps d'exécution pour cette méthode (améliorations par insertions d'éléments à partir de six méthodes constructives) est à peu près constant pour une valeur de n fixée; les temps observés ont été de .5 sec. pour $n = 15$ et de 1.1 sec. pour $n = 20$.

Les données pour les 42 premiers problèmes tests sont celles de RKLL et correspondent aux problèmes les plus difficiles pour leur méthode, soit ceux avec un taux de retard de .6 et .8 pour 15 tâches,et .4 , .6 et .8 pour 20 tâches. Les temps d'exécution et le nombre de noeuds explorés dans le Branch-and-Bound sont donnés en tables 1 et 2 pour la méthode de RKLL (CDC Cyber 73-

28) et pour notre algorithme (CDC Cyber 74-18). Etant donné que le Cyber 74 est considéré comme étant de 2.5 à 3.5 fois plus rapide que le Cyber 73 (estimation fournie par les ingénieurs-système CDC), il apparaît que notre algorithme est nettement plus efficace que celui de RKLL, au moins pour les problèmes difficiles. Nous n'avons pas effectué de tests pour des valeurs plus faibles de n ou t à cause des faibles temps de calcul impliqués et du fait que notre algorithme est précédé par cette méthode heuristique qui requiert un temps à peu près constant; pour ces problèmes faciles, un algorithme de programmation dynamique exploitant la relation de précédence (Srinivasan [1971] , Baker [1975] , [1976]) est sans doute le meilleur choix.

Le second ensemble de 42 problèmes concerne le problème du retard total minimum ("total tardiness problem" ou encore "mean tardiness problem"); les données sont identiques à celles des 42 premiers problèmes, sauf que tous les poids ont été fixés à 1. Bien que de nombreuses simplifications soient possibles dans ce cas, nous avons effectué ces tests avec exactement le même algorithme que pour le premier ensemble de problèmes tests afin de pouvoir étudier son comportement sur ce cas particulier. Les résultats, temps d'exécution et nombre de noeuds dans le Branch-and-Bound, sont donnés dans la table 3. Il apparaît clairement d'après ces résultats que les problèmes avec poids sont plus difficiles .

Nous n'avons pas effectué de comparaisons directes avec les résultats obtenus par Fisher [1976] car nous avons obtenu les données correspondant à ses problèmes-tests alors que n'avions plus accès à ces moyens de calcul (il n'est pas certain non plus que les machines utilisées soient facilement comparables). Nous pouvons toutefois indiquer deux choses: d'abord la relaxation

utilisée par Fisher a un temps de calcul proportionnel au temps d'exécution total et les résultats qu'il donne sont pour des problèmes dont les temps d'exécution p_i sont compris entre 1 et 10, alors que nos problèmes tests ont des temps d'exécution généralement compris entre 50 et 150 (Fisher indique bien la possibilité de "changer d'échelle" les temps d'exécution mais, à notre connaissance, cette possibilité n'a jamais été testée). D'autre part, Fisher rapporte certaines comparaisons entre son algorithme, exécuté sur un IBM 360-67 et celui de RKLL sur un CDC Cyber 73-38 et précise "généralement, leur temps d'exécution ont été à peu près égaux aux notres (Fisher) pour les problèmes à 20 tâches". Il semble donc, d'après ces indications, que notre algorithme soit plus efficace que celui de Fisher, au moins pour les problèmes à 20 tâches.

Récemment, K.R.Baker nous a communiqué les résultats obtenus par un"algorithme des chaînes" qu'ila développé avec L.Schrague (Baker [1975], [1976]) sur le même premier ensemble de 42 problèmes de retard pondéré; chacun de ces problèmes est résolu en moins de 1.2 seconde sur un IBM 370-168 par cet algorithme. Il semble que cette version de la programmation dynamique conçue pour tirer parti de la relation de précédence est actuellement le meilleur algorithme pour cette classe de problèmes.

n	t	nombre de problèmes	nouvel algorithme* moyen	median	maximum	R.K.L.L. $ median	maximum
15	.6	12	1.9	1.6	5.9	6.3	121.8
15	.8	12	1.7	1.6	3.4	45.6	85.6
20	.4	6	2.7	1.5	6.4	1.1	20.3
20	.6	6	13.6	6.5	30.7	180.8	300.
20	.8	6	12.0	11.0	20.8	300.	300.

* CDC Cyber 74-18 $ CDC Cyber 73-28

Table 1. Temps de solution pour les problèmes de retards pondérés

n	t	nombre de problèmes	nouvel algorithme moyen	median	maximum	R.K.L.L. median	maximum
15	.6	12	29.5	17.	121.	647.	9564.
15	.8	12	28.0	24.	57.	4532.	9952.
20	.4	6	11.3	1.	34.	25.	1206.
20	.6	6	118.8	48.	302.	11105.	--
20	.8	6	136.6	120.	327.	--	--

Table 2. Nombre de noeuds énumérés pour les problèmes de retards pondérés.

n	t	nombre de problèmes	temps de solution (sec. CPU, CDC Cyber 74-19) moyen	maximum	nombre de noeuds dans l'énumération moyen	maximum
15	.6	12	.8	1.	9.	15.
15	.8	12	.7	1.	12.4	43.
20	.4	6	1.1	1.6	5.5	14.
20	.6	6	2.7	5.7	21.	52.
20	.8	6	3.7	12.8	38.8	168.

Table 3. Résultats pour les problèmes de retard total ($a_i = 1$ pour tous i)

-8 Améliorations possibles :

L'algorithme que nous avons présenté, et le programme que nous avons écrit pour le problème des retards pondérés, peuvent être améliorés de plusieurs façons.

On peut d'abord chercher à raffiner la relation de précédence à chaque noeud du Branch-and-Bound. On peut aussi étendre les réductions, introduites par le Théorème 4-3, à des cas qui ne sont pas traités par les corollaires 4-3 et 4-4, c'est-à-dire dans le "milieu" du réseau R'. En outre, on peut aussi utiliser l'algorithme PCC2 dans ce réseau; la raison pour laquelle nous avons adopté l'algorithme PCC est sa simplicité d'application en rapport avec la façon dont le réseau R' est conservé en mémoire.

On peut obtenir de meilleures bornes en tenant mieux compte de la relation de précédence dans le calcul des longueurs des arcs C^s_{ij} dans le réseau R' si l'on peut calculer les valeurs $\underline{t}^s_{ij}$ avec plus de précision. Par exemple, pour $s = |B_{ij}| + 1$, la meilleure valeur possible pour $\underline{t}^s_{ij}$ s'obtient en choisissant, parmi les tâches dans D_{ij} qui n'ont aucun prédécesseur dans D_{ij} une tâche de plus petit temps d'exécution. Pour des valeurs plus grandes de s, une telle sélection devient un problème très difficile (en fait NP-complet) mais il est très possible que l'on puisse obtenir des bornes inférieures plus précises à un coût de calcul additionnel raisonnable.

A coté des tentatives précédentes pour obtenir de meilleures bornes, on peut également modifier la structure du Branch-and-Bound. Ainsi, on peut étendre les bornes implicites à l'avant-dernière période car les longueurs C^{n-1}_{ij} représentent exactement la contribution de la tâche j au coût total si elle est exécutée

en dernière position; par conséquent on peut utiliser un principe de séparation qui fixe d'un seul coup les deux dernières tâches; on évite ainsi d'avoir à recalculer le réseau R' et à effectuer les itérations de plus court chemin pour les sous-problèmes de niveau intermédiaire (n-1,n-3,...). Cette idée peut évidemment s'étendre aux trois dernières positions, et plus généralement aux r dernières positions; il existe sans doute une valeur limite pour r à partir de laquelle il devient beaucoup trop difficile de traiter les r-tuples à placer dans les r dernières positions mais il n'est pas du tout certain que la valeur r = 1 actuellement utilisée soit la meilleure possible.

EN GUISE DE CONCLUSION . . .

Nous avons présenté une méthode de solution pour le TDTSP, ainsi que deux de ses applications. Alors que, dans le cas du problème du voyageur de commerce, application directe, les résultats numériques étaient assez décevants, il est apparu que cette approche pouvait être intéressante dans le cas de l'application indirecte au problème d'ordonnancement. Nous allons indiquer quelques autres applications indirectes pour lesquelles le TDTSP pourrait s'avérer utile.

Au problème d'affectation quadratique (défini au § 2-1), on peut associer une relaxation par le TDTSP en définissant les longueurs C^s_{kj} des arcs dans le réseau multiparti, d'une façon semblable à la définition des b_{js} (cf. § 2-2), mais avec la restriction supplémentaire $x_{k,s-1}=1$. Il faut toutefois noter que, pour ce problème, le test de dominance ne peut pas s'appliquer.

Le problème de rangement linéaire est un cas particulier du problème d'affectation quadratique, dans lequel les activités doivent être localisées en des points situés sur une même droite (problème à une dimension). Dans ce cas, le calcul des longueurs des arcs est simplifié et le test de dominance peut s'appliquer.

Le problème de la détermination d'un ordre à distance minimale d'une relation binaire donnée est aussi un cas particulier du problème d'affectation quadratique, pour lequel le calcul des longueurs des arcs est également simplifié et le test de dominance s'applique aussi. On peut également définir une autre relaxation par le TDTSP en tenant compte à la fois des arcs de la relation binaire incidents à k et à j dans la définition des C^s_{kj} ; ces longueurs peuvent encore être obtenues par une simple méthode de rangement.

A coté de ces quelques exemples de problèmes de permutation optimale, on peut généraliser l'approche décrite au chapitre 3 au cas de plusieurs véhicules; le réseau associé n'est alors plus multiparti car les différentes tournées se terminent, en général, avant d'avoir visité toutes les villes. Des problèmes de collecte et de distribution peuvent être modélisés ainsi, en introduisant les contraintes de capacité des véhicules dans les sous-problèmes de cheminement.

De même, des problèmes d'ordonnancement sur machines parallèles peuvent être traités en généralisant de la même façon l'approche décrite au chapitre 4 ; on pourrait aussi aborder de la sorte des problèmes d'atelier ("flow-shop, job-shop scheduling"), mais ceci nécessiterait une étude plus approfondie.

Si certains résultats encourageants ont déjà pu être obtenus, il nous apparaît qu'il reste beaucoup à faire, tant pour améliorer les méthodes de solution (ou en proposer de nouvelles) que pour tenter de cerner le domaine d'application de l'approche que nous avons présentée ici.

REFERENCES

Aho A., Hopcroft J. and Ullman J.

1974- "The design and analysis of computer algorithms". Addison-Wesley.

Baker K.R.

1974- "Introduction to sequencing and scheduling". Wiley.

1975- "Finding an optimal sequence by dynamic programming: an extension to task chains". GSBA Paper 145, Graduate School of Business Administration, Duke University, Durham NC (USA).

1976- "Computational experience with a sequencing algorithm adapted to the tardiness problem". GSBA Paper 163, Graduate School of Business Administration, Duke University, Durham NC (USA).

Bazaraa M.S. and Goode J.J.

1976- "The traveling salesman problem: a duality approach". Georgia Institute of Technology, Atlanta Ga (USA).

Bellman R.

1958- "On a routing problem". Quart. Appl. Math.16,pp. 87-90.

Bellmore M. and Malone J.

1971- "Pathology of TSP subtour elimination algorithms". Operations Research 19, pp. 278-307.

Bellmore M. and Nemhauser G.L.

1968- "The traveling salesman problem: a survey". Operations Research 16, pp. 538-558.

Camerini P.M., Fratta L. and Maffioli F.

1973- "A heuristically guided algorithm for the traveling salesman problem".
Journal of the Institution of Computer Science 4,2, pp.31-35.

1975- "On improving relaxation methods by modified gradient techniques".
Mathematical Programming Study 3, pp.26-34.

Chen S. and Saigal R.

1977- "A primal algorithm for solving a capacitated network flow problem with additional linear constraints".
Networks 7, pp.59-79.

Christofides N.

1970- "The shortest hamiltonian chain of a graph".
S.I.A.M. Journal on Applied Mathematics 19, pp. 689-696.

Conway R.W., Maxwell W.L. and Miller L.W.

1967- "Theory of scheduling".
Addison-Wesley.

Cunningham W.H.

1976- "A network simplex method".
Mathematical Programming 11, pp.105-116.

Dantzig G.B., Fulkerson D.R. and Johnson S.M.

1954- "Solution of a large scale traveling traveling salesman problem".
Operations Research 2, pp.393-410.

Dijkstra E.W.

1959- "A note on two problems in connexion with graphs".
Numerische Mathematik 1, pp. 269-271.

Edmonds J. and Karp R.M.

1971- "Theoretical improvements in algorithmic efficiency for network flow problems".
Journal of the A.C.M. 19,2, pp. 248-264.

Emmons H.

1969- "One-machine sequencing to minimize certain functions of job tardiness".
Operations Research 17, pp.701-715.

Fisher M.J.

1976- "A dual algorithm for the one-machine scheduling problem".
Mathematical Programming 11, pp.229-251.

Fisher M.L., Northup W.D. and Shapiro J.F.

1975- "Using duality to solve discrete optimization problems: theory and computational experience".
Mathematical Programming Study 3, pp. 56-94.

Fox K.R.

1973- "Production scheduling on parallel lines with dependencies".
Ph.D. Dissertation, The Johns Hopkins University, Baltimore Md (USA).

Garfinkel R. and Nemhauser G.L.

1972- "Integer programming".
Wiley.

Geoffrion A.M.

1974- "Lagrangian relaxation for integer programming".
Mathematical Programming Study 2, pp. 82-114.

Gilsinn J. and Witzgall C.

1973- "A performance comparison of labeling algorithms for calculating shortest path trees".
NBS Technical Note 777, National Bureau of Standards Washington D.C. (USA).

Goffin J.L.

1976- "On convergence rate of subgradient optimization methods". Working paper 76-34, Faculty of Management, McGill University, Montréal Qué. (Canada).

Hadley G.

1964- "Nonlinear and dynamic programming". Addison-Wesley.

Hardgrave W.W. and Nemhauser G.L.

1962- "On the relation between the traveling salesman and the longest path problem". Operations Research 10, pp. 647-657.

Held M. and Karp R.M.

1970- "The traveling salesman problem and minimum spanning trees". Operations Research 18, pp. 1138-1162.

1971- "The traveling salesman problem and minimum spanning trees: part II". Mathematical Programming 1, pp. 6-25.

Houck D.J. and Vemuganti R.R.

1976- "The traveling salesman problem and shortest n-paths". ORSA/TIMS Meeting, Spring 1976, Philadelphia Pa (USA).

Karp R.M.

1972 "Reducibility among combinatorial problems", in Miller R.E. and Thatcher J.W. (editors), "Complexity of computer computations". Plenum Press.

Klingman D. and Russell R.

1975- "Solving constrained transportation problems". Operations Research 23, pp. 91-106.

Knuth D.

1973- "The art of computer programming III: Sorting and searching". Addison-Wesley.

Kruskal J.B.

1956- "On the shortest spanning subtree of a graph and the traveling salesman problem".
Proceedings of the American Mathematical Society 2, pp. 48-50.

Lawler E.L.

1963- "The quadratic assignment problem".
Management Science 19, pp. 586-599.

Lemarechal C.

1975- "An extension of Davidon methods for non-differentiable problems".
Mathematical Programming Study 3, pp. 95-109.

Lin S.

1965- "Computer solution of the traveling salesman problem".
Bell System Technical Journal 44, pp. 2275-2269.

Little J.D., Murty K.G., Sweeney D.W. and Karel C.

1963- "An algorithm for the traveling salesman problem".
Operations Research 11, pp. 979-989.

Marsten R.E. and Morin T.L.

1976- "Branch and bound strategies for dynamic programming".
Operations research 24, pp.611-627.

Matthys G.

1961- "Flot optimum dans un reseau à capacités de faisceaux".
Actes du 2e congrès international de recherche opérationnelle, Aix-en-Provence (septembre 1960). Dunod.

Nemhauser G.L.

1972- "A generalized permanent label setting algorithm for the shortest path between specified nodes".
Journal of Math. Anal. and Appl. 38, pp. 328-334.

Padberg M.W. and Rao M.R.

1974- "The traveling salesman problem and a class of polyhedra of diameter two".
Mathematical Programming 7, pp. 32-45.

Reiter S. and Sherman G.

1965- "Discrete optimizing"
S.I.A.M. Journal 13, pp. 864-889.

Rinnooy Kan A.H.G., Lageweg B.J. and Lenstra J.K.

1975- "Minimizing total costs in one-machine scheduling".
Operations Research 23, pp. 908-927.

Rosseel M.

1968- "Comments on a paper by R.Saigal".
Operations Research 16, pp. 1232-1234.

Roy B.

1970- "Algèbre moderne et théorie des graphes".
Dunod.

Saigal R.

1968- "A constrained shortest route problem".
Operations Research 16, pp. 388-401.

Smith T.H.C., Srinivasan V. and Thompson G.L.

1975- "Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems".
Management Sciences Research Report 369, Carnegie-Mellon University, Pittsburgh Pa. (USA).

Smith T.H.C. and Thompson G.L.

1975- "A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation". Management Sciences Research Report 356, Carnegie-Mellon University, Pittsburgh Pa. (USA).

Srinivasan V.

1971- "A hybrid algorithm for the one-machine sequencing problem to minimize total tardiness".
Naval Research Logistics Quarterly 18, pp.317-327.

Srinivasan V. and Thompson G.L.

1973- "Solving scheduling problems by applying cost operators to assignment models",
in Elmaghraby S.E. (editor) "Symposium on the theory of scheduling and its applications".
Springer-Verlag (New York).

Tomizawa N.

1971- "On some techniques useful for solution of transportation network problems".
Networks 1, pp. 173-194.

Wagner H.M.

1970- "Principles of management science".
Prentice-Hall.

Wolfe P.

1975- "A method of conjugate subgradients for minimizing nondifferentiable functions".
Mathematical Programming Study 3, pp. 145-173.