Titre: Title:	Méthode d'optimisation pour la planification de la production dans une mine à ciel ouvert
Auteur: Author:	Beyime Ould Tachefine
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ould Tachefine, B. (1997). Méthode d'optimisation pour la planification de la production dans une mine à ciel ouvert [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8939/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8939/
Directeurs de recherche: Advisors:	François Soumis
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

MÉTHODE D'OPTIMISATION POUR LA PLANIFICATION DE LA PRODUCTION DANS UNE MINE À CIEL OUVERT

BEYIME TACHEFINE

DÉPARTEMENT DE MATHÉMATIQUES

ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)

(MATHÉMATIQUES DE L'INGÉNIEUR)

JANVIER 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

MÉTHODE D'OPTIMISATION POUR LA PLANIFICATION DE LA PRODUCTION DANS UNE MINE À CIEL OUVERT

présentée par: TACHEFINE Beyime

en vue de l'obtention du diplôme de: Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de:

M. SMITH Benjamin T., PH.D., président du jury

M. SOUMIS François, Ph.D., membre et directeur de recherche

M. DESROSIERS Jacques, Ph.D., membre

M. GOFFIN Jean-Louis, Ph.D., membre externe

À mes parents, Sidi et Ziwana

À Hassen et Mettou

REMERCIEMENTS

Je remercie mon directeur de recherche, François SOUMIS, pour avoir accepté de m'encadrer dans mes recherches. Sa confiance et son soutien continu ont rendu possible la réalisation de ce travail.

Je tiens aussi à remercier M. Benjamin T. SMITH de me faire l'honneur de présider mon jury, ainsi que M. Jean-Louis GOFFIN et M. Jacques DESROSIERS pour avoir accepté de faire partie du dit jury.

Je remercie tous ceux, qui de près ou de loin ont contribué à la réalisation de ce travail, je pense particulièrement à M. Claude LEMARECHAL qui m'a initié à la méthode des faisceaux et M. KIWIEL avec lequel j'ai eu des discussions fructueuses.

Enfin je remercie mes collègues du GERAD qui ont rendu mon séjour des plus agréables. De même je remercie tous les membres de la communauté Mauritanienne de Montréal pour leur fraternité éternelle.

RÉSUMÉ

Le thème central de cette thèse est le problème de planification multi-périodes de l'exploitation d'une mine à ciel ouvert. Ce problème de très grande taille comporte plus d'un million de variables binaires et plus de 10 millions de contraintes. Aucune méthode de résolution exacte, ni avec erreur mesurable n'a été proposée jusqu'à maintenant pour ce problème. Le développement d'une bonne approche de résolution à ce problème est d'une importance capitale pour les exploitants miniers. Elle leur permettra de disposer d'un outil de prise de décision capable de fournir des plans d'exploitation réalistes qui tiennent compte de toutes les contraintes d'exploitation et qui maximisent le profit des opérations. Notons qu'une économie de 1% sur les opérations annuelles d'une exploitation minière se chiffre généralement en millions de dollars.

Une formulation de ce problème est présentée au chapitre 1 de cette thèse. Ce chapitre présente, en plus de la formulation, une revue de la littérature sur ce sujet et met en évidence la structure particulière de ce problème. Plusieurs problèmes connus par les exploitants miniers sont des cas particuliers du problème de planification multi-périodes. Le problème des contours finaux abordé dans cette thèse comme un problème de fermeture maximale sur un graphe est obtenu en considérant un problème de planification réduit à une seule période sans contraintes de ressource. De même le problème de design de fosse avec contraintes de ressource correspond à un problème d'une seule période. Dans cette thèse, nous avons abordé ces différents problèmes afin de développer une méthode efficace pour la résolution du problème de planification multi-périodes.

Le problème de la fermeture maximale sur un graphe, connu chez les exploitants miniers sous le nom du problème des contours finaux, se résout à l'aide d'un algorithme de flot maximum. Comme les graphes en jeu sont de très grande taille, l'utilisation d'un algorithme performant est grandement recherché. Le chapitre 2 de cette thèse présente une étude comparative des algorithmes de flot maximum les plus prometteurs. Des améliorations pratiques à ces algorithmes tenant compte de la nature de nos graphes sont proposées. Des résultats sur des graphes tests générés aléatoirement et d'autres obtenus sur les données d'un gisement en cours d'exploitation sont présentés. Cette étude nous a permis de sélectionner l'algorithme des pré-flots utilisant une stratégie de sélection des nœuds actifs selon la plus grande distance (en anglais :*Highest Label Preflow Algorithm*) pour la résolution de ce problème. Avec cet algorithme, on est arrivé à une implémentation qui résout le problème de fermeture maximale sur un graphe de 150 000 nœuds et de l'ordre de 1,5 millions d'arcs en moins de 15 secondes CPU sur une station de travail HP9000/735.

Le problème de design de fosses avec contraintes de ressource consiste à déterminer une fermeture maximale sur un graphe en présence d'un ensemble de contraintes de ressource. Pour sa résolution, nous proposons la relaxation lagrangienne des contraintes de ressource afin de se ramener à un problème de fermeture maximale classique déjà traité au chapitre 2. Nous ajustons les multiplicateurs de Lagrange par la méthode des faisceaux qui s'est avérée la méthode la plus appropriée comparativement à celle du sous-gradient ou à celle des plans sécants. Cette approche nous fournit une borne supérieure pour la solution optimale du problème mais elle ne produit pas toujours des solutions primales réalisables du problème. Ainsi pour renforcer cette approche, nous lui avons incorporée une heuristique simple qui permet dans une première phase de convertir en une solution primale réalisable toute

solution primale obtenue lors de la résolution du problème lagrangien mais ne satisfaisant pas les contraintes de ressource. La deuxième phase de cette heuristique utilise une recherche tabu simple pour améliorer la valeur de la solution réalisable obtenue à la fin de la première phase. La valeur de la solution réalisable retenue constitue une borne inférieure pour la solution optimale. La résolution de ce problème de planification réduit à une seule période est détaillée au chapitre 5 intitulé *"Fermeture maximale sur un graphe avec contraintes de ressource"*. Les tests, présentés dans ce chapitre, ont été réalisés sur des graphes de tailles moyennes (jusqu'à 10 000 nœuds et 100 000 arcs). Les résultats obtenus montrent des gaps très petits et des temps de calcul raisonnables. En effet, pour ajuster les multiplicateurs de Lagrange, il suffit de résoudre moins de 10 fois le problème de fermeture maximale classique. Cette approche de résolution est la première qui réussit à traiter des problèmes de grande taille semblables à ceux rencontrés en pratique. Elle innove par rapport aux tentatives connues dans la littérature par l'utilisation d'une méthode performante d'ajustement des multiplicateurs de Lagrange et par l'utilisation d'un algorithme adapté pour la résolution des sous-problèmes.

Les développements du chapitre 5 sont à la base du schéma de traitement du problème de planification multi-périodes. Pour traiter ce dernier, nous résolvons les problèmes correspondant à chacune des périodes dans l'ordre croissant des périodes en utilisant l'approche présentée au chapitre 5 où l'heuristique est modifiée de façon à obtenir des solutions primales compatibles avec celles retenues lors des périodes précédentes. Les détails de cette approche de résolution du problème multi-périodes de la planification dans une mine à ciel ouvert sont présentés au chapitre 6 intitulé *" Fermeture maximale sur un réseau multi-périodes en présence de contraintes de ressource: application à la planification minière"*. Les résultats obtenus par cette approche sur des données réelles sont excellents. Sur des gisements de 150 000 blocs

avec un horizon de 10 périodes et 3 contraintes de ressource par période, elle permet d'obtenir des solutions réalisables avec des gaps de l'ordre de 1%. Le temps machine requis pour la résolution des problèmes de cette taille, comportant 1 500 000 variables et 15 millions de contraintes, est de moins de 6 heures sur une station de travail HP9000/735. Les tests sur des problèmes de différentes tailles générés aléatoirement sont concluants: les gaps pour ces problèmes varient de 0.1% à 3.0%. Parmi les problèmes aléatoires, nous avons identifié les caractéristiques d'une classe de problèmes présentant de petits gaps; les problèmes miniers possèdent généralement ces caractéristiques. Nous avons suggéré des améliorations pouvant mener à une réduction des gaps pour les autres problèmes qui n'étaient pas l'objectif principal de cette thèse. Cette méthode de résolution à erreur mesurable est la première méthode pouvant traiter efficacement les problèmes de planification multi-périodes de grande taille dans toute leur complexité.

ABSTRACT

This thesis mainly studies the multi-period production scheduling problem in an open pit mine. This large-scale problem involves more than one million binary variables and more than 10 million constraints. No exact method nor a method with measurable error has been developed to solve this problem until now. The development of a good solution approach would allow miners to have a decision making tool. This tool will enable the construction of an exploitation plan which takes into account all production constraints and maximizes profits. Note that a saving of 1% on annual operations in a mine is generally estimated at millions of dollars.

A mathematical formulation for this problem is presented in Chapter 1. This chapter also presents a review of the subject and underlines the particular structure of this problem. Many problems known by miners are particular cases of the global multi-period problem. The open pit design problem which is treated in this thesis as a maximal closure problem on a graph is obtained from the multi-period problem by considering one period without resource constraints. The open pit design with resource constraints is also a multi-period problem reduced to one period. In this thesis, all these particular problems are treated in order to develop an efficient solution method for the multi-period problem.

The open pit design problem which is a maximal closure problem is solved using a maximal flow algorithm. As the graphs involved in mine applications are very large, the use of an efficient maximal flow algorithm is widely sought. Chapter 2 of this thesis presents a comparative study of the best maximal flow algorithms. Practical improvements to these algorithms based on a particular structure of our

graphs are also proposed. Numerical results obtained from graphs randomly generated and others obtained from a mine deposit under exploitation are presented in this chapter. This study allows us to select the preflow algorithm using the selection strategy of active nodes wich have the highest distance labels (*Highest Label Preflow Algorithm*). With this algorithm, the maximal closure problem is solved on graphs involving 150 000 nodes and around 1,5 million arcs in less than 15 seconds CPU.

The open pit design with resource constraints problem consists in determining a maximal closure on a graph with supplementary resource constraints. For the solution of this problem, we use the Lagrangian Relaxation of resource constraints in order to obtain a classical maximal closure problem which is treated in Chapter 2. In the Lagrangian relaxation scheme, the Lagrange multipliers are adjusted using a bundle method which is more appropriate than sub-gradient or cutting plane methods. This approach gives an upper bound on the optimal solution value of the problem but does not always obtain a primal feasible solution. So we have incorporated a simple heuristic into this approach attempting to obtain a primal feasible solution. This heuristic consists of two phases. The first phase transforms any infeasible solution obtained during the solution process to a feasible one that satisfies all resource constraints. The second phase improves the value of the primal solution obtained in the first phase using a tabu search. The value of the best primal feasible solution obtained is a lower bound on the optimal solution value of the problem. The solution details of the planning problem reduced to one period is discussed in Chapter 5 titled *" Maximal closure on a graph with resource constraints"*. The tests presented in this chapter have been conducted on medium-sized graphs (10 000 nodes and 100 000 arcs). The results obtained show small gaps between the upper and lower bounds. We also observe that each problem tested was solved using fewer than 10 subproblem evaluations on average. The solution approach developed for

the maximal closure problem with resource constraints is the first one to succeed in solving large-scale problems similar to those encountered in practice. This approach innovates with its use of an efficient method for multiplier adjustment and an appropriate algorithm for solving the subproblem.

The solution of the multi-period scheduling problem is based on the approach developed for the one period problem in Chapter 5. Each problem corresponding to each period is treated in an increasing order of periods using the approach presented in Chapter 5 where the heuristic is modified in order to obtain primal solutions compatible with those obtained for previous periods. The details of this solution process are presented in Chapter 6 titled *" Maximal closure on a multi-period network with resource constraints: An application to mining industries"*. This solution approach produces good results when applied to real data. For testing, we have considered a deposit of 150 000 blocks, a planning horizon of 10 periods and 3 resource constraints per period. The solutions obtained for this type of problem present gaps around 1%. The CPU time on these large-scale problems, involving 1 500 000 variables and 15 million constraints, is less than 6 hours using an HP9000/735 machine. The results obtained on problems randomly generated with different sizes are also satisfactory. The gaps lie between 0.1% and 3%. Among the randomly generated problems, we have identified the characteristics of a class of problems which present small gaps. The problems arising from mining applications generally present those characteristics. We have proposed improvements to the heuristic allowing reduction of gaps for the other problems whose the study is not the main objectif of this thesis.

The solution method developed in this thesis is the first one which can efficiently treat complex large-scale multi-period production scheduling problems in an open pit mine.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Dans cette thèse, nous nous sommes intéressé à la résolution du problème de la planification de l'exploitation d'une mine à ciel ouvert. Nous considérons un gisement minier représenté par un modèle de blocs tridimensionnel qui renseigne sur la position des blocs dans le gisement et sur l'ensemble des informations géologiques (teneur en minerai, densité, taux de récupération, etc.). Nous supposons en plus l'existence d'un modèle de valorisation des blocs qui renseigne sur les revenus générés à la suite de l'exploitation de chaque bloc durant chaque période de la durée de vie du gisement.

Le problème de la planification de l'exploitation consiste à considérer un horizon de planification de plusieurs périodes où, à chaque période, le plan d'extraction à déterminer doit satisfaire un ensemble de contraintes de ressource. Ces contraintes fixent généralement pour chaque période, la quantité de minerai désirée, les capacités de transport et de traitement disponibles, la teneur moyenne recherchée et les proportions de mélange des différentes catégories de minerai à respecter. La planification de l'exploitation cherche à déterminer, pour chaque période de l'horizon de planification, l'ensemble des blocs à exploiter de façon à maximiser la somme des revenus tout en respectant l'ensemble des contraintes de ressource et l'ensemble des contraintes physiques inhérentes à l'exploitation de tout gisement minier à ciel ouvert (pentes d'extraction, préséance, etc.).

Ce problème de la planification de l'exploitation multi-périodes peut être considéré à long, moyen ou court terme dépendemment de la longueur des périodes et

du type de contraintes de ressource considérées. Habituellement, une période correspond à l'année dans une planification à long terme, au mois ou au trimestre dans une planification à moyen terme et à la semaine ou au jour dans une planification à court terme.

Comme nous le verrons dans la revue de la littérature, plusieurs problèmes connus dans les industries minières sont des cas particuliers du problème de planification multi-périodes. Par exemple, le problème des contours finaux n'est autre que le problème que nous considérons, mais réduit à une seule période et sans contraintes de ressource. De même, le problème de design de fosse avec contraintes de ressource correspond à un problème de planification où l'horizon est d'une seule période.

Le problème de planification multi-périodes se formule mathématiquement comme un programme linéaire en variables binaires de très grande taille. Pour un gisement de 100 000 blocs et un horizon de 10 périodes, la formulation utilise de l'ordre de 1 million de variables et 10 millions de contraintes. La taille et la présence de variables binaires rendent ce problème parmi les plus difficiles à traiter directement par les outils de l'optimisation. Plusieurs tentatives de développement d'approches de résolution à ce problème ont vu le jour durant les dernières années. Les plus récentes sont l'essai de la décomposition de Dantzig-Wolfe par T.B. Johnson et l'essai de la relaxation lagrangienne avec ajustement des multiplicateurs par la méthode du sous-gradient effectué par K. Dagdeleen. Ces deux tentatives se sont butées à la taille du problème. Néanmoins, elles ont permis de mieux comprendre la structure du problème. En pratique dans les industries minières, le problème est encore traité manuellement par des procédés heuristiques. Quelques méthodes informatisées sont apparues récemment. Ces méthodes, non basées sur les techniques d'optimisation, présentent deux inconvénients majeurs: elles ne traitent pas

les contraintes de ressource de façon adéquate et ne peuvent quantifier la qualité des solutions proposées. L'utilisation des ces outils se réduit à une fonction d'assistance à la prise de décision plutôt qu'une méthode de prise de décision.

De nos jours, la rationalisation des ressources et la quête d'une plus grande rentabilité exigent pour les industries minières une planification optimale de l'exploitation. Ainsi le développement d'une bonne approche de résolution à ce problème de planification multi-périodes leur permettra de disposer d'un outil de prise de décision. Cet outil sera capable de construire des plans d'exploitation réalistes qui tiennent compte de toutes les contraintes d'exploitation tout en optimisant les profits des opérations.

L'objectif de cette thèse est de développer une approche de résolution efficace pour le problème de planification multi-périodes. Le chapitre 1 de cette thèse présente une formulation du problème faisant ressortir sa structure et discute des travaux connus dans la littérature dont il a été l'objet. Dans ce chapitre, l'approche de la relaxation lagrangienne des contraintes de ressource est identifiée comme une approche prometteuse avec laquelle on peut efficacement exploiter la structure particulière du problème. Les chapitres 2 et 3 traitent le problème en absence de contraintes de ressource, respectivement sur une période et sur plusieurs périodes. Dans le chapitre 2, une étude des algorithmes de flot maximum est effectuée en vue de sélectionner le meilleur algorithme permettant de traiter le problème d'une période sans contraintes de ressource. Le chapitre 4 présente les principes de la relaxation lagrangienne et les méthodes d'optimisation duales pouvant être utilisées pour l'ajustement des multiplicateurs de Lagrange. Au chapitre 5, nous présentons, pour le problème de la planification réduit à une seule période, une approche de résolution basée sur les résultats des chapitres précédents. Une extension de cette

approche est présentée au chapitre 6 pour la résolution du problème global de planification multi-périodes. Une conclusion sur les résultats obtenus et les futurs axes de recherche clôt ce travail.

CHAPITRE 1

Planification de l'exploitation dans une mine à ciel ouvert: formulation et revue de la littérature

1.1 Introduction

L'importance des gains que pourrait apporter de meilleurs outils de planification pour les exploitations minières a incité les chercheurs de ce domaine à introduire les techniques d'optimisation. Ainsi les problèmes posés ont été représentés par des modèles mathématiques qui permettent l'application de ces techniques. D'abord les ingénieurs miniers ont introduit le modèle tridimensionnel de blocs pour représenter un gisement minier. Un modèle de blocs consiste à subdiviser le gisement minier en blocs sous forme de parallélipipèdes et à associer à chaque bloc un ensemble d'informations géologiques, physiques et économiques. Ces informations renseignent habituellement sur la position du bloc dans le gisement, les quantités de minerai et de stérile qu'il contient, ainsi que les coûts et revenus que son extraction peut engendrer.

La planification de l'exploitation dans une mine à ciel ouvert consiste, pour un horizon de plusieurs périodes, à déterminer les blocs à exploiter à chaque période de façon à maximiser les revenus totaux, tout en respectant l'ensemble des contraintes imposées par le système de production. Les revenus générés par l'exploitation des blocs à la première période peuvent être calculés à partir des informations disponibles pour chaque bloc dans le modèle. Par actualisation de ces revenus, on détermine

pour chaque bloc les revenus que son extraction générera durant les périodes futures. En plus de l'actualisation, une correction est apportée à ces revenus pour tenir compte de la fluctuation des cours du marché du minerai et d'autres facteurs qui les influencent directement. La prise en considération de la dimension temps en planification minière permet d'aboutir à des décisions plus réalistes. Elle permet autant que possible d'extraire au plus tôt le minerai rentable et de retarder l'extraction du stérile.

Suivant l'horizon de planification considéré, la planification peut être à long, moyen ou court terme. Par exemple, une planification à long terme peut considérer un horizon de plusieurs années où chaque année correspond à une période et elle devient à court terme pour un horizon d'un trimestre avec des périodes hebdomadaires. Le problème de planification comporte deux grandes catégories de contraintes:

- Les contraintes physiques et géologiques qui assurent qu'un bloc ne peut être exploité que s'il est à découvert, et qui garantissent le respect des pentes d'extraction imposées par la géologie minière.

- Les contraintes de ressource qui définissent pour chaque période les quantités de minerai et de stérile à exploiter. Ces contraintes expriment aussi la capacité des concentrateurs, la disponibilité des pelles et des camions et les capacités propres des différentes unités du système de production. Ces contraintes de ressource comprennent aussi les contraintes de mélange indiquant les pourcentages acceptables pour les différents constituants du mélange de minerai à extraire.

L'objectif de cette recherche est de proposer une méthode de résolution efficace au problème de la planification de l'exploitation dans une mine à ciel ouvert. Ainsi, en vue de dégager une méthodologie de recherche, nous présentons sa formulation

mathématique en faisant ressortir la structure des différents problèmes qui lui sont liés et une revue de la littérature faisant l'état de l'art sur le sujet.

1.2 Formulation du problème

En vue de mieux comprendre la structure du problème global de la planification de l'exploitation dans une mine à ciel ouvert, nous allons présenter le problème des contours finaux et celui des séquences d'exploitation. Un modèle mathématique décrivant le problème de la planification sera déduit à la lumière de la présentation des problèmes précédents.

1.2.1 Problème des contours finaux d'exploitation

La détermination des contours finaux d'exploitation dans une mine à ciel ouvert consiste à déterminer l'ensemble des blocs à exploiter afin de maximiser le revenu total généré par l'exploitation, en respectant l'ensemble des contraintes physiques. Dans ce problème, on considère un gisement de n blocs dont chacun fournit un revenu c_i $(i = 1, ..., n)$ et des ensembles Γ_i de blocs qu'il faut au préalable avoir extrait avant d'extraire chacun des blocs i.

Soit x_i la variable de décision associée à chaque bloc i $(i = 1, ..., n)$:

$$x_i = \begin{cases} 1 & \text{si le bloc } i \text{ est extrait,} \\ 0 & \text{sinon.} \end{cases}$$

Le problème des contours finaux se formule comme suit :

$$\max \sum_{i=1}^{n} c_i x_i \tag{1.1}$$

s.c.

$$x_i - x_j \leq 0 \ , \ j \in \Gamma_i \ \ (i = 1, ..., n) \tag{1.2}$$

$$x_i \in \{0,1\} \ \ (i = 1, \ldots, n). \tag{1.3}$$

L'objectif (1.1) porte sur la maximisation des revenus générés alors que les contraintes (1.2) sont des contraintes de préséance entre les blocs qui imposent qu'un bloc i ne peut être exploité qu'après que l'ensemble de blocs de Γ_i ait été extrait.

Le problème des contours finaux est parmi les grands problèmes formulés et résolus en planification minière. Mathématiquement, il se présente comme un problème de détermination de la fermeture maximale dans un graphe G, où les noeuds correspondent aux blocs et un arc (i, j) du noeud i vers le noeud j traduit une contrainte de préséance $x_i \leq x_j$. Chaque noeud i du graphe G est valorisé par le revenu c_i fourni par le bloc i, ce revenu pouvant être positif ou négatif. La figure 1.1 présente une disposition bidimensionnelle de blocs et le graphe de préséance associé.

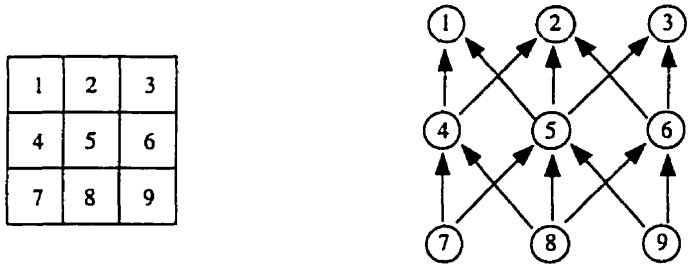


Figure 1.1 Disposition des blocs et Graphe de préséance G associé

1.2.2 Problème des séquences d'exploitation

La détermination des séquences d'exploitation consiste à considérer un horizon de planification de P périodes et à déterminer pour chaque période l'ensemble des blocs

à extraire. On connaît pour chaque bloc i le revenu c_i^p que son extraction génère à la période p $(p = 1, ..., P)$. Pour formuler ce problème, on définit les variables de décision suivantes:

$$x_i^p = \begin{cases} 1 & \text{si le bloc } i \text{ est exploité à la période } p, \\ 0 & \text{sinon.} \end{cases}$$

Le problème des séquences d'exploitation se formule de manière compacte comme suit :

$$\max \sum_{q=1}^{P} C^q X^q \tag{1.4}$$

s.c.

$$\sum_{q=1}^{p} EX^q \leq 0 \quad (p = 1, ..., P) \tag{1.5}$$

$$\sum_{q=1}^{P} X^q \leq \mathbf{1} \tag{1.6}$$

$$X^p \geq 0 \quad (p = 1, ..., P). \tag{1.7}$$

où : $X^p = (x_1^p, x_2^p, ..., x_n^p)^T$ vecteur des variables pour la période p;

$C^p = (c_1^p, c_2^p, ..., c_n^p)^T$ vecteur des revenus pour la période p;

$\mathbf{1} = (1, 1, ..., 1)^T$ vecteur colonne à n composantes unitaires;

E : transposée de la matrice d'incidence sommets-arcs du graphe de préséance entre les blocs. Chaque ligne de cette matrice correspond à un arc et chaque colonne correspond à un noeud. Une ligne qui correspond à un arc (i, j) présente $+1$ à la colonne i, -1 à la colonne j et zéro partout ailleurs.

Dans cette formulation, l'objectif porte sur la maximisation des revenus générés par l'exploitation et les contraintes (1.5) traduisent les relations de préséance entre les

blocs durant toutes les périodes. Les contraintes (1.6) garantissent qu'un bloc ne peut être extrait qu'une seule fois durant toutes les périodes.

A l'instar du problème des contours finaux, le problème des séquences est un problème de détermination de la fermeture maximale dans un réseau multi-périodes. Pour faire apparaître la structure de ce réseau, on reformule le problème en utilisant les variables suivantes:

$$w_i^p = \begin{cases} 1 & \text{si le bloc } i \text{ est extrait à l'une} \\ & \text{des périodes } q \quad (q \leq p), \\ 0 & \text{sinon.} \end{cases}$$

En posant $W^p = (w_1^p, w_2^p, ..., w_n^p)^T$, les nouvelles variables se calculent à partir des anciennes par la formule suivante:

$$W^p = \sum_{q=1}^{p} X^q.$$

En fonction des nouvelles variables, le problème (1.4)-(1.7) se réécrit:

$$\max \sum_{p=1}^{P-1} (C^p - C^{p+1}) W^p \quad + \quad C^P W^P \tag{1.8}$$

s.c.

$$EW^p \leq 0 \quad (p = 1, ..., P) \tag{1.9}$$

$$W^p - W^{p+1} \leq 0 \quad (p = 1, ..., P-1) \tag{1.10}$$

$$W^P \leq \mathbf{1} \tag{1.11}$$

$$W^1 \geq 0. \tag{1.12}$$

Dans cette nouvelle formulation, les contraintes (1.9) et (1.11) correspondent respectivement aux contraintes (1.5) et (1.6) de l'ancienne formulation. Les contraintes (1.10) et (1.12) sont déduites de la contrainte de non négativité (1.7) de l'ancienne formulation.

La matrice des contraintes de cette nouvelle formulation est la transposée de la matrice d'incidence sommets-arcs d'un réseau multi-périodes $\overline{G}$ constitué de P copies G^p $(p = 1, ..., P)$ du graphe de préséance G décrit pour le problème des contours finaux. Pour des fins pratiques, nous désignerons par (i/p) un noeud i dans le sous-graphe G^p. Les sous-graphes G^p sont inter-connectés par des arcs allant de tout noeud (i/p) dans G^p à son semblable $(i/p+1)$ dans G^{p+1}. La figure 1.2 illustre ce type de réseau multi-périodes.

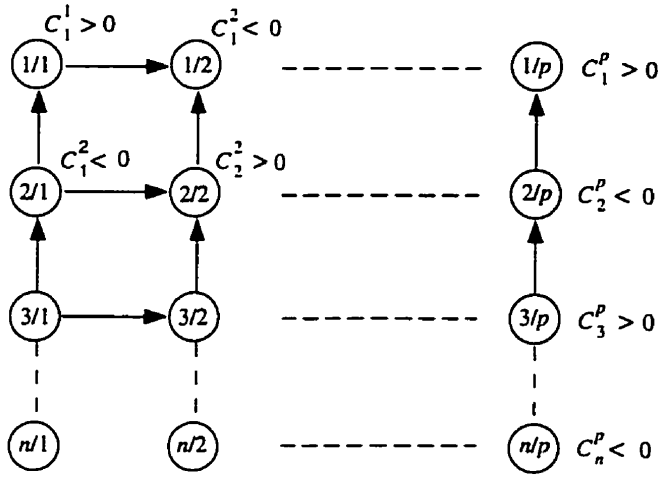


Figure 1.2 Réseau multi-périodes $\overline{G}$.

1.2.3 Problème de la planification de l'exploitation

La planification de l'exploitation dans une mine à ciel ouvert se formule comme un problème des séquences d'exploitation auquel sont ajoutées des contraintes de ressource. Les contraintes de ressource se définissent pour chaque période de l'horizon de planification. Elles traduisent les capacités des différentes unités du système de production ainsi que les contraintes de mélange. Par exemple, des bornes sur les quantités de minerai et de stérile à exploiter durant chaque période sont habituellement imposées, de même chaque unité comme le broyage, la séparation, etc., peut

avoir une propre capacité à respecter. Pour une période p ($p = 1, ..., P$), les contraintes de ressource se formulent comme des contraintes linéaires en fonction des variables associées aux blocs à la période p. Par exemple, soit a_{ki} la quantité de minerai (ressource k) contenu dans le bloc i et soit B_k^p la quantité de minerai totale demandée à la période p. La contrainte de ressource se formule comme suit:

$$\sum_{i=1}^{n} a_{ki} x_i^p \leq B_k^p$$

De manière compacte, l'ensemble des contraintes de ressource relatif à une période s'écrit sous forme matricielle comme suit:

$$A^p X^p \leq B^p$$

où:

A^p : matrice des coefficients ($m \times n$)

m : nombre de contraintes de ressource

n : nombre de blocs

B^p : vecteur colonne des limites de capacité.

Le problème de la planification de l'exploitation se formule comme suit:

$$\max \sum_{q=1}^{P} C^q X^q$$

s.c.

$$A^p X^p \leq B^p \quad (p = 1, ..., P)$$

$$(1.5), (1.6) \ \textit{et} \ (1.7).$$

Cette formulation est obtenue à partir de celle du problème des séquences d'exploitation en ajoutant les contraintes de ressource. Le problème de la planification de l'exploitation dans une mine à ciel ouvert se classe parmi les problèmes de très grande taille. Il présente pour une situation réelle des centaines de milliers de variables et

des millions de contraintes.

Actuellement, les exploitants miniers ne disposent pas d'outils de planification basés sur l'optimisation et pouvant prendre en considération l'ensemble des contraintes définies dans la formulation précédente. L'enjeu économique d'une bonne planification est de taille. En considérant une exploitation qui commercialise 20 millions de tonnes de minerai à un revenu de 30 dollars la tonne, une amélioration de 1% correspondrait à une valeur monétaire de 6 millions de dollars. Ainsi le développement d'une approche de résolution efficace pour ce problème est d'une importance capitale pour les exploitants miniers. En plus de leur assurer une exploitation optimale, elle leur permettra de disposer d'un outil de planification qui peut être considéré à long, moyen ou court terme dépendant de l'horizon considéré.

Ce problème, sous sa forme globale, est intrinsèquement lié, comme le montre sa formulation, au problème des séquences d'exploitation et celui des contours finaux. Ainsi le problème des séquences s'obtient à partir du problème global par relaxation des contraintes de ressource et celui des contours finaux n'est autre qu'un problème des séquences réduit à une seule période. Ces relations entre les différents problèmes formulés dans cette section, nous amène à conclure que la recherche d'une bonne approche de résolution du problème global passe nécessairement par la résolution efficace du problème des séquences et celui des contours finaux.

1.3 Revue de la littérature

Vu l'importance des problèmes découlant de la formulation du problème global de la planification, nous présentons dans cette revue les différentes approches de résolution qui ont été proposées pour chacun de ces problèmes.

1.3.1 Contours finaux d'exploitation

Le problème des contours finaux a été le premier problème qui a attiré l'attention des chercheurs dans ce domaine. Il a fait l'objet de plusieurs études qui ont conduit à plusieurs approches de résolution. Ces approches se classent en deux grandes catégories: les approches heuristiques et les approches exactes.

1.3.1.1 Approches heuristiques

Les heuristiques sont des méthodes qui fonctionnent bien dans la plupart des cas mais elles ne garantissent pas l'optimalité des solutions trouvées. Plusieurs heuristiques ont été développées pour la résolution du problème des contours finaux, dont celles basées sur le principe du cône mobile et celles basées sur le principe de la programmation dynamique.

1. **Heuristiques du cône mobile :** Une heuristique du cône mobile est un processus d'essais et erreurs qui analyse plusieurs contours possibles en déplaçant le sommet d'un cône d'un bloc à valeur positive à un autre et à chaque position, évalue la valeur totale des blocs contenus dans le cône. Les contours du cône s'ajustent habituellement aux pentes d'extraction maximales permises. Ce processus prend fin lorsque tous les cônes centrés aux blocs à valeurs positives ont été évalués. La solution qui sera retenue est celle correspondant à l'extraction des blocs contenus dans le cône d'évaluation maximale. Ces heuristiques diffèrent essentiellement par leur mise en oeuvre. Plusieurs variantes dont celles développées par Lemieux [33], Marino & Slama [35], Korobov [30] et Philips [36] sont utilisées dans les industries minières.

2. **Heuristiques de programmation dynamique :** Ces méthodes appliquent le principe de la programmation dynamique pour la résolution du problème

des contours finaux, mais elles restent heuristiques car elles ne peuvent traiter tous les états et étapes du cheminement d'une résolution par programmation dynamique. Les variantes les plus connues de ce type d'heuristiques sont celles développées par Johnson [25], Barne [3] et Koenigsberg [28].

Les méthodes heuristiques développées pour la résolution du problème des contours finaux ont été les premières méthodes à la disposition des ingénieurs miniers pour faire le design des fosses d'exploitation. Les heuristiques du cône mobile ont connues un grand succès à cause de leur simplicité tant au niveau compréhension qu'au niveau mise en oeuvre. Elles sont restées l'outil efficace de design des fosses jusqu'à l'apparition des méthodes exactes. Les heuristiques de programmation dynamique, qui sont apparues durant la même décennie que les méthodes exactes, n'ont pas été utilisées comme outil de planification dans les industries minières. Ces heuristiques garantissent de meilleures solutions mais elles sont généralement plus difficiles à comprendre et à mettre en oeuvre comparativement aux heuristiques du cône mobile.

1.3.1.2 Approches exactes

En se basant sur la formulation mathématique précédente, deux méthodes de résolution ont été proposées pour le problème des contours finaux. La première approche est due à Lerchs & Grossman [34] qui ont identifié le problème comme une fermeture maximale dans un graphe orienté. Les noeuds de ce graphe représentent les blocs et les arcs traduisent les relations de préséance entre les blocs. A chaque nœud de ce graphe est associée une valeur qui est représentée par le coût de la variable associée au nœud dans l'objectif (1.1) de la formulation du problème des contours finaux. L'algorithme que Lerchs & Grossman ont proposé est une méthode qui permet de trouver la fermeture maximale sur le graphe de préséance. La deuxième approche

a été proposé par Picard [37] et elle consiste à résoudre le problème de fermeture maximale sur le graphe de préséance comme un problème de flot maximum sur un graphe construit à partir du graphe de préséance. Ce nouveau graphe est obtenu par ajout d'une source s reliée à tout noeud i à valeur positive ($c_i > 0$) par un arc (s, i) de capacité égale à la valeur du noeud, et d'un puits t relié à tout noeud j à valeur négative ou nulle ($c_j \leq 0$) par un arc (j, t) de capacité égale à moins la valeur du noeud j. Dans ce nouveau graphe les arcs de préséance ont des capacités infinies. La fermeture maximale recherchée sur le graphe de préséance correspondra à l'ensemble des noeuds du côté de la source dans la coupe minimale déterminée par un algorithme de flot maximum sur le nouveau graphe proposé par Picard.

La résolution du problème des contours finaux par l'approche de Picard nécessite l'utilisation d'un algorithme de flot maximum. Comme plusieurs algorithmes sont disponibles dans la littérature, nous présentons les différentes idées de ces algorithmes en vue de guider notre choix sur l'un d'eux pour la résolution du problème des contours finaux. Le premier algorithme de flot maximum a été développé par Ford & Fulkerson [14]. Cet algorithme est basé sur la notion de chaîne d'augmentation de flot qui se définit comme une chaîne de la source au puits d'un graphe telle que pour tout arc (i, j) direct, le flot f_{ij} est inférieur à la capacité de l'arc et pour tout arc inverse (k, j), le flot f_{kj} est positif. Dans cet algorithme, un processus d'étiquettage est utilisé pour déterminer une chaîne d'augmentation de flot et une fois déterminée, le flot est augmenté le long de la chaîne autant que possible par augmentation des flots sur les arcs directs et diminution des flots sur les arcs inverses. Ce processus de détermination d'une chaîne augmentante et d'augmentation du flot le long de cette chaîne est répété jusqu'à l'impossibilité de construction d'une nouvelle chaîne d'augmentation de flot. L'algorithme de Ford & Fulkerson, dans sa version originale, présente deux inconvénients majeurs qui sont:

- La convergence n'est garantie que pour des capacités rationelles.
- La complexité est $O(Vnm)$, où n est le nombre de noeuds, m est le nombre d'arcs et V est la valeur du flot. Cette complexité pseudo-polynomiale peut conduire à des temps de calcul relativement élevés en présence de graphes mal conditionnés.

Depuis l'apparition de l'algorithme de Ford & Fulkerson, plusieurs autres algorithmes cherchant à pallier aux insuffisances précitées ont été proposés pour la résolution du problème de flot maximum. Dans ce cadre, Edmonds & Karp [12] améliorent l'algorithme de Ford & Fulkerson en déterminant à chaque itération la plus courte chaîne d'augmentation de flot par utilisation d'une recherche en largeur d'abord (premier étiqueté, premier visité). Cette amélioration a conduit à une complexité de $O(nm^2)$ tout en garantissant la convergence en présence de capacités non rationelles. Une deuxième génération d'algorithmes a pris naissance à la suite de l'algorithme de Dinic [11] qui utilise le concept de graphe à niveaux *(layered network)* pour obtenir une meilleure complexité de $O(n^2m)$. La complexité de l'algorithme de Dinic a été amélioré par Karzanov [30] en $O(n^3)$ par utilisation des pré-flots sur le graphe à niveaux. Un pré-flot est un flot satisfaisant les contraintes de capacité sur les arcs mais pouvant ne pas satisfaire la conservation de flot aux noeuds. Les idées de base tirées du graphe à niveaux et de la méthode des pré-flots ont été exploitées dans d'autres algorithmes cherchant à améliorer la complexité par utilisation de structures de données appropriées. Cherkasky [5], par une modification de la méthode des pré-flots de Karzanov et sa combinaison avec l'approche de Dinic, aboutit à une complexité de $O(n^2m^{1/2})$. Par la suite, Galil [19] raffine la méthode de Cherkasky pour arriver à une complexité de $O(n^{5/3}m^{2/3})$. Sleator & Tarjan [41, 44], par l'utilisation des arbres dynamiques comme structure de données, proposent un

algorithme de complexité $O(nm\log(n))$. Goldberg & Tarjan [22] améliorent cet algorithme par l'introduction du concept de distances sur les noeuds du graphe et aboutissent à une complexité de $O(nm\log(n^2/m))$. Enfin récemment, Ahuja & Orlin [1] ont proposé un algorithme pour les graphes à capacités entières ayant une complexité de $O(nm + n^2\log(U))$ où U est la plus grande capacité sur les arcs.

Dans le chapitre 2 de cette thèse, nous procédons à une analyse comparative des meilleurs algorithmes de flot maximum en vue du choix de l'un d'eux pour la résolution du problème de la fermeture maximale sur un graphe.

1.3.2 Séquences d'exploitation

Le problème des séquences d'exploitation a été abordé dans la littérature comme un sous-problème qui apparaît dans la structure du problème global de la planification. Il s'obtient par relaxation des contraintes de ressource. Sa structure de réseau multi-périodes a été mise en évidence par Dagdeleen [7] dans sa thèse de doctorat. Pour résoudre ce problème, Dagdeleen & Johnson [8] ont proposé une décomposition qui consiste à résoudre, pour chaque, période un problème de contours finaux à l'aide de l'algorithme de Lerchs & Grossman. Ils tiennent compte de l'interaction entre les périodes par modification des coûts sur les blocs en passant d'une période à la suivante. Afin de présenter la décomposition de Dagdeleen, on définit les termes suivants:

c_i^p : coût associé au bloc i à la période p; ce coût représente le bénéfice généré par l'exploitation du bloc i à la période p.

$\overline{c}_i^p = c_i^p - c_i^{p+1}$: coût relatif associé au bloc i à la période p.

$\overline{\overline{c}}_i^p$: coût modifié du bloc i à la période p (voir étape 3 de l'algorithme ci-dessous).

Les étapes de l'algorithme proposé par Dagdeleen sont les suivantes:

Étape 1

- calculer les coûts des blocs pour chaque période p $(p = 1, ..., P)$

  $\overline{c}_i^p = c_i^p - c_i^{p+1}$ pour $p = 1, ..., P-1$;

  $\overline{c}_i^p = c_i^p$ pour $p = P$;

  $\overline{\overline{c}}_i^p = \overline{c}_i^p$ pour $p = 1$.

- poser $p = 0$

Étape 2

- $p = p + 1$
- résoudre le problème des contours finaux pour la période p en utilisant les coûts modifiés $\overline{\overline{c}}_i^p$ $(i = 1, ..., N)$.
- identifier l'ensemble des blocs solution A_p.
- si $p = P$ aller à l'étape 4.

Étape 3

- modifier les coûts des blocs pour la période $p+1$ comme suit :

  $\overline{\overline{c}}_i^{p+1} = \overline{c}_i^{p+1} + \overline{c}_i^p$ pour tout $i \in A_p$;

  $\overline{\overline{c}}_i^{p+1} = \overline{c}_i^{p+1}$ pour tout $i \notin A_p$.

- si $p = P$ aller à l'étape 4, sinon aller à l'étape 2.

Étape 4

- si la solution trouvée vérifie $A_1 \subset A_2 \subset \cdots \subset A_P$, alors extraire A_1 à la période 1 et $(A_p - A_{p-1})$ à la période p pour $p = 2, \ldots, P$, sinon poser:

$$\overline{A}_p = \bigcap_{q=p}^{P} A_q \quad \text{pour } p = 1, ..., P,$$

  poser $p = 0$ et aller à l'étape 2, en ne considérant que les blocs $\in \overline{A}_p$ pour le sous-problème p.

La méthode proposée par Dagdeleen est une heuristique qui ne garantit pas l'optimalité de la solution. Un exemple de petite taille sur lequel elle ne trouve pas la solution optimale peut être facilement construit. Néanmoins cette approche a l'avantage de contourner la difficulté du problème en le décomposant de manière à travailler sur des sous-problèmes de tailles raisonnables. Cette idée de décomposition mérite d'être explorée pour développer une approche de décomposition exacte.

1.3.3 Planification de la production

Le développement d'une approche de résolution basée sur les techniques d'optimisation pour le problème de la planification de l'exploitation dans une mine à ciel ouvert a été l'objet de beaucoup de recherches jusqu'ici. Comme ce problème a été formulé en un programme en nombres entiers à variables binaires, la première tentative de résolution était l'application des techniques de la programmation entière. Ces méthodes se sont avérées incapables de le résoudre à cause de la taille de sa formulation qui comporte des centaines de milliers de variables et des millions de contraintes. Pour contourner la difficulté de la taille du problème, des auteurs ont suggéré l'agrégation des blocs, ce qui diminue considérablement le nombre de variables et de contraintes dans le modèle. Cette idée n'est pas toujours réaliste et toute planification qui en découle est fausse car la réalité même du problème devient modifiée.

Quelques tentatives de résolution du problème en considérant sa taille réelle ont été réalisées. La plupart des tentatives cherchent à exploiter la structure particulière du problème. Johnson [24] a appliqué la décomposition de Dantzig-Wolfe au problème, cherchant ainsi à résoudre des sous-problèmes de tailles raisonnables. Le schéma de décomposition de Johnson a grandement contribué à la compréhension

de la structure globale du problème, mais il subsiste de sérieuses difficultés: Il faut entre autre obtenir une solution entière tandis que la décomposition de Dantzig-Wolfe produit une solution en nombres réels. Récemment Dagdeleen [7, 8] a proposé une approche de résolution basée sur la relaxation lagrangienne des contraintes de ressource et la résolution d'un sous-problème de séquences d'exploitation. Dans cette approche, l'auteur utilise la méthode de sous-gradient pour ajuster les multiplicateurs de Lagrange et propose l'heuristique présentée précédemment pour la résolution de son sous-problème. L'approche par la relaxation lagrangienne qui semble prometteuse pour traiter le problème n'a pas encore été essayée sur des cas réels. Dagdeleen a présenté son schéma sur un exemple très petit de vingt blocs par période.

Trois méthodes d'ajustement des multiplicateurs de Lagrange ont été proposées:

- L'ajustement paramétrique proposé par Davis & Williams [10] et qui consiste à augmenter d'une unité le multiplicateur correspondant à la contrainte dont la borne supérieure est la plus violée et à diminuer d'une unité le multiplicateur correspondant à la contrainte dont la borne inférieure est la plus violée.

- l'ajustement par sous-gradient introduit par Dagdeleen [7] et qui consiste à ajuster les multiplicateurs en se basant sur le degré de violation des contraintes. Cet ajustement modifie les multiplicateurs dans la direction du sous-gradient en se déplaçant d'un pas calculé à chaque itération.

- La dernière méthode d'ajustement proposée par Eleveli & *al.* [13] est une combinaison des deux premières. Elle utilise une méthode de sous-gradient avec un calcul de pas basé sur des courbes de distribution cumulative sur les valeurs des blocs rentables. Cette approche présente des similitudes avec la méthode du

lagrangien augmenté qui donne une manière d'ajuster les multiplicateurs basée sur une pénalisation des contraintes. L'effet des courbes utilisées dans cette méthode de calcul de pas peut être interprété comme l'effet de la pénalisation dans un lagrangien augmenté.

Les première et dernière méthodes d'ajustement proposées ont été employées seulement pour la résolution d'un problème de planification réduit à une seule période. Récemment Kim & Zhao [45], sur un problème de planification réduit à une seule période, affirment que l'application de la relaxation lagrangienne ne peut conduire à une solution optimale en cas d'existence de deux ou plusieurs solutions optimales pour le sous-problème qui dans leur cas est un problème de contours finaux. Les auteurs rattachent cette difficulté à la nature de l'algorithme des contours finaux employé pour la résolution du sous-problème et pensent que la difficulté ne peut être évitée par la manière d'ajuster les multiplicateurs de Lagrange. Notons que cette situation devrait être difficile à rencontrer en pratique, surtout sur des problèmes de grande taille comme celui-ci. Les auteurs appuient leur thèse par un exemple dans lequel les contraintes de ressource relaxées sont des contraintes d'égalité exprimées en nombre de blocs, donc très rigides comparativement aux contraintes de ressource habituellement rencontrées qui sont des contraintes d'inégalité. La difficulté soulevée par les auteurs est un cas très particulier dont l'occurence est très improbable dans des problèmes de grande taille. Cette situation peut être traduite par le graphique de la figure 1.3. Ce graphique présente l'évolution du revenu maximum en fonction du nombre de blocs exploités. Ce revenu augmente avec le nombre de blocs suivant la droite AC. La contrainte imposée sur le nombre de blocs est représentée en trait pointillé. La solution optimale recherchée est le point B qui satisfait la contrainte imposée, et les points A et C sont les points les plus proches de la solution optimale qui peuvent être obtenus par la relaxation lagrangienne. Cette situation très particulière ne peut avoir lieu que sur des problèmes

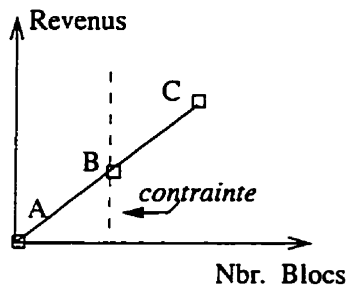


Figure 1.3 Caractéristique du problème considéré par Kim & Zhao.

de très petite taille. Pour les problèmes de grande taille comme le nôtre, le revenu total en fonction du nombre de blocs exploités, présente l'allure de la courbe de la figure 1.4. Cette courbe est d'allure générale concave même si elle peut présenter

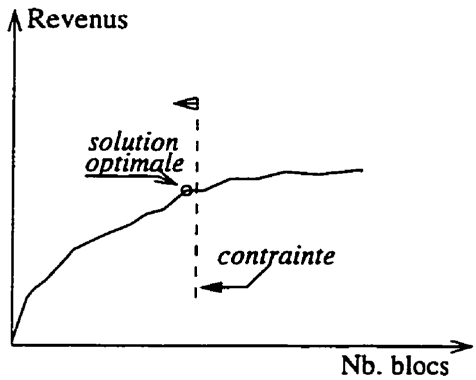


Figure 1.4 Caractéristique d'un problème de planification de grande taille.

des irrégularités locales. En effet le revenu croit rapidement au début car on extrait des blocs rentables et faciles à atteindre, par la suite le rendement diminue quand les blocs restants deviennent moins rentables ou plus difficiles à atteindre. De plus, à cause de la grande taille du problème, la courbe présente une multitude de points qui peuvent être obtenus par la relaxation lagrangienne. Pour une contrainte d'inégalité, il existera toujours un point la satisfaisant et relativement près de sa borne, qui peut être obtenu comme solution optimale par la relaxation lagrangienne. En conclusion, nous pensons que la difficulté soulevée par ces auteurs n'est pas pertinente pour

notre situation, et soutenons que l'approche de la relaxation lagrangienne reste une approche prometteuse qui mérite d'être explorée.

1.4 Conclusion

Cette recherche vise à développer une méthode de résolution efficace pour le problème de la planification de l'exploitation dans une mine à ciel ouvert. Ce problème se formule comme un programme linéaire en variables binaires de très grande taille, ne pouvant être traité par les techniques habituelles de la programmation entière. Ainsi, dans le développement d'une approche de résolution, il est essentiel de mettre à profit sa structure particulière. Suivant cette ligne de pensée, l'approche que nous cherchons à élaborer sera basée sur un schéma de relaxation lagrangienne des contraintes de ressource qui nous ramène à la résolution du problème des séquences d'exploitation. Le problème des séquences d'exploitation est un problème de flot sur un réseau multi-périodes de très grande taille qui ne peut être traité directement par les algorithmes existants de flot maximum. Nous retiendrons pour sa résolution l'idée de sa décomposition qui nous permet de travailler sur des graphes de tailles raisonnables.

Dans les chapitres suivants, nous traitons des points suivants:

- La résolution du problème des contours finaux par flot maximum sera à la base du schéma de décomposition envisagé pour le problème des séquences d'exploitation. Ainsi sa résolution rapide et efficace est grandement recherchée. Une étude comparative sur les différents algorithmes existants de flot maximum est nécessaire en vue de sélectionner le mieux adapté au problème des contours finaux et de voir dans quelle mesure ses performances peuvent être améliorées par adaptation à notre structure particulière de graphe. Cette étude

est présentée au chapitre 2 de cette thèse.

- La résolution du problème des séquences d'exploitation comme un problème de flot maximum sur un réseau multi-périodes est traitée au chapitre 3. Dans ce chapitre nous avons tenté de résoudre le problème de flot maximum en décomposant le réseau multi-périodes en plusieurs graphes de taille raisonnable. L'approche que nous proposons ne garantie pas l'optimalité de la solution en toute situation. De ce fait nous ne l'utiliserons pas dans le traitement du problème des séquences qui apparaît comme sous-problème dans un schéma de relaxation lagrangienne.

- Dans le chapitre 4 on fait un rappel sur les méthodes d'optimisation que nous utiliserons dans les chapitres 5 et 6 pour l'ajustement des multiplicateurs de Lagrange.

- La prise en considération des contraintes de ressource du problème de la planification de l'exploitation est une phase primordiale dans cette recherche. Nous explorons deux alternatives pour l'ajustement des multiplicateurs dans une approche de relaxation lagrangienne. La méthode du sous-gradient et la méthode des faisceaux (que nous voyons comme une méthode de plans sécants stabilisée) seront implémentées et comparées sur des problèmes de planification réduits à une période. Le chapitre 5 présente l'article exposant cette approche.

- Nous développons par la suite un schéma global de résolution du problème de planification de l'exploitation dans une mine à ciel ouvert. Ce schéma consiste à traiter par relaxation lagrangienne des problèmes séparés correspondant à chacune des périodes de l'horizon de planification et à utiliser des heuristiques permettant de déterminer des solutions réalisables pour l'ensemble du problème à partir des solutions non réalisables obtenues pour chacune des périodes lors

de l'optimisation duale. Cette approche de résolution sera testée sur un jeu de données réelles représentant un gisement minier en cours d'exploitation. Le chapitre 6 présente les détails de cette méthode de résolution.

CHAPITRE 2

Résolution du problème de la fermeture maximale par des algorithmes de flot maximum

2.1 Introduction

Dans l'approche de résolution que nous avons adoptée pour le problème de planification de l'exploitation dans une mine à ciel ouvert, le problème de la fermeture maximale sur un graphe apparaît comme un sous-problème dont la résolution efficace conditionne grandement le succès de notre approche. Que ce soit dans le problème de planification multi-périodes ou dans celui réduit à une période, le problème de la fermeture maximale est résolu quelques dizaines de fois même avec les meilleures méthodes d'ajustement des multiplicateurs. Ainsi pour pouvoir traiter des problèmes de très grande taille, il est primordial d'utiliser l'algorithme qui mettra le moins de temps possible pour effectuer le traitement.

Le problème de la fermeture maximale a été largement traité dans la littérature et plusieurs approches de résolution lui ont été proposées. Parmi ces approches, nous retiendrons dans cette thèse l'approche proposée par Picard [37] qui est simple et systématique. Dans cette approche, le problème de la fermeture maximale est identifié comme un problème de coupe minimale qu'on peut déterminer par un algorithme de flot maximum. Picard a formulé le problème de fermeture maximale comme un problème quadratique en variables binaires et a montré qu'il est équivalent à la formulation quadratique d'un problème de coupe minimale.

Dans ce chapitre, nous montrons que ce résultat peut être obtenu directement en passant par le dual d'une formulation linéaire du problème de la fermeture maximale. Par la suite, nous exposons les meilleurs algorithmes de flot maximum qui peuvent être utilisés pour la détermination d'une coupe minimale et nous discutons des améliorations qui peuvent être apportées aux versions originales de ces algorithmes en profitant de la structure particulière du problème considéré. Enfin nous présentons des tests comparatifs sur ces algorithmes en vue de guider le choix de l'un d'eux pour la résolution du problème de la fermeture maximale sur un graphe.

2.2 Fermeture maximale et flot maximum sur un graphe

Considérons un graphe orienté $G(N, A)$ où N est l'ensemble des nœuds et A est l'ensemble des arcs. Nous désignons l'ensemble des successeurs d'un nœud $i \in N$ par Γ_i et l'ensemble des prédécesseurs d'un nœud $i \in N$ par Γ_i^{-1}. On associe à chaque nœud $i \in N$ une valeur c_i, qui peut être positive, négative ou nulle. On appelle fermeture sur le graphe G, un ensemble de nœuds $S \subseteq N$ tel que, pour tout nœud $i \in S$, les successeurs de i appartiennent à S, i.e., $\forall\, i \in S$ on a $j \in S$, $\forall\, j \in \Gamma_i$. On définit la valeur d'une fermeture S par la somme des valeurs de ses nœuds. Une fermeture est dite maximale si sa valeur est la plus grande parmi les valeurs de toutes les fermetures possibles.

Le problème de la fermeture maximale sur le graphe G se formule en utilisant les variables de décision suivantes:

$$x_i = \begin{cases} 1 & \text{si le nœud } i \text{ appartient à la fermeture,} \\ 0 & \text{sinon.} \end{cases}$$

$$\max \sum_{i \in N} c_i x_i \tag{2.1}$$

s.c.

$$x_i - x_j \leq 0, \quad \forall\, j \in \Gamma_i, \ \forall\, i \in N \tag{2.2}$$

$$x_i \in \{0,1\}, \quad \forall\, i \in N. \tag{2.3}$$

L'objectif (2.1) de cette formulation vise à maximiser la valeur de la fermeture recherchée. La contrainte (2.2) traduit les relations de préséance entre un nœud et ses successeurs dans le graphe. Pour construire le dual de (2.1) – (2.3), nous considérons la relaxation linéaire $x_i \leq 1$ et $x_i \geq 0$ des contraintes (2.3) et associons aux contraintes $x_i \leq 1$ les multiplicateurs $\alpha_i \geq 0$ et aux contraintes (2.2) les multiplicateurs $f_{ij} \geq 0$.

Le dual du problème de la fermeture maximale sur un graphe G se formule ainsi:

$$\min \sum_{i \in N} \alpha_i$$

$$c_i - \Big(\sum_{j \in \Gamma_i} f_{ij} - \sum_{j \in \Gamma_i^{-1}} f_{ji}\Big) \leq \alpha_i, \quad \forall\, i \in N$$

$$\alpha_i \geq 0, \quad \forall i \in N$$

$$f_{ij} \geq 0 \quad \forall\, (i,j) \in A.$$

Posons les deux relations suivantes:

$$\text{(a)} \quad f_{si} = \sum_{j \in \Gamma_i} f_{ij} - \sum_{j \in \Gamma_i^{-1}} f_{ji} \qquad \text{si } c_i > 0$$

$$\text{(b)} \quad f_{it} = \sum_{j \in \Gamma_i^{-1}} f_{ji} - \sum_{j \in \Gamma_i} f_{ij} \qquad \text{si } c_i \leq 0.$$

A l'aide de ces deux relations, le programme dual précédent s'écrit:

$$\min_{\alpha_i \geq 0} \sum_{i \in N} \alpha_i$$

$$c_i - f_{si} \leq \alpha_i, \quad \text{si } c_i > 0$$

$$c_i - f_{it} \leq \alpha_i, \quad \text{si } c_i \leq 0$$

$$f_{si} = \sum_{j \in \Gamma_i} f_{ij} - \sum_{j \in \Gamma_i^{-1}} f_{ji}, \quad \text{si } c_i > 0$$

$$f_{it} = \sum_{j \in \Gamma_i^{-1}} f_{ji} - \sum_{j \in \Gamma_i} f_{ij}, \quad \text{si } c_i \leq 0$$

$$\alpha_i \geq 0, \quad \forall i \in N$$

$$f_{ij} \geq 0, \quad \forall (i,j) \in A.$$

L'optimum de ce programme dual est obtenu pour les valeurs de α_i suivantes:

$$\alpha_i = 0 \quad \text{si } c_i \leq 0$$

$$\alpha_i = c_i - f_{si}^* \quad \text{si } c_i > 0,$$

où f_{si}^* est l'optimum du problème PF suivant:

$$\max \sum_{i\,:\,c_i > 0} f_{si}$$

$$f_{si} \leq c_i \quad \text{si } c_i > 0$$

$$f_{it} \leq |c_i| \quad \text{si } c_i \leq 0$$

$$f_{si} + \sum_{j \in \Gamma_i^{-1}} f_{ji} - \sum_{j \in \Gamma_i} f_{ij} = 0 \quad \forall\ i\ :\ c_i > 0$$

$$f_{it} + \sum_{j \in \Gamma_i} f_{ij} - \sum_{j \in \Gamma_i^{-1}} f_{ji} = 0 \quad \forall\ i\ :\ c_i \leq 0$$

$$f_{ij} \geq 0\ ,\ f_{si} \geq 0 \text{ et } f_{it} \geq 0.$$

Le programme PF est la formulation d'un problème de flot maximum défini sur un graphe G^e construit comme une extension du graphe G. Le graphe G^e est obtenu par l'ajout d'une source s et d'un puits t à l'ensemble des nœuds N de G. La source s est reliée dans G^e à tout nœud $i \in N$ de valeur $c_i > 0$ par un arc (s, i) de capacité égale à c_i. De même tout nœud $j \in N$ de valeur $c_j \leq 0$ est relié au puits t dans G^e par un arc (j, t) de capacité égale à $|c_j|$. Les capacités des arcs (i, j) déjà présents dans le graphe G sont considérées infinies.

Dans la suite du texte, nous désignons par $N^e = N \cup \{s, t\}$ et $A^e = A \cup \{(s, i) : i \in N \text{ et } c_i > 0\} \cup \{(j, t) : j \in N \text{ et } c_j \leq 0\}$, respectivement l'ensemble des nœuds et l'ensemble des arcs du graphe G^e. De même nous notons c_{ij} la capacité d'un arc $(i, j) \in A^e$.

La valeur optimale du problème de fermeture maximale dans G est déterminée par : $\sum_{i\,:\,c_i>0}(c_i - f^*_{si})$ où f^*_{si} est le flot optimal sur l'arc (s, i) dans le graphe G^e. L'ensemble des nœuds faisant partie de la fermeture maximale recherchée sur le graphe G est l'ensemble des nœuds appartenant à N et du côté de la source s par rapport à une coupe minimale dans G^e. En conclusion, ceci montre que le problème de fermeture maximale sur le graphe G se résout comme un problème de coupe minimale sur le graphe étendu G^e. Une coupe minimale sur G^e est déterminée par un algorithme de flot maximum. Dans la suite nous étudierons les algorithmes de flot maximum existants.

2.3 Algorithmes de flot maximum

Le problème de flot maximum est l'un des problèmes de graphe qui a connu le plus grand nombre de développements. Plusieurs algorithmes de plus en plus performants

ont été développés depuis l'apparition du premier algorithme proposé par Ford & Fulkerson [14] au début des années 60. Sans être exhaustif, nous allons présenter dans cette section deux familles d'algorithmes qui nous semblent parmi les plus performants et les plus simples à implémenter.

La première famille est celle des algorithmes basés sur le principe de la chaîne augmentante. L'algorithme de Ford & Fulkerson est le précurseur de cette famille. Nous présentons de cette famille *l'algorithme du chemin augmentant minimal*, qui utilise les chaînes les plus courtes en priorité (Edmonds & Karp [12]). La complexité théorique de cet algorithme est $O(n^2m)$ où n est le nombre de nœuds et m est le nombre d'arcs dans le graphe.

La deuxième famille est celle des algorithmes de pré-flot. Nous présentons l'algorithme générique de pré-flot et regardons deux façons de l'implémenter. Ces deux façons permettent d'améliorer la complexité théorique de l'algorithme générique qui est également $O(n^2m)$.

Pour la clarté de l'exposé des algorithmes, nous définissons les notions suivantes:

- Graphe résiduel:

  Soit f un flot de s à t dans G^e compatible avec les contraintes de capacité $0 \leq f_{ij} \leq c_{ij}$. On appelle graphe résiduel associé au flot f, le graphe $\overline{G}^e$ qui possède le même ensemble de nœuds que G^e et dont l'ensemble des arcs est construit comme suit:

  A chaque arc (i,j) de G^e est associé au plus deux arcs dans $\overline{G}^e$:

  l'arc (i,j) si $r_{ij} = c_{ij} - f_{ij} > 0$

  l'arc (j,i) si $r_{ji} = f_{ij} > 0$.

Nous utilisons la notation $V(i)$ pour désigner l'ensemble des arcs émanant du nœud i dans le graphe résiduel $\overline{G}^e$.

- Concept de distance:

  On appelle distance *valide* une fonction $d(.)$ de l'ensemble des nœuds vers l'ensemble des entiers naturels, telle que $d(t) = 0$ et $d(i) \leq d(j) + 1$ pour tout arc (i, j) du graphe résiduel.

  Cette définition signifie que la distance d'un nœud i est une borne inférieure sur la distance minimale de i à t dans le graphe résiduel. La distance d'un nœud i est dite exacte si elle est égale à la distance minimale de i à t. Par rapport à cette définition de distance, on définit comme arc *admissible* dans le graphe résiduel, tout arc (i, j) vérifiant $d(i) = d(j) + 1$. De même, on définit comme chaîne *admissible* de s à t, toute chaîne composée uniquement d'arcs admissibles. Une chaîne admissible est une chaîne de longueur minimale de s à t.

2.3.1 Algorithme du chemin augmentant minimal

L'idée de base de cet algorithme (en anglais :*Shortest Augmenting Path Algorithm*) est de déterminer à chaque itération une chaîne augmentante de s à t et d'y acheminer le maximum de flot possible. La différence par rapport aux autres algorithmes de sa famille est que cet algorithme utilise successivement les chaînes les plus courtes entre la source s et le puits t. Au lieu d'utiliser à chaque itération un algorithme de plus court chemin pour trouver une chaîne augmentante, ce qui naturellement est très coûteux en terme de complexité, l'algorithme maintient à jour comme étiquette sur chaque nœud sa distance minimale par rapport au puits du graphe. Notons que les distances initiales sont déterminées par une recherche en largeur d'abord en

partant du puits du graphe. Cette recherche coûte $O(m)$ où m est le nombre d'arcs. L'*algorithme du chemin augmentant minimal* procède par augmentation de flot le long de chaîne admissible dans le graphe résiduel. L'algorithme construit cette chaîne en partant du nœud source s et en construisant itérativement la chaîne en effectuant l'une des opérations **avance** ou **retire** à partir du dernier nœud de la chaîne partielle déjà formée. On appellera ce dernier nœud, le *nœud courant.* L'opération **avance** est effectué si le *nœud courant* i admet un arc admissible (i, j) dans le graphe résiduel. Cette opération ajoute l'arc (i, j) à la chaîne partielle et considère ensuite le nœud j comme *nœud courant.* Dans l'autre situation où l'algorithme n'arrive pas à trouver un arc admissible pour faire progresser la chaîne, il effectue l'opération **retire**. Cette opération supprime le dernier arc de la chaîne partielle, modifie l'étiquette de distance sur le *nœud courant* i et considère ensuite comme *nœud courant*, le prédécesseur de i dans la chaîne partielle avant suppression du dernier arc. Les opérations **avance** et **retire** sont répétées jusqu'à ce que la chaîne partielle atteigne le puits du graphe. Dans un tel cas, l'algorithme procède à une augmentation de flot le long de la chaîne trouvée. *L'algorithme du chemin augmentant minimal* termine s'il n'y a plus de chaînes admissibles entre s et t dans le graphe résiduel. Une telle situation se produit quand la condition $d(s) \geq n$ est vérifiée, où n est le nombre de nœuds.

Les étapes de cet algorithme sont les suivantes:

Début

- Initialiser le flot à zéro $f_{ij} := 0$, $\forall\, (i, j) \in A^e$
- Déterminer les distances exactes $d(i)$, $\forall\, i \in N^e$
- $i = s$
- Tant que $d(s) < n$ faire

– s'il existe un arc admissible (i, j) faire

avance(i)

si $i = t$ **augmente** et poser $i = s$

– sinon **retire(i)**

Fin.

Les procédures **avance(i)**, **retire(i)** et **augmente** sont définies comme suit:

Procédure **avance(i)**

Début

soit (i, j) un arc admissible

poser $pred(j) = i$ et $i = j$

Fin.

Procédure **retire(i)**

Début

$d(i) = \min\{d(j) + 1 : (i, j) \in V(i) \text{ et } r_{ij} > 0\}$

si $i \neq s$ poser $i = pred(i)$

Fin.

Procédure **augmente**

Début

Soit P la chaîne augmentante définie par le vecteur $pred$

$\delta = \min\{r_{ij} : (i, j) \in P\}$

augmenter le flot de δ sur la chaîne P

Fin.

2.3.2 Algorithmes de pré-flot

La complexité des algorithmes basés sur les chaînes augmentantes est essentiellement dûe au temps passé dans l'augmentation de flot le long des chaînes. Pour éviter cette situation, les algorithmes de pré-flot, au lieu d'utiliser le principe de chaînes augmentantes, cherchent à pousser le flot simultanément à partir de la source du graphe jusqu'à son puits. Ils maintiennent ainsi un pré-flot sur chaque arc du graphe. Un pré-flot est un "flot" satisfaisant les contraintes de capacité sur les arcs mais non les contraintes de conservation aux nœuds. Un pré-flot doit satisfaire la relaxation suivante:

$$e(i) = \sum_{j \in \Gamma_i^{-1}} f_{ji} - \sum_{j \in \Gamma_i} f_{ij} \geq 0 \quad \forall\, i \in N^e \setminus \{s,t\}.$$

$e(i)$ est appelé surplus de flot au nœud i et désigne la quantité de flot accumulée au nœud i en attente d'être transférée. Les surplus de flot au puits et à la source vérifient $e(t) \geq 0$ et $e(s) \leq 0$. Un nœud du graphe avec un surplus strictement positif est dit actif. Par convention, nous considérons que la source et le puits ne sont jamais actifs. La présence d'un nœud actif dans le graphe est signe d'une contrainte de conservation de flot non satisfaite. Ainsi pour atteindre la faisabilité (conversion des pré-flots en flots), l'algorithme sélectionne un nœud actif et tente de transférer son surplus vers ses voisins. Il choisit les voisins les plus proches du puits t. A partir d'un nœud i, les voisins les plus proches du puits sont ceux accessibles à partir du nœud i par des arcs admissibles (un arc est admissible si $d(i) = d(j) + 1$ et sa capacité résiduelle r_{ij} est strictement positive). Si le nœud actif que l'algorithme examine n'admet pas d'arcs admissibles, l'algorithme augmente son étiquette de distance jusqu'à la création d'au moins un arc admissible. Ce processus de sélection de nœud actif, de transfert de flots ou reétiquetage du nœud est continué jusqu'à ce

qu'il n'y ait plus de nœuds actifs.

Nous présentons les étapes de l'algorithme générique de pré-flot et discuterons par la suite deux façons de sélectionner les nœuds actifs.

Algorithme générique de pré-flot :

Début

- **Pré-traitement**
- Tant que le graphe contient un nœud actif, faire
  sélectionner un nœud actif i
  Push_relabel(i)

Fin.

Les procédures **Pré-traitement** et **Push_relabel(i)** sont les suivantes:

Procédure **Pré-traitement**

Début

- initialiser le flot à zéro $f_{ij} := 0,\ \forall\ (i,j) \in A^e$
- déterminer les distances exactes $d(i),\ \forall\ i \in N^e$
- $f_{si} = c_{si}$ pour tout arc (s,i)
- poser $d(s) = n$ où $n = |N^e|$ est le nombre de nœuds du graphe

Fin.

Procédure **Push_relabel(i)**

Début

- Si le graphe contient un arc admissible (i, j)

  pousser $\delta = \min\{e(i)\ ,\ r_{ij}\}$ unités de flot du nœud i vers le nœud j,

  mettre à jour $e(i)$ et $e(j)$

  Sinon

  poser $d(i) = \min\{d(j) + 1 : (i, j) \in V(i) \text{ et } r_{ij} > 0\}$

Fin.

Quand l'algorithme générique de pré-flot sélectionne un nœud actif i, nous disons que le nœud est examiné si l'algorithme pousse son surplus de flot vers ses voisins via des arcs admissibles jusqu'à ce que $e(i)$ devienne nul ou que l'algorithme modifie l'étiquette de distance sur le nœud.

Le comportement de l'algorithme générique de pré-flot dépend de la manière dont les nœuds actifs sont sélectionnées. Nous présentons deux façons de sélectionner les nœuds actifs qui conduisent à de meilleures complexités théoriques comparativement à l'algorithme générique. Ces alternatives sont:

1. Sélection des nœuds actifs suivant un ordre FIFO (*first in first out*);
2. Sélection des nœuds actifs les plus éloignés du puits en priorité. Dans ce cas l'algorithme choisit le nœud actif possédant la plus grande étiquette de distance.

La première alternative conduit à l'algorithme communément appelé *FIFO Preflow Algorithm*. Cet algorithme a été devéloppé par Goldberg [21] en introduisant le concept de distance dans l'algorithme proposé par Shiloach & Vishkin [40]. Le *FIFO Preflow Algorithm* examine les nœuds actifs suivant un ordre premier venu premier servi. L'algorithme maintient une liste des nœuds actifs gérée comme une liste. Il sélectionne le nœud à examiner à partir de la tête de la liste et ajoute au

fur et à mesure les nouveaux nœuds actifs à la fin de cette liste. Quand l'algorithme sélectionne un nœud, il l'examine tant qu'il est actif et son étiquette de distance n'a pas changée. Si l'étiquette de distance change, le nœud est ajoutée à la fin de la liste. Le critère d'arrêt de cet algorithme est l'obtention d'une liste vide des nœuds actifs. La complexité théorique du *FIFO Preflow Algorithm* est $O(n^3)$. Cette complexité est meilleure que celle de l'algorithme générique $O(n^2m)$.

La deuxième alternative conduit à la version de l'algorithme générique appelée *Highest Label Preflow Algorithm.* Cette version a été proposée par Goldberg & Tarjan [22] et sa complexité est de $O(n^2m^{\frac{1}{2}})$. L'idée de base de cet algorithme est de toujours sélectionner le nœud actif le plus éloigné du puits pour examen. Pour opérer cette sélection, sans trop d'effort, l'algorithme maintient un tableau de dimension $2n-1$, où chaque élément d'indice k dans ce tableau est constitué de la liste des nœuds actifs à la distance k du puits du graphe. Appelons ce tableau TAB; on a alors TAB[k]=$\{i : i$ est actif et $d(i) = k\}$ avec $k = 1, \ldots, 2n-1$. Pour retrouver à l'aide de cette structure le nœud actif à sélectionner, il suffit de maintenir dans une variable, qu'on appellera **level** une borne supérieure sur la plus grande valeur de l'indice k correspondant à une liste non vide. En examinant successivement TAB[**level**],TAB[**level** -1], etc., jusqu'à l'obtention d'une liste non vide TAB[p], on sélectionne un nœud quelconque dans TAB[p] et on pose **level** = p. Si durant l'examen d'un nœud i, la distance $d(i)$ change, on remet à jour la structure TAB et la valeur de la variable **level**.

2.4 Améliorations pratiques

Nous avons montré dans la première section de ce chapitre que la résolution du problème de flot maximum sur le graphe étendu G^e a pour but la détermination d'une

coupe minimale. Dans les trois algorithmes que nous avons exposés, cette coupe minimale peut être connue avant l'atteinte des critères d'arrêt spécifiés. Cette situation peut être exploitée pour réduire le temps de traitement des problèmes. Considérons par exemple, *l'algorithme du chemin augmentant minimal* dont le critère d'arrêt est $d(s) \geq n$. Cet algorithme, après avoir acheminé un flot maximum de s à t, met beaucoup de temps à changer les étiquettes de distance sur les nœuds pour arriver à satisfaire son critère d'arrêt. De même les algorithmes de pré-flot qui, après avoir acheminé le maximum de flot au puits, passe du temps supplémentaire pour ramener les surplus sur les nœuds vers la source du graphe. Pour détecter et mettre à profit ce genre de situation, Ahuja, Orlin & Magnanti [2] proposent l'utilisation d'un tableau $numb[k] : k = 1, \ldots, n-1$ où la valeur de $numb[k]$ est le nombre de nœuds dont la distance exacte par rapport au puits du graphe est k. Le tableau $numb$ est initialisé au début de chaque algorithme en même temps que l'initialisation des distances exactes. Il se modifie à chaque fois que l'algorithme change l'étiquette de distance sur un nœud d'une valeur $k1$ à une valeur $k2$ en ajoutant 1 à $numb[k2]$ et en retranchant 1 à $numb[k1]$. Pour terminer l'algorithme, il suffit de vérifier à cette étape que $numb[k1] = 0$. La présence d'une telle situation signifie que la source et le puits du graphe ne sont plus reliés par un chemin admissible dans le graphe résiduel, d'où la présence d'une coupe minimale $(S, \overline{S})$ définie comme suit: $S = \{i : d(i) > k1\}$ et $\overline{S} = \{i : d(i) < k1\}$.

Nous utilisons cette technique comme critère d'arrêt dans tous les algorithmes que nous avons implémenté. De même, nous avons constaté à l'aide de tests que nous ne reproduirons pas ici, que conformément à la structure des graphes tirés des problèmes miniers, il est avantageux de donner priorité aux arcs originaux sortants d'un nœud dans les recherches des arcs admissibles. Ce choix a été retenu dans chacune des implémentations.

D'autres améliorations spécifiques à chaque famille d'algorithmes ont été aussi retenues.

- Dans l'implémentation de *l'algorithme du chemin augmentant minimal*, nous conservons pour chaque nœud le premier arc à explorer. Ceci permet d'éviter des tests redondants d'arcs admissibles.

- Dans les deux algorithmes de pré-flot, nous avons retenu les points suivants:

  - Ne jamais sélectionner un nœud actif dont l'étiquette de distance est supérieure ou égale au nombre de nœuds du graphe.

  - Continuer de traiter le *nœud courant* tant qu'il est actif, même si son étiquette de distance change. Ceci permet de sauver des opérations de mise à jour de la structure du tableau de listes utilisé dans le *Highest Label Preflow Algorithm.* De même il assure une nette amélioration dans le cas du *FIFO Preflow Algorithm.*

  - Conserver la plus petite distance lors de la recherche d'arcs admissibles dans la procédure **Push_relabel** afin d'accélérer la mise à jour de la distance du *nœud courant* si on ne trouve pas d'arcs admissibles de trouvés.

2.5 Comparaison des algorithmes

Dans cette section, nous comparons les trois algorithmes de flot maximum exposés précédemment sur des graphes présentant une structure semblable à celle des graphes modélisant les gisements miniers. Considérons un gisement minier constitué de n blocs répartis sur plusieurs niveaux. Ce gisement se représente par un graphe qui associe un nœud à chaque bloc du gisement et un arc à chaque relation de préséance entre deux blocs. Dans l'espace à trois dimensions, chaque nœud i de ce graphe possède au maximum neuf successeurs qui correspondent aux neuf blocs

dont l'extraction doit être effectué avant celle du bloc correspondant au nœud i. Les nœuds qui correspondent aux blocs à la frontière du gisement ont moins de neuf successeurs. Notons que tous les arcs correspondant aux relations de préséance sont munis de capacités infinies. De même chaque nœud correspondant à un bloc est relié par un arc de capacité finie soit à la source, soit au puits.

Nous considérons des problèmes tests où le gisement possède une forme régulière de parallélépipède dont la base est un rectangle présentant x blocs d'un côté et y blocs de l'autre côté. La hauteur z est définie par le nombre de niveaux dans le gisement. Pour valoriser les blocs dans ce gisement, nous générons leurs valeurs aléatoirement sur un intervalle de -10.0 à 10.0 avec deux chiffres significatifs.
Dans une première phase, nous examinons le comportement des différents algorithmes en fonction de la forme du gisement. Par la suite nous allons analyser leurs comportements en fonction de la répartition des valeurs des blocs. Dans ce cas, nous considérons des gisements où les niveaux sont subdivisés de façon alternée en p niveaux constitués uniquement de blocs à valeurs positives et de p niveaux constitués uniquement de blocs à valeurs négatives. Ce genre de graphe simule plus ou moins la présence de strates de minerai et strates de stérile dans les vrais gisements. Habituellement les strates dans les gisements ne sont pas horizontales, elles possèdent une certaine inclinaison par rapport à l'horizontale. Nous les avons supposé horizontales par souci d'uniformité des problèmes tests. A la fin, nous comparons les trois algorithmes sur les données réelles du gisement du Mont Wright exploité par la compagnie Québec Cartier Mining. Puis nous étudions sur ce gisement l'effet de la variation des valeurs des blocs sur le comportement des algorithmes.

Dans la suite du texte, nous utilisons les appellations suivantes pour désigner chacun des algorithmes:

SAPA : *Shortest Augmenting Path Algorithm*;

FFPA : *FIFO Preflow Algorithm*;

HLPA : *Highest Label Preflow Algorithm.*

Les résultats présentés dans les tableaux 2.1, 2.2 et 2.3 représentent les moyennes des résultats obtenus sur cinq problèmes tests de même caractéristique. Notons que tous les tests ont été conduits sur une machine HP 9000/735 (124 Mips, 40 Mflops). Le tableau 2.1 présente le temps moyen en secondes des différents algorithmes en fonction du nombre de niveaux dans le graphe. Nous avons considéré des graphes dont la base est de 40 par 40 blocs et nous avons fait varier le nombre de niveaux de 10 à 70. Les graphes de 70 niveaux présentent 112 000 nœuds et près de 1 120 000 arcs.

Tableau 2.1 Temps d'exécution moyen *(en secondes)* en fonction du nombre de niveaux du graphe.

	Nombre de niveaux						
Algorithmes	10	20	30	40	50	60	70
SAPA	11.67	30.08	54.32	59.28	106.42	60.22	144.47
FFPA	12.90	40.06	79.28	95.24	157.91	171.79	287.63
HLPA	7.81	24.41	43.12	39.73	66.31	72.42	127.36

Nous remarquons dans ce tableau la nette supériorité de l'algorithme HLPA sur les deux autres. L'algorithme FFPA se détache comme le moins performant des trois. Le SAPA, tout en restant dans la majorité des cas moins performant que le HLPA, reste de loin meilleur que le FFPA. Les trois algorithmes présentent une croissance du temps de calcul en fonction du nombre de nœuds de façon régulière. Le taux de cette croissance est de loin inférieur au taux de croissance auquel on peut s'attendre théoriquement. Une observation des temps présentés dans ce tableau montre que

si t_1 est le temps requis par l'un des algorithmes pour résoudre le problème de flot maximum sur l'un des graphes à 10 niveaux, alors le temps requis pour un graphe ayant $10k$ niveaux est de l'ordre de $2kt_1$. Ce comportement est fort intéressant, dans le sens qu'en pratique les temps pris par ces algorithmes sur le type de graphe qui nous intéresse ne croissent pas rapidement si le nombre de nœuds du graphe augmente. Cette croissance du temps de calcul en fonction du nombre de nœuds est linéaire plutôt que cubique tel qu'établi théoriquement.

Le tableau 2.2 présente le temps moyen d'exécution des algorithmes sur des graphes ayant 30 niveaux et dont la base est variable. Nous considérons que l'un des côtés de cette base est composé de $60/x$ blocs et l'autre côté est composé de $10x$ blocs, pour x variant de 1 à 5. Les graphes présentent 18 000 nœuds et environ 180 000 arcs. Les résultats de ce tableau ne font pas ressortir un comportement partic-

Tableau 2.2 Temps d'exécution moyen *(en secondes)* en fonction de la forme de la base du graphe.

	Paramètre x				
Algorithmes	1	2	3	4	5
SAPA	13.33	10.19	11.41	10.75	10.76
FFPA	21.61	16.91	20.42	16.70	17.50
HLPA	12.43	8.79	10.34	8.83	9.03

ulier en fonction de la forme de la base du graphe. Les problèmes tests présentant le même nombre de nœuds et le même nombre d'arcs, le temps d'exécution moyen des algorithmes est resté sensiblement le même quelque soit la forme de la base du graphe. Ce tableau confirme cependant, comme le tableau précédent, la supériorité du *Highest Label Preflow Algorithm* sur les autres.

Pour simuler la présence de strates dans les gisements, considérons maintenant des graphes dont la base est constituée de 40 par 40 blocs et fixons le nombre de niveaux à 40. Ces graphes présentent 64 000 nœuds et environ 640 000 arcs. Dans ces graphes nous valorisons les blocs en alternant p niveaux de blocs à valeurs positives avec p niveaux de blocs à valeurs négatives. Le tableau 2.3 présente le temps moyen d'exécution des différents algorithmes en fonction de l'épaisseur des strates p. La colonne correspondant à $p = 0$ donne les temps pour des graphes sans strate. Ce tableau montre qu'en général la présence de strates est favorable à la réduction

Tableau 2.3 Temps d'exécution moyen *(en secondes)* en fonction de l'épaisseur des strates.

	Épaisseur des strates p					
Algorithmes	0	2	5	8	10	20
SAPA	59.28	38.63	33.08	24.72	30.72	34.37
FFPA	95.24	97.27	93.22	58.39	71.99	29.92
HLPA	39.73	45.40	41.46	34.75	41.66	30.32

des temps de calcul pour tous les algorithmes même si cette réduction n'est pas très importante. L'algorithme SAPA est dans plusieurs cas plus performant que le HLPA. Ceci peut être expliqué par le fait que dans ce genre de graphes stratifiés, les longueurs des chaînes augmentantes sont en moyenne de deux fois l'épaisseur des strates. Ces chaînes sont donc relativement courtes et faciles à identifier, ce qui naturellement est favorable au fonctionnement de l'algorithme SAPA.

Pour voir le comportement des algorithmes sur des données réelles, nous avons considéré le gisement du Mont Wright exploité par la Compagnie Québec Cartier Mining. Ce gisement est constitué de 147 359 blocs réparties sur 43 niveaux. Nous connaissons pour chaque bloc de ce gisement, sa teneur, sa densité et son poids

de récupération. En se basant sur ces informations, une valeur c_i a été associée à chaque bloc i. En considérant un paramètre r, on fait varier les valeurs des blocs comme suit : $c_i(r) = c_i - r|c_i|$. Le tableau 2.4 présente le comportement des algorithmes en fonction de la variation des valeurs sur les blocs en considérant les valeurs du paramètre r allant de 0 à 0.4, par pas de 0.05. Cette modification des valeurs des blocs simule l'effet des multiplicateurs duaux sur la résolution des problèmes de flot maximum lors de la résolution du problème dual obtenu par relaxation des contraintes de ressource. Les résultats de ce tableau montre de manière éloquente

Tableau 2.4 Temps d'exécution *(en secondes)* en fonction de la valorisation des nœuds; cas du gisement Mont Wright.

	Paramètre r								
Algorithmes	0	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
SAPA	51.34	42.39	59.64	52.10	59.21	38.47	34.73	33.58	34.79
FFPA	127.67	92.45	126.07	109.75	114.79	122.92	88.00	103.81	93.29
HLPA	18.85	15.91	15.05	15.65	16.17	15.32	14.41	15.22	15.09

que sur les données réelles du gisement du Mont Wright, le HLPA est nettement le meilleur algorithme. Son temps est seulement $\frac{1}{3}$ du temps de l'algorithme SAPA. L'influence de la variation des valeurs des blocs sur la performance des algorithmes est plus difficile à mettre en évidence. Une augmentation de la valeur du paramètre r se traduit sur le graphe par une diminution des capacités des arcs sortant de la source et une augmentation des capacités des arcs arrivant au puits. Cette situation n'est pas supposée affecter de manière sensible le comportement des algorithmes. Néanmoins elle peut diminuer le temps mis par l'algorithme SAPA dans la recherche de chaînes augmentantes et réduire le nombre de chaînes augmentantes ne saturant pas un arc venant de la source ou arrivant au puits. Pour les algorithmes de pré-flot, cette situation peut favoriser une diminution du temps durant lequel

un nœud peut rester actif dans le graphe résiduel. Ce qui peut être favorable à une réduction du temps de calcul. Globalement les résultats de ce tableau montrent une légère diminution non régulière des temps en fonction d'une augmentation de la valeur du paramètre r pour l'algorithme SAPA. L'algorithme HLPA montre une légère diminution au début puis garde sensiblement les mêmes temps pour les valeurs de r allant de 0.25 à 0.4. Notons que l'algorithme HLPA, plus stable avec la variation des capacités sur les arcs, reste toujours meilleur que l'algorithme SAPA.

2.6 Conclusion

En comparant les résultats des différents algorithmes, nous avons retenu le *Highest Label Preflow Algorithm* comme algorithme de résolution du problème de la fermeture maximale sur les graphes représentant des gisements miniers.

Avec cet algorithme, le problème de la fermeture maximale, équivalent en littérature minière au problème des contours finaux, est résolu en moyenne en 15 secondes sur des graphes de 150 000 nœuds et environ 1 500 000 arcs. Le problème des contours finaux intervient comme sous-problème dans le schéma de relaxation lagrangienne adopté pour le traitement du problème de planification multi-périodes de l'exploitation dans un gisement minier. Sa résolution rapide conditionne la réussite d'une telle approche. Avec l'algorithme retenu, le sous-problème des contours finaux peut être résolu une centaine de fois en moins de 30 minutes. Ainsi on peut s'attendre, avec l'utilisation de cet algorithme, à la résolution du problème de planification en un temps raisonnable.

CHAPITRE 3

Algorithme de décomposition pour le problème de planification dans une mine à ciel ouvert

3.1 Présentation

En l'absence des contraintes de ressource, le problème de planification multi-périodes de l'exploitation d'une mine à ciel ouvert est un problème de fermeture maximale sur un réseau multi-périodes. La résolution efficace de ce problème conditionne fortement le succès de l'approche de relaxation lagrangienne qu'on emploiera pour le traitement du problème global de planification. Dans une application réelle présentant une centaine de milliers de blocs dont l'exploitation est à planifier sur un horizon de 10 périodes, le réseau en jeu présente 1 million de nœuds et une dizaine de millions d'arcs. Ainsi la détermination d'une fermeture maximale sur un tel réseau par un algorithme de flot maximum est quasiment impossible à cause de la taille mémoire nécessaire pour son stockage.

Ce chapitre qui présente l'article *A Decomposition Flow Algorithm for the Operations Planning Problem in Open Pit Mines* est une tentative de résolution de ce problème par décomposition du réseau. Cette approche décompose le réseau en relaxant tous les arcs inter-périodes. Elle se ramène ainsi à P graphes de tailles raisonnables où P est le nombre de périodes. Cette décomposition tient compte des arcs relaxés par transfert de capacités résiduelles d'un graphe correspondant à une période au graphe correspondant à la période suivante.

L'algorithme proposé dans cet article peut trouver la solution optimale sous certaines conditions, mais dans la plupart des cas, ces conditions ne sont pas vérifiées. Néanmoins les solutions obtenues sont très proches des solutions optimales. L'avantage majeure d'une telle décomposition est l'économie d'espace mémoire et du temps de calcul réalisée comparativement à un traitement direct du réseau, lorsque possible, par un algorithme de flot.

Cet article présente la structure du problème et l'algorithme de décomposition. De même, il présente des résultats comparatifs entre le traitement direct de quelques réseaux de petites tailles par un algorithme de flot et leurs traitement par l'algorithme de décomposition proposé.

3.2 A Decomposition Flow Algorithm for the Operations Planning Problem in Open Pit Mines

Beyime Tachefine and François Soumis
GERAD and École Polytechnique de Montréal
5255 avenue Decelles, Montréal, Canada, H3T 1V6

abstract

In operations planning in an open pit mine an economic blocks model is generally used which attempts to determine which blocks will be extracted during each period. Lagrangean relaxation of the capacity constraints leads to a maximal flow problem on a multi-period network. The best known method, proposed by Dagdeleen, is a heuristic method of decomposition in periods. We propose an algorithm based on the idea of Dagdeleen which yields an optimal solution provided that some conditions are met. When the solution is not optimal, it is an excellent lower bound. Numerical results show the substantial gain in time achieved by this algorithm as well as the quality of the solution.

résumé

Une mine à ciel ouvert peut être représentée par un ensemble de blocs et le problème de planification des opérations consiste à déterminer quels blocs seront extraits à chaque période de façon à maximiser le profit tout en respectant les limites de capacité. La relaxation lagrangienne des contraintes de capacité amène à résoudre un problème de flot maximal dans un réseau multi-périodes de très grande taille. Une méthode heuristique de décomposition par période a été proposée par Dagdeleen. Cet article présente un algorithme de décomposition qui produit une solution optimale quand certaines conditions sont satisfaites. Quand la solution n'est pas optimale, elle constitue une excellente borne inférieure. Des résultats numériques illustrent le gain de temps substantiel obtenu en utilisant la méthode de décomposition au lieu de traiter directement le problème de flot maximal.

3.2.1 Introduction

The problem of operations planning in an open pit mine is of practical importance in the mining industry. To this day, the research efforts aiming at devising an optimal algorithm for this problem have not been successful. The contributions made to the study of this problem ranged from the introduction of the concept of tridimensional economic blocks model, that allows the modeling of a mine deposit, to the mathematical formulation of the problem and attempts to solve it. Operations planning in an open pit mine consists of determining which blocks will be extracted during each period of the planning schedule. For each period the economic blocks model assigns values to each block in terms of its geological, physical and economical features. The choice of the blocks to be selected is made in order to maximize the income generated by the operations during the entire planning horizon, while satisfying in the same time all the constraints of the production system.

The optimization techniques in mining operations planning where first introduced following the works of Lerchs and Grossman [4] on an open pit mine contour optimizing algorithm. This algorithm amounts to determining the maximal closure on a graph, which is equivalent, as Picard [5] proved, to the search for a maximum flow on a new graph that can be derived from the initial one. These works permit to develop softwares able to solve optimally large scale real-life contour problems. In the light of the exact algorithms developed for the open pit mine contour, the operations planning problem was formulated as an integer linear program using 0–1 binary variables and its particular structure was pointed out by Dagdeleen [2]. Unfortunately, the size of the model for a planning problem is larger than for a contour problem. The size is multiplied by the number of periods.

Following these efforts, many solution methods were proposed for solving the operations planning problem, the most recent being that of Dagdeleen. This method uses a Lagrangean relaxation approach to reduce the problem to another one that will be called in this text *the operating sequence problem.* Dagdeleen proposed a decomposition method for solving this operating sequence problem, however the method turned out not to be optimal.

In this paper, we will first present a formulation of the operations planning problem, then derive the operating sequence problem using a Lagrangean relaxation approach. Furthermore we show that this later problem is equivalent to a maximum flow problem in a multi-period network. The main contribution of this paper is an algorithm of decomposition of the maximum flow problem that is guaranteed to be optimal provided that some conditions are met. In addition, this decomposition algorithm allows a substantial gain in both execution time and storage space requirements compared to a traditional algorithm of maximum flow applied to the entire network problem.

In Section 3.2.2, we present the formulation of the problem and the decomposition algorithm in Section 3.2.3. Sections 3.2.4 and 3.2.5 respectively discuss the optimality of the algorithm and present numerical results with conclusions.

3.2.2 Problem Formulation

In modelling the extraction of a mineral deposit, an economic blocks model is generally used. This model consists of partitioning the deposit into parallelipedic blocks and a determination for each block of certain characteristic informations (mineral, metallurgical, topographical and economic information). Operations planning in this context amounts to an attempt to determine which blocks will be extracted

during each period of the planning horizon. This decision should maximize income, while respecting the following constraints:

- Capacity constraints, which limit the quantities which may be extracted during each period;
- Geometric constraints, which require that extraction slopes be respected and which guarantee that no blocks may be extracted unless it is uncovered. The geometric constraints may be interpreted as precedence constraints between blocks which indicate that a block may only be extracted if a set of blocks which constrain it physically have already been extracted. This set of precedence constraints can be summarized by a directed graph in which the nodes represent blocks and in which there exists an arc from node i to node j if block j constrains block i. For example, consider a two-dimensional representation of nine blocks in which in order to extract a block it is necessary to have previously extracted the block immediately above it and the two neighboring blocks on either side of it. Such a block layout and the precedence graph associated with it are illustrated in Figure 3.1.

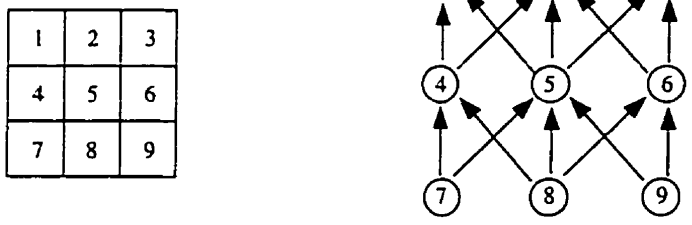


Figure 3.1 Block layout and associated precedence graph

We define the following notation, which will be used in the formulation of the planning problem.

n : number of blocks;

i : index of blocks;

P : number of periods;

p : index on periods;

c_i^p : income generated by block i in period p (this income can be positive, negative or null);

C^p : $(c_1^p, c_2^p, \ldots, c_n^p)$ income vector for period p;

x_i^p : activity variable which takes the value 1 if block i is extracted at period p and 0 otherwise;

X^p : $(x_1^p, x_2^p, \ldots, x_n^p)^T$ activity variable vector for period p;

E : transpose of the node-arc incidence matrix for the graph representing precedence relations;

A : matrix of coefficients for the capacity constraints;

B^p : vector of upper bounds on capacity for period p;

$\mathbb{1}$: n-dimensional unit vector;

Using this notation, the planning problem can be formulated as follows:

$$\max C^1X^1 + C^2X^2 + \ldots + C^PX^P \tag{3.1}$$

$$\left.\begin{array}{rlll} \text{s. t.} & AX^1 & & \leq B^1 \\ & & AX^2 & \leq B^2 \\ & & \ddots & \vdots \\ & & AX^P & \leq B^P \end{array}\right\} \tag{3.2}$$

$$\left.\begin{array}{rcl} EX^1 & & \leq 0 \\ EX^1 + EX^2 & & \leq 0 \\ \vdots & & \\ EX^1 + EX^2 + \ldots + EX^P & & \leq 0 \\ X^1 + X^2 + \ldots + X^P & & \leq \mathbb{1} \end{array}\right\} \tag{3.3}$$

$$\left.\begin{array}{l} X^p \geq 0, \quad p = 1, \ldots, P \\ X^p \in \{0,1\}^n, \quad p = 1, \ldots, P \end{array}\right\} \tag{3.4}$$

In this formulation, the objective function (3.1) indicates that operating income is to be maximized, subject to three sets of constraints. Constraint set (3.2) includes the capacity constraints and constraint set (3.3) includes precedence constraints for each period p and a constraint which requires that any block not be extracted more than once. The non negativity and binary constraints are provided in (3.4).

The proposed solution approach for the problem (3.1)–(3.4) is the Lagrangean relaxation of the capacity constraints (3.2), which allows a solution of a reduced problem called the *operating sequence problem*. For this problem, the reduced costs $\bar{C}^p$ $(p = 1, \ldots, P)$ are defined as: $\bar{C}^p = C^p - \lambda^p A$, where λ^p $(p = 1, \ldots, P)$ are the lagrangian multipliers vectors associated with constraints (3.2). Thus the objective of this problem consists in maximizing a function based on the reduced costs:

$$\max \bar{C}^1 X^1 + \bar{C}^2 X^2 + \ldots + \bar{C}^P X^P \tag{3.5}$$

The operating sequence problem (3.5) subject to (3.3) and (3.4) will be taken up as a maximal flow problem on a multi-period, acyclic network. In order to highlight the structure of this problem, we carry out the following change of variables:

$$W^p = \sum_{k=1}^{p} X^k$$

The problem then becomes:

$$\max \overline{\bar{C}}^1 W^1 + \overline{\bar{C}}^2 W^2 + \ldots + \overline{\bar{C}}^P W^P \tag{3.6}$$

$$\left.\begin{array}{rcccccccl} \text{s. t.} & EW^1 & & & & & & & \leq 0 \\ & & EW^2 & & & & & & \leq 0 \\ & & \vdots & & \ddots & & & & \vdots \\ & & & & & & & EW^P & \leq 0 \\ & W^1 & - & W^2 & & & & & \leq 0 \\ & & & W^2 & - & W^3 & & & \leq 0 \\ & & & & & \ddots & & & \leq 0 \\ & & & & & & W^{P-1} & - \quad W^P & \leq 0 \end{array}\right\} \tag{3.7}$$

$$W^p \in \{0,1\}^n, \quad p = 1, \ldots, P. \tag{3.8}$$

The cost $\overline{\bar{C}}_i^p$ is defined by $(\bar{C}_i^p - \bar{C}_i^{p+1})$ for $p = 1, \ldots, P-1$ and by $\bar{C}^P$ for period P. The constraint matrix in (3.7) represents the transpose of the node-arc incidence matrix of a multi-period network. This problem can be represented by a network such as the one presented in Figure 3.2, where the sign of the cost values have been arbitrarily given.

Solving the problem (3.6) subject to (3.7) and (3.8) is equivalent to finding a maximal closure of this network. The work of Picard [5] shows that this problem is in turn equivalent to a problem of maximal flow on a network G constructed from the multi-period precedence network to which a source node s and sink node t have been added. In addition, each vertex i/p such that $\overline{\bar{c}}_i^p \leq 0$ is connected to the sink t by an arc $(i/p, t)$ with capacity $u_i^p = -\overline{\bar{c}}_i^p$, and each vertex i/p such that $\overline{\bar{c}}_i^p > 0$ is connected to the source s by an arc $(s, i/p)$ with capacity $u_i^p = \overline{\bar{c}}_i^p$. The capacities on the other arcs, which neither have their head at t or their tail at s, are considered infinite.

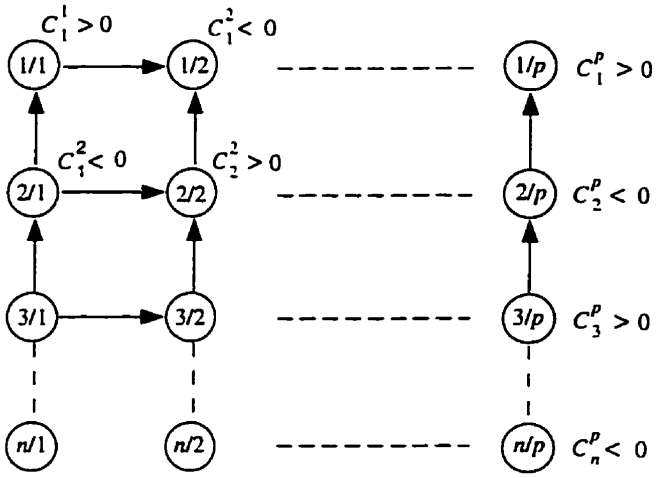


Figure 3.2 Multi-period Precedence Network

The operating sequence problem can thus be reduced to the maximal flow problem in the multi-period network G.

3.2.3 Decomposition algorithm

Consider the multi-period network described in Section 3.2.2 and let G^p denote the graph corresponding to period p. The graph G^p is made up of all nodes i/p, the source s, and the sink t, and all arcs incident to only these nodes. The algorithm which we propose solves the maximal flow problem for each graph G^p and takes account of inter-period arcs $(i/p, i/p+1)$ by modifying capacities on the arcs in graph G^{p+1}, which corresponds to period $p+1$.

In order to present the steps of the algorithm explicitly, we use the following notation:

u_i^p : capacity of arc $(s, i/p)$ or of arc $(i/p, t)$ in the original network G,

f_i^p : optimal flow on arc $(s, i/p)$ or on arc $(i/p, t)$ in graph G^p,

$\bar{u}_i^p$: modified capacity of arc $(s, i/p)$ or of arc $(i/p, t)$ in graph G^p,

S^p : the set of indices i such that node i/p is labelled on the last iteration of the maximal flow algorithm on graph G^p,

M^p : the set of indices i such that node i/p is considered in the graph G^p.

Steps of the Decomposition algorithm:

Step 1: Initialization

- $M^p = \{1, 2, 3, \ldots, n\}; \quad p = 1 \ldots P$
- $\bar{u}_i^1 = u_i^1, \quad i \in M^1$
- $p = 0$

Step 2: Solution of the maximum flow problem on G^p

- $p = p + 1$
- solve the maximum flow problem for graph G^p taking into account modified capacities $\bar{u}_i^p$
- form S^p
- If $p = P$ go to Step 4.

Step 3: Computation of modified capacities for G^{p+1}

- Calculate modified capacities for graph G^{p+1}
  - for any arc $(s, i/p+1)$ such that $i \in S^p$, let $\bar{u}_i^{p+1} = u_i^{p+1} + (\bar{u}_i^p - f_i^p)$
  - for any arc $(i/p+1, t)$ such that $i \in S^p$, let $\bar{u}_i^{p+1} = u_i^{p+1} - (\bar{u}_i^p - f_i^p)$

(in this latter case if $\bar{u}_i^{p+1}$ is negative, arc $(i/p+1, t)$ becomes an arc $(s, i/p+1)$ with capacity $-\bar{u}_i^{p+1}$)

- for the other arcs $(s, i/p+1)$ or $(i/p+1, t)$ such that $i \notin S^p$, let $\bar{u}_i^{p+1} = u_i^{p+1}$

- Go to Step 2.

Step 4: Stopping criterion

- If $S^1 \subseteq S^2 \subseteq \ldots \subseteq S^P$, the solution has been found;
- Otherwise, for $p = 1, \ldots P$, let $M^p = \cap_{j=p}^{P} S^j$. Set $p = 0$ and go to Step 2, replacing graphs G^p $(p = 1, \ldots, P)$ by the subgraphs obtained by eliminating nodes i/p for $i \in N - M^p$.

3.2.4 Discussion of the optimality

The proposed algorithm uses a decomposition of the maximum flow problem on a multi-period network. A theoretical proof of the optimality of the flows obtained for the networks satisfying the termination condition of Step 4 at the first iteration of the algorithm has been established, Tachefine [6]. At each iteration, the algorithm removes all vertices that do not meet the inclusion criteria in order to ensure that the termination condition is ultimately achieved. This approach, while not unique, gives good results and leads to bounds that are relatively close to the optimal solution obtained by a direct application of the Maximum Flow algorithm on the entire network.

The major difference between this algorithm and that of Dagdeleen [2] is in the way capacities are modified (Step 3 of the algorithm). Dagdeleen's method proceeds as follows:

- for all $i \in S^p$,

$$\bar{u}_i^{p+1} = u_i^{p+1} + u_i^p \quad \text{if} \quad \begin{cases} u_i^p \text{ is the capacity of an arc } (s, i/p) \\ \text{and } u_i^{p+1} \text{ is the capacity of an arc } (s, i/p+1) \\ \text{OR} \\ u_i^p \text{ is the capacity of an arc } (i/p, t) \\ \text{and } u_i^{p+1} \text{ is the capacity of an arc } (i/p+1, t) \end{cases}$$

$$\bar{u}_i^{p+1} = u_i^p - u_i^{p+1} \quad \text{if} \quad \begin{cases} u_i^p \text{ is the capacity of an arc } (s, i/p) \\ \text{and } u_i^{p+1} \text{ is the capacity of an arc } (i/p+1, t) \end{cases}$$

$$\bar{u}_i^{p+1} = u_i^{p+1} - u_i^p \quad \text{if} \quad \begin{cases} u_i^p \text{ is the capacity of an arc } (i/p, t) \\ \text{and } u_i^{p+1} \text{ is the capacity of an arc } (s, i/p+1) \end{cases}$$

(In the latter two cases, an arc $(s, i/p+1)$ or $(i/p+1, t)$ with negative $\bar{u}_i^{p+1}$ is replaced by, respectively, an arc $(i/p+1, t)$ or $(s, i/p+1)$ with capacity $-\bar{u}_i^{p+1}$.)

- For any $i \notin S^p$, $\bar{u}_i^{p+1} = u_i^p$.

While the Dagdeleen's method has the merit of introducing the idea of decomposition, it does not guarantee the optimality of the solution, and may even leads to unfeasible solutions for the network problem. For instance, consider the layout of 4 blocks, valued over two periods, for which we seek to solve the operating sequence problem (Figure 3.3).

In a node labelled x/y, x is the block number according to the layout presented in Figure 3.3 and y corresponds to the period. The numbers associated to arcs represent the capacities. These values are obtained from the block values with

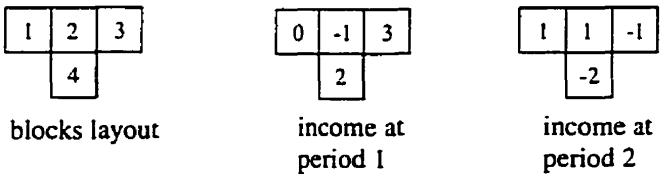


Figure 3.3 Blocks and values of an example problem

the approach presented in Section 3.2.2. This problem can be modeled by the network of Figure 3.4. The Dagdeleen's method applied to this example leads to an

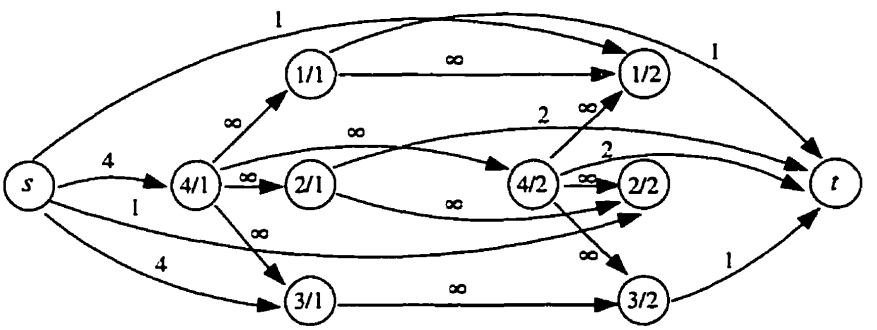


Figure 3.4 Network corresponding to the example problem

unfeasible solution for the flow problem. This solution with a value of 4, translates into a cut $C(R, R')$ with: $R' = \{t\}$ and $R = \{s, 1/1, 2/1, 3/1, 4/1, 1/2, 2/2, 3/2, 4/2\}$ and corresponds to the extraction of the 4 blocks in the first period.

Our Decomposition algorithm, finds a maximum flow of value 5, corresponding to the cut $C(R, R')$ with $R = \{s, 3/1, 1/2, 2/2, 3/2\}$ and $R' = \{1/1, 2/1, 4/1, 4/2, t\}$. This solution corresponds to the extraction of block #3 in the first period and the extraction of blocks #1 and #2 in the second period.

3.2.5 Numerical results and conclusions

An implementation of the algorithm was tested using a mine deposit consisting of 4 levels of 20 × 16 blocks each (that is, a deposit of 1280 blocks). The incomes,

generated by the blocks during each period, was obtained using a cost model that computes the future values of the blocks. We added to this value a random proportion to account for factors other than the interest rate. For example, a block i with income c_i^1 in the first period, generates at period n an income given by the following formula:

$$c_i^n = \frac{c_i^1}{(1+t)^{n-1}}(1+k\theta)$$

in which t is the interest rate at period n, θ is a random number $\in [-1, 1]$, and $k \in [0, 1]$ is a parameter that indicates the percentage of income value randomly allocated.

We present the numerical results obtained for 3 groups of problems whose block generated incomes where computed for the following values of the parameter k : $0.1; 0.5; 1.0$. The results obtained by using our Decomposition algorithm and the Ford & Fulkerson labeling Maximum Flow algorithm are compared in terms of execution time and quality of solution.

Problem	Number of vertices	Maximum Flow Algorithm$^{(MF)}$ CPU time (sec.)	Maximum Flow Algorithm$^{(MF)}$ Solution value	Decomposition Algorithm$^{(D)}$ CPU time (sec.)	Decomposition Algorithm$^{(D)}$ Solution value	Cost variation ((MF)-(D))/(MF)	Execution time ratio (D)/(MF)
Group # 1 $k = 0.1$ $t = 0.1$							
1	1280	16.4	15311	15.1	15311	0	0.92
2	2560	98.9	15317	29.6	15317	0	0.30
3	3840	376.3	15321	79.7	15314	0.03%	0.21
4	5120	1048.3	15321	156.5	15313	0.05%	0.15
5	6400	2073.4	15321	188.3	15310	0.07%	0.09
Group # 2 $k = 0.5$ $t = 0.1$							
6	2560	285.5	15933	131.4	15925	0.05%	0.46
7	3840	1590	16243	331.3	16231	0.07%	0.21
8	5120	4731.0	16398	711.2	16248	0.9%	0.15
9	6400	10939.0	16443	973.9	16216	1.4%	0.09
Group # 3 $k = 1.0$ $t = 0.1$							
10	2560	357.2	17098	170.3	17098	0	0.47
11	3840	1873.1	18104	435.5	18090	0.07%	0.23
12	5120	6114.3	18765	1192	18531	1.2%	0.19
13	6400	14501	19120	1217.7	18850	1.4%	0.08

The comparison of the results obtained shows that the Decomposition algorithm proposed has a better execution time than the Maximum Flow algorithm applied to the entire network. This gain in execution time increases with the number of periods; that is, with the size of the network. To illustrate this point, we can look at group 1, which corresponds to a realistic blocks valuation in the deposit. The execution time of problem #5 on a Sun Sparc2 workstation was 3 min. using the Decomposition algorithm as opposed to 34 min. using the Maximum Flow algorithm. The substantial gain in execution time obtained by the Decomposition algorithm suggests the possibility of using this algorithm for solving real world problems in a

reasonnable time. The difference between the solution values is relatively small: it is 1.4% for the problems in group 3 where the structure of the revenus generated by the blocks is entirely random and 0.07% for the problems in group 1 where the revenu structure is more realistic. In the light of these first promising results, we do think that the use of our Decomposition algorithm in a global optimization approach can provide an efficient way of solving the open pit mine operations planning problem while keeping execution time and storage requirements at a minimum.

References

[1] BAZARAA, M. & JARVIS, J.J. (1977). Linear Programming and Networks Flows. *John Wiley & Sons (Eds).*

[2] DAGDELEEN, K. (1985). *Optimum Multi-Period Open Pit Mine Production Scheduling.* Ph.D. Dissertation, Colorado School of Mines Golden, USA.

[3] DAGDELEEN, K. & JOHNSON, T.B. (1986). Optimum Open Pit Mine Production Scheduling by Lagrangian Parametrization. *Proceedings of the 19th APCOM Symposium*, 127–141.

[4] LERCHS & GROSSMAN (1965). Optimum Design of Open Pit Mines. *C.M.I. Bulletin*, **68 no. 633**, 47–54.

[5] PICARD, J.C. (1976). Maximal Closure of a Graph and Applications to Combinatorial Problem. *Management Science*, **22 no 11**.

[6] TACHEFINE, B. (1991). *Détermination du plan d'extraction d'une mine à ciel ouvert.* M.Sc.A. Thesis, École Polytechnique de Montréal, Canada.

3.3 Conclusion

Ce chapitre présente une approche de décomposition au problème des séquences d'exploitation qui est un problème de fermeture maximale sur un réseau multi-périodes de très grande taille. La difficulté principale de ce problème est la prise en mémoire de la structure d'un tel réseau pour y exécuter un algorithme de flot maximum. L'approche de décomposition proposée dans ce chapitre remédie à cette difficulté mais ne garantie pas l'optimalité de la solution. Néanmoins les solutions obtenues par cette approche restent très proches des solutions optimales qui peuvent être obtenues par le traitement de tout le réseau par un algorithme de flot maximum.

Cette tentative de décomposition fait suite aux derniers travaux apparaîssant dans la littérature en vue de résoudre correctement ce problème. Elle est basée sur l'idée de décomposition introduite par Dagdeleen mais elle a l'avantage de proposer de meilleures solutions.

Le fait que l'algorithme de décomposition développé dans ce chapitre ne garantie pas l'optimalité constitue une limitation sérieuse à son utilisation dans l'approche de relaxation lagrangienne que nous utiliserons par la suite pour la résolution du problème global de la planification de l'exploitation dans une mine à ciel ouvert. Ainsi comme on le verra au chapitre 6 de cette thèse, nous avons abandonné l'idée de traiter le problème des séquences d'exploitation tel qu'il est. Nous opterons pour une relaxation des contraintes inter-périodes afin de les traiter de façon heuristique.

CHAPITRE 4

Relaxation lagrangienne et méthodes d'optimisation

4.1 Introduction

Dans cette thèse, nous tentons de traiter les problèmes de la planification de l'exploitation dans une mine à ciel ouvert par une approche de relaxation lagrangienne. L'approche de relaxation lagrangienne est une approche fréquemment utilisée dans la résolution des problèmes formulés en nombres entiers. Elle présente les avantages suivants:

- elle exploite certaines structures intéressantes du problème à résoudre,
- elle permet de déterminer une borne sur la solution optimale; à chaque solution non admissible du problème lagrangien, on peut utiliser une heuristique qui convertit cette solution en une solution réalisable pour le problème de départ; la qualité de la solution retenue est mesurable par l'écart entre la borne obtenue et la valeur de la solution réalisable.

La formulation des problèmes rencontrés en planification minière a été présentée au premier chapitre. Pour éviter toute confusion avec les formulations précédentes, nous utiliserons une formulation générale compacte pour y présenter la relaxation lagrangienne. Cette formulation est traduite par le programme linéaire suivant:

$$
\begin{aligned}
(\mathrm{P}) \quad & \max \quad f(x) \\
& \text{s.c.} \\
& Ax \leq b \\
& x \in D,
\end{aligned}
$$

où x représente le vecteur des variables de décision du problème. Rappelons que ces variables sont binaires et que le domaine D est défini par les contraintes de préséance entre les blocs présents dans le problème de planification. La fonction $f(x)$ est une fonction linéaire des variables de décision du problème et représente l'objectif d'un problème de planification. Le groupe des contraintes $Ax \leq b$ représente les contraintes de ressource et constituent les contraintes liantes du problème. En présence de ces contraintes, le problème reste difficile à traiter. Les contraintes $x \in D$ représentent la structure d'un graphe. En l'absence du premier groupe de contraintes, le problème est un problème de flot sur un réseau dont la taille est fonction du nombre de périodes considérées dans l'horizon de planification. Ce problème de réseau se résout à l'aise des algorithmes de flot maximum. Le chapitre 2 traite ce sujet. Pour exploiter la structure de réseau présente dans le problème P, nous proposons la relaxation lagrangienne des contraintes de ressource. Dans ce chapitre nous présentons cette relaxation lagrangienne et les méthodes d'optimisation que nous avons utilisées pour l'ajustement des multiplicateurs de Lagrange.

4.2 Relaxation lagrangienne

En se basant sur la formulation précédente, associons au groupe des contraintes de ressource le vecteur des multiplicateurs $\lambda \geq 0$. Le lagrangien du problème P s'écrit:

$$L(x, \lambda) = f(x) + \lambda^T(b - Ax) \quad : \quad x \in D.$$

La valeur optimale du lagrangien qu'on note $g(\lambda)$ est appelée fonction duale:

$$g(\lambda) = \max_{x \in D} L(x, \lambda).$$

Le problème du lagrangien $\max_{x \in D} L(x, \lambda)$ est un problème de fermeture maximale sur un réseau. Ce problème possède la propriété d'intégralité dans le sens que, si les contraintes d'intégrité sur les variables sont remplacées par leurs relaxations linéaires, la solution du problème reste entière. La fonction duale $g(\lambda)$ définie pour $\lambda \geq 0$ est une fonction convexe linéaire par morceaux qui admet en tout point λ_i un sous-gradient $s(\lambda_i)$ que nous noterons par la suite s_i, défini par $s_i = b - Ax(\lambda_i)$ où $x(\lambda_i)$ est le vecteur solution obtenu en maximisant le lagrangien $L(x, \lambda)$ au point λ_i. Par le théorème de la dualité faible, on a la propriété que pour tout $\lambda \geq 0$ et pour tout $x(\lambda) \in D$, la valeur de la fonction duale en λ est supérieure ou égale à la valeur de l'objectif de P:

$$\forall\ \lambda \geq 0,\ \ \forall\ x \in D,\ \ g(\lambda) \geq f(x).$$

Comme toute valeur de la fonction duale $g(\lambda)$ constitue une borne supérieure sur la solution optimale de P, la meilleure borne s'obtient donc en minimisant la fonction duale par rapport à λ. Comme le lagrangien possède la propriété d'intégralité, la borne obtenue par l'optimisation de la fonction duale sera de même valeur que la borne obtenue en résolvant la relaxation linéaire du problème P. De même les multiplicateurs de Lagrange correspondent aux multiplicateurs duaux associés à la relaxation linéaire. Le problème d'optimisation de la fonction duale est ce qu'on appelle le dual de P:

$$\min\{g(\lambda)\ \ :\ \ \lambda \geq 0\}.$$

Étant donné la nature de $g(\lambda)$, la résolution du problème dual est généralement faite par une méthode d'optimisation itérative telle que la méthode du sous-gradient ou

celle des plans sécants de Kelley [27]. Nous exposerons dans les sections 2 et 3 de ce chapitre les détails de ces deux méthodes. Nous présentons, en plus de ces deux approches, la méthode de faisceaux sous sa forme duale telle que proposée par Lemaréchal [18].

Notons que dans le cadre de la planification minière, une évaluation de la fonction duale $g()$ consiste à résoudre un problème de flot maximum sur un réseau de très grande taille, ce qui est généralement coûteux en temps de calcul. Il est donc important de choisir une méthode d'optimisation qui évalue le moins possible de fois la fonction $g(\lambda)$. La méthode du sous-gradient comme celle des plans sécants sont généralement lentes et parfois instables dans le sens qu'un point proche de l'optimum peut être suivi d'un point éloigné de l'optimum. Ce genre de comportement nécessite souvent un grand nombre d'itérations et un nombre élevé d'évaluations de la fonction duale avant l'atteinte de l'optimum recherché. La méthode de faisceaux a été développée théoriquement comme une méthode stable qui doit permettre l'atteinte de l'optimum en un nombre limité d'évaluations.
Nous comparons la méthode du sous-gradient avec la méthode de faisceaux dans l'article présenté au chapitre 5 en les évaluant sur des problèmes de planification minière. Dans les sections suivantes, nous présentons les détails algorithmiques de ces trois approches que nous aurons à utilisér ultérieurement.

4.3 Méthode du sous-gradient

Pour minimiser la fonction $g(\lambda)$, l'approche classique est celle du sous-gradient. Cette méthode consiste en un processus itératif qui part d'un point initial λ_1 et qui à l'itération k détermine un sous-gradient de la fonction $g(\lambda)$ au point courant λ_k. On se déplace ensuite le long de la direction opposée au sous-gradient d'un pas

prédéterminé pour calculer le point suivant λ_{k+1}. Ce processus itératif est répété jusqu'à la satisfaction d'un test d'arrêt de l'algorithme. Les étapes de l'algorithme du sous-gradient sont les suivantes:

- **Étape 1 : Initialisation**

  Soit λ_1 un point initial.

  $k = 1$.

- **Étape 2 : Mise à jour des multiplicateurs**

  Déterminer un sous-gradient s_k de la fonction $g(\lambda)$ au point λ_k.

  Poser $\lambda_{k+1} = max(0, \lambda_k - t_k \frac{s_k}{\|s_k\|})$.

- **Étape 3 : Test d'arrêt**

  Si $\|\lambda_k - \lambda_{k-1}\| \leq \epsilon$ ou $|g(\lambda_k) - g(\lambda_{k-1})| \leq \delta$, alors fin, avec λ_k comme solution approchée; sinon poser $k = k + 1$ et aller à l'étape 2.

Pour le test d'arrêt de cet algorithme, on demande à ce que la différence entre les valeurs de la fonction au point courant et le point qui l'a précédé, soit inférieure ou égale à une tolérance δ, soit la condition $|g(\lambda_k) - g(\lambda_{k-1})| \leq \delta$. De même, on peut aussi utiliser la norme de la différence entre le point courant et celui qui l'a précédé, soit $\|\lambda_k - \lambda_{k-1}\| \leq \epsilon$, où ϵ est une tolérance fixée. L'efficacité de l'algorithme du sous-gradient du point de vue vitesse de convergence est liée au choix du pas t_k à chaque itération. Trois expressions sont généralement utilisées pour la détermination du pas. Elles assurent des convergences linéaires sous certaines conditions. Ces trois approches sont les suivantes:

- Méthode du pas constant : $t_k = t_0$;
- Méthode de la série convergente : $t_k = \beta(\alpha)^k$ où $0 < \alpha < 1$;

- Méthode de relaxation : $t_k = \rho \frac{g(\lambda_k) - \overline{g}}{\|s_k\|}$, où $\overline{g}$ est une estimation de la valeur optimale et ρ est une constante telle que $0 < \rho < 2$.

Signalons que les tests pour cette méthode ont été conduits sur des instances du problème de la fermeture maximale sur un graphe avec contraintes de ressource. Ce type de problème correspond aux problèmes de planification minière où on a considéré une seule période comme horizon. Nous avons retenu pour ces tests la méthode de relaxation pour l'ajustement du pas. Les résultats de ces tests comparatifs avec la méthode de faisceaux sont présentés dans le chapitre 5 de cette thèse.

4.4 Méthode des plans sécants

La méthode des plans sécants de Kelley [27] est une méthode largement utilisé pour l'optimisation des fonctions convexes non partout différentiables. Elle consiste à maintenir une approximation de la fonction à optimiser à l'aide de linéarisations tangentielles en plusieurs points, et à chaque itération, elle détermine le minimum de la fonction approximative et génère au point où ce minimum est trouvé un nouveau plan tangent qui vient enrichir l'approximation. Le processus est poursuivi jusqu'à ce que l'optimum recherché soit atteint. Pour le développement de cette méthode, considérons le problème à résoudre: $min\{g(\lambda) \ : \ \lambda \geq 0\}$. Soit λ_i $(i = 1, \ldots, k)$ un ensemble de points où pour chaque λ_i, on connaît la valeur de la fonction $g(\lambda_i)$ et un sous-gradient s_i de $g()$ au point λ_i. La fonction $g()$ peut être approchée par ses linéarisations tangentielles aux points λ_i, $(i = 1, \ldots, k)$. Pour déterminer le minimum de cette approximation, on résout le programme linéaire suivant:

$$(PS)_k \quad \min \theta_k$$

$$g(\lambda_i) + s_i^T(\lambda - \lambda_i) \leq \theta_k, \ \ i = 1, \ldots, k$$

$$\lambda \geq 0.$$

Le nombre de contraintes dans le programme $(PS)_k$ peut devenir très grand au cours des itérations. Habituellement cette difficulté est contournée par la résolution du dual de $(PS)_k$. En associant à chaque contrainte de linéarisation $\theta_k + s_i^T \geq g(\lambda_i) - s_i^T\lambda_i \ \ (i = 1, \ldots, k)$, un multiplicateur $\alpha_i \geq 0$, le dual de $(PS)_k$ s'écrit :

$$(DS)_k \quad \theta_k = \max \sum_{i=1}^{k} \alpha_i (g(\lambda_i) - s_i^T \lambda_i)$$

$$\sum_{i=1}^{k} \alpha_i = 1$$

$$\sum_{i=1}^{k} \alpha_i s_i \geq 0$$

$$\alpha_i \geq 0, i = 1, \ldots, k.$$

Le programme $(DS)_k$ s'écrit en fonction des points extrêmes du domaine de réalisabilité de P comme suit:

$$\theta_k = \max \sum_{i=1}^{k} \alpha_i f(\overline{x}^i)$$

$$\sum_{i=1}^{k} \alpha_i = 1$$

$$\sum_{i=1}^{k} \alpha_i (b - A\overline{x}^i) \geq 0$$

$$\alpha_i \geq 0, i = 1, \ldots, k.$$

où $\overline{x}^i$ est l'optimum du lagrangien en λ_i. Le dual $(DS)_k$ possède $m + 1$ contraintes où m est la dimension du vecteur des variables λ, mais le nombre de colonnes croît au cours des itérations. La méthode des plans sécants utilisant le problème maître $(PS)_k$ est une approche de génération de contraintes et celle utilisant le problème maître $(DS)_k$ est une approche de génération de colonnes ou Décomposition de Dantzig-Wolfe [9]. La méthode des plans sécants de Kelley vue comme une méthode de génération de colonnes consiste à résoudre à chaque itération le dual $(DS)_k$ pour

obtenir un nouveau point où la linéarisation de $g()$ enrichit le programme $(DS)_k$, qui est résolu de nouveau itérativement jusqu'à la satisfaction d'un critère d'arrêt. Les étapes de l'algorithme décrivant ce processus sont les suivantes:

- **Étape 0 : Initialisation**

  Soit λ_1 un point initial.

  Poser $k = 1$.

- **Étape 1 : Résolution du sous-probième**

  Déterminer $\overline{x}^k$ tel que

$$\overline{x}^k \in argmax_{x \in D}(f(x) + \lambda_k^T(b - Ax))$$

  Poser $g(\lambda_k) = f(\overline{x}^k) + \lambda_k^T(b - A\overline{x}^k)$ et $s_k = b - A\overline{x}^k$

- **Étape 2 : Résolution du problème maître**

  Résoudre le dual $(DS)_k$:

$$\theta_k = \max \sum_{i=1}^{k} \alpha_i(g(\lambda_i) - s_i^T\lambda_i)$$
$$\sum_{i=1}^{k} \alpha_i = 1$$
$$\sum_{i=1}^{k} \alpha_i s_i \geq 0$$
$$\alpha_i \geq 0,\ i = 1 \ldots, k.$$

  Déterminer le nouveau point λ_{k+1} qui est le vecteur des variables duales associées aux contraintes $\sum \alpha_i s_i \geq 0$ dans la solution optimale de $(DS)_k$.

- **Étape 3 : Test d'arrêt**

  Si $g(\lambda_k) - \theta_k \leq \epsilon$, alors fin, avec λ_{k+1} comme solution approchée,

  sinon poser $k = k + 1$ et aller à l'étape 1.

Notons qu'à l'étape 1 de cet algorithme, on résout le lagrangien en λ_k, qui est un problème de flot maximum sur un réseau. A l'étape 2, le problème maître $(DS)_k$ peut être résolu par un algorithme de programmation linéaire.

4.5 Méthode de faisceaux

La méthode de faisceaux (*Bundle method*) a été initialement développée pour remédier aux insuffisances des méthodes de ϵ-descente. Elle utilise, pour déterminer une direction de descente, un problème quadratique et une recherche linéaire plus adéquate pour se déplacer le long de la direction de descente. Nous n'avons pas l'intention d'élaborer cet aspect et nous référons le lecteur intéressé aux travaux de Lemaréchal [18, 31, 32].

Dans les paragraphes suivants, nous présentons une méthode des plans sécants stabilisée qui remédie aux insuffisances précitées de la méthode des plans sécants classique. Puis, nous déduisons la méthode de faisceaux comme une méthode des plans sécants stabilisée dans laquelle une recherche linéaire est insérée. Considérons donc à l'itération k le problème des plans sécants défini à la section précédente, dans lequel seulement $l \leq k$ contraintes de linéarisation sont conservées (on aura pris soin de les renumérotées de 1 à l), et auquel on ajoute une pénalité quadratique. Ce nouveau problème que l'on appellera $(PB)_k$, s'écrit comme suit:

$$(PB)_k \quad \min \theta_k + \frac{1}{2}\mu_k \|\lambda - \lambda_k\|^2$$

$$g(\lambda_i) + s_i(\lambda - \lambda_i) \leq \theta_k \quad i = 1, \ldots, l$$

$$\lambda \geq 0,$$

avec μ_k un paramètre réel et λ_k représentant le point courant et qu'on appelle "centre". L'idée de base de cette méthode est de résoudre à chaque itération le problème précédent pour déterminer une direction de descente puis de se déplacer le long de cette direction à partir du centre pour trouver un nouveau centre meilleur. Le problème de direction tel qu'il est formulé peut présenter un nombre élevé de contraintes. Ainsi, la résolution de son dual est plus appropriée. Pour écrire le dual du problème $(PB)_k$, associons aux contraintes de linéarisation les multiplicateurs $\alpha_i \geq 0$ et aux contraintes de non négativité les variables $\beta_i \geq 0$. Le lagrangien de $(PB)_k$ s'écrit:

$$-\theta_k - \frac{1}{2}\mu_k\|\lambda - \lambda_k\|^2 + \sum_{i=1}^{l} \alpha_i(\theta_k - g(\lambda_i) - s_i(\lambda - \lambda_i)) + \beta^T\lambda.$$

L'optimum (maximum) du lagrangien par rapport à θ_k et à λ est atteint si les conditions suivantes, dérivées des conditions d'optimalité du premier ordre, sont satisfaites:

(a) $\quad \sum_{i=1}^{l} \alpha_i = 1$

(b) $\quad \lambda - \lambda_k + \frac{1}{\mu_k}(\sum_{i=1}^{l} \alpha_i s_i - \beta^T) = 0.$

En se servant de ces deux relations, le dual $(DB)_k$ de $(PB)_k$ s'écrit:

$$\min \frac{1}{2\mu_k}\|\sum_{i=1}^{l} \alpha_i s_i - \beta^T\|^2 + \sum_{i=1}^{l} \alpha_i(e_i - g(\lambda_k)) + \beta^T\lambda_k$$

$$s.c. \quad \sum_{i=1}^{l} \alpha_i = 1$$

$$\alpha_i \geq 0 \quad \beta_i \geq 0, \quad i = 1, \ldots, l.$$

avec $e_i^k = g(\lambda_k) - (g(\lambda_i) + s_i(\lambda_k - \lambda_i))$ qui représente l'écart de linéarisation de $g(\lambda)$ entre λ_k et λ_i. On utilisera par la suite la notation e_i pour désigner e_i^k si aucune confusion n'est possible. En multipliant l'objectif de $(DB)_k$ par le paramètre μ_k et en lui ajoutant la constante $g(\lambda_k)$, le terme linéaire de ce nouvel objectif

$\mu_k(\sum_{i=1}^{l} \alpha_i e_i + \beta^T \lambda_k)$ s'interprète comme la dualisation de la contrainte

$$\sum_{i=1}^{l} \alpha_i e_i + \beta^T \lambda_k \leq \epsilon$$

où le paramètre ϵ est intrinsèquement lié au multiplicateur μ_k. En tenant compte de cette transformation de $(DB)_k$, on obtient le problème, communément appelé *problème de faisceaux* et qui s'écrit:

$$\min \frac{1}{2} \| \sum_{i=1}^{l} \alpha_i s_i - \beta^T \|^2$$

$$s.c. \quad \sum_{i=1}^{l} \alpha_i = 1$$

$$\sum_{i=1}^{l} \alpha_i e_i + \beta^T \lambda_k \leq \epsilon$$

$$\alpha_i \geq 0 \quad \beta_i \geq 0.$$

Le problème de faisceau est construit en se basant sur un faisceau d'informations $F = \{(s_i, e_i, \lambda_i), \quad i = 1, \ldots, l\}$ généré au cours des itérations. Le compteur l désigne la longueur du faisceau et représente le nombre de linéarisations utilisées dans le problème de faisceau. La résolution de ce problème donne une direction de descente. Soient $\overline{\alpha}$ et $\overline{\beta}$ les vecteurs solutions du problème de faisceau. La direction de descente est donnée par: $d = \overline{\beta} - \sum_{i=1}^{l} \overline{\alpha}_i s_i$. Dans une méthode de plans sécants stabilisée, le prochain point est déterminé par la relation (b) déduite précédemment comme suit : $\lambda_k + \frac{1}{\mu_k} d$, où μ_k est le multiplicateur dual de la contrainte $\sum_{i=1}^{l} \alpha_i e_i + \beta^T \lambda_k \leq \epsilon$ dans la solution optimale de $(DB)_k$. Comme mentionné en début de section, la méthode de faisceaux qui est une méthode plus générale, utilise une recherche linéaire à cette étape pour déterminer le nouveau point sous la forme $\max(0, \lambda_k + t_k d)$. Le pas à déterminer t_k doit, soit assurer une décroissance de la fonction $g()$, soit permettre d'enrichir le faisceau. La méthode de faisceaux consiste en gros à déterminer une direction de descente en résolvant le problème de faisceau et à déterminer le

prochain point par une recherche linéaire le long de la direction de descente. A chaque itération, le point déterminé conduit à l'une des étapes suivantes:
- étape de descente qui survient quand le point courant implique une décroissance suffisante de la fonction. Dans ce cas le centre λ_k est modifié.
- étape nulle qui survient quand la décroissance n'est pas suffisante mais le sous-gradient correspondant au point courant permet d'enrichir le problème de faisceau .

Les étapes de l'algorithme décrivant cette méthode [18] sont:

- **Étape 0 (Initialisation)**

  Soit λ_1 un point de départ et s_1 le sous-gradient de $g(\lambda)$ au point λ_1.

  On définit les paramètres suivants:

  $m = 0.1$, $m' = 0.2$, $\overline{m} = 0.1$, $\delta = 0.1$, $\underline{\epsilon} = 0.1$ et $\overline{\epsilon} = g(\lambda_1) - \overline{g}$ ou $\overline{g}$ est une estimation de la solution optimale.

  Poser $l = 1$, le nombre d'éléments dans le faisceau et $k = 1$, l'index des itérations.

  Initialiser le faisceau $F = \{(s_1, e_1 = 0)\}$

- **étape 1 (Direction de descente)**

  Poser $\epsilon = max(\underline{\epsilon}, min(\epsilon, \overline{\epsilon}))$

  Résoudre le problème de faisceaux, soit $\overline{\alpha}$ et $\overline{\beta}$ les vecteurs solutions.

  Poser $\hat{s} = \sum_{i=1}^{l} \overline{\alpha}_i s_i - \overline{\beta}^T$ et $\hat{e} = \sum_{i=1}^{l} \overline{\alpha}_i e_i + \overline{\beta}^T \lambda_k$.

- **Étape 2 (Test d'arrêt)**

  Si $\|\hat{s}\| > \delta$, aller à l'étape 3;

  sinon si $\hat{e} \leq \underline{\epsilon}$, fin;

  sinon poser $\overline{\epsilon} = max(0.1\hat{e}, \underline{\epsilon})$ et aller à l'étape 1.

- **Étape 3 (Recherche linéaire)**

  Déterminer le paramètre réel t afin que le nouveau point $\lambda_+ = \max(0, \lambda_k - t\hat{s})$,

auquel est associé le sous-gradient s_+ et l'écart de linéarisation e_+, réalise l'une des deux étapes suivantes:

a) étape de descente:

$$g(\lambda_+) \leq g(\lambda_k) - mt\|\hat{s}\|^2 \text{ et } e_+ > \overline{m}\epsilon$$

b) étape nulle:

$$e_+ \leq \overline{m}\epsilon \text{ et } s_+^T\hat{s} \leq m'\|\hat{s}\|^2$$

- **Étape 4 (Modification du centre)**

Poser $s_{l+1} = s_+$

Si le pas t implique une étape nulle, poser:

$$\lambda_{k+1} = \lambda_k$$

$$e_{l+1} = g(\lambda_k) - g(\lambda_+) - ts_+^T\hat{s}.$$

Si le pas t implique une étape de descente, poser:

$$\lambda_{k+1} = \lambda_+$$

$$e_{l+1} = 0$$

$$e_i = e_i + g(\lambda_+) - (g(\lambda_k) - ts_i^T\hat{s}) \quad \forall\, i = 1, \ldots, l$$

Ajouter au faisceau le nouvel élément (s_{l+1}, e_{l+1})

Poser $k = k + 1$, $l = l + 1$, choisir un nouveau ϵ et aller à l'étape 1.

Le bon fonctionnement de cet algorithme dépend grandement du choix du paramètre ϵ posé à chaque itération et de l'efficacité de la recherche linéaire. Pour le choix du ϵ à l'étape 4, Lemaréchal [18] propose plusieurs approches, dont les suivantes:

(a) $\epsilon = g(\lambda_k) - \overline{g}$

(b) $\epsilon = t\|\hat{s}\| + t\mu\epsilon$

(c) $\epsilon = \rho(g(\lambda_k) - \overline{g})$

(d) $\epsilon = g(\lambda_{k-1}) - g(\lambda_k)$

(e) $\epsilon = \epsilon_k$

Les alternatives (a) et (c) nécessitent la connaissance d'une approximation de la solution optimale recherchée. Dans notre implémentation, nous avons utilisé l'alternative (d). Ainsi, nous modifions à chaque itération le paramètre ϵ comme suit:

$$\epsilon_{k+1} = max(\underline{\epsilon}, min(0.1(g(\lambda_k) - g(\lambda_{k-1})), \bar{\epsilon})).$$

Pour la recherche linéaire, l'objectif est de déterminer un nouveau point $\lambda_+ = \lambda_k - t\hat{s}$ qui implique, soit une étape de descente ou soit une étape nulle. Nous avons utilisé la recherche proposée par Lemaréchal dont l'algorithme est le suivant:

- **Étape 0 (initialisation)**

  Poser $t_L = 0$ et $t_R = 0$

  Poser $t = t_0$ un pas initial.

- **Étape 1 (évaluation)**

  Poser $\lambda_+ = \lambda_k - t\hat{s}$.

  Évaluer $g(\lambda_+)$, le sous-gradient associé s_+ et l'écart de linéarisation e_+

- **Étape 2 (test d'une étape nulle)**

  Si $e_+ \leq \overline{m}\epsilon$ et $s_+^T\hat{s} \leq -m'\|\hat{s}\|^2$ alors fin, avec t comme pas impliquant une étape nulle.

- **Étape 3 (test pour t grand)**

  Si $g(\lambda_+) > g(\lambda_k) - mt\|\hat{s}\|^2$, poser $t_R = t$ et aller à l'étape 6.

- **Étape 4 (étape de descente)**

  Si $e_+ > m\epsilon$ stop, avec t comme pas impliquant une étape de descente;

  sinon poser $t_L = t$ et aller à l'étape 5.

- **Étape 5 (extrapolation)**

  Si $t_R = 0$, trouver un nouveau t par extrapolation à partir de t_L et aller à l'étape 1.

- **Étape 6 (interpolation)**

  Trouver par interpolation entre t_L et t_R un nouveau t et aller à l'étape 1.

Dans cette recherche linéaire, pour chaque pas t qu'on essaie, on évalue la fonction $g()$, ce qui dans notre cas consiste à résoudre un problème de flot maximum sur un graphe de très grande taille. Une telle résolution est très coûteuse en temps de calcul. Il est donc essentiel que la recherche linéaire termine avec un pas convenable après un nombre réduit d'évaluations de la fonction $g()$. Le choix du pas initial et la manière dont l'interpolation est réalisée affecte considérablement le comportement de la recherche.

Choix du pas initial : Pour déterminer un pas initial, nous supposons que la fonction $\theta(t) = g(\lambda_k - t\hat{s})$ s'approche par une fonction quadratique en t de la forme $\frac{1}{2}at^2 - s_k^T\hat{s}t + g(\lambda_k)$. Nous choisirons comme pas initial la valeur de t où la fonction quadratique atteint son minimum. Soit t_0 cette valeur, qui est donnée par:

$$t_0 = 2(\theta(0) - \theta(t_0))/s_k^T\hat{s}.$$

En considérant que la décroissance de la fonction $\theta()$ entre le point $t = 0$ et le point $t = t_0$ est égale à celle de la fonction $g()$ entre λ_{k-1} et λ_k, l'expression donnant un pas initial est:

$$t_0 = 2(g(\lambda_{k-1}) - g(\lambda_k))/s_k^T\hat{s}.$$

Interpolation : Soient t_L et t_R respectivement la borne inférieure et la borne supérieure

de l'intervalle d'interpolation. Pour assurer la réduction de l'intervalle, le nouveau pas à essayer est déterminé par l'expression suivante:

$$t = min\{\rho t_L + (1-\rho)t_R, max(t^*, (1-\rho)t_L + \rho t_R)\}$$

où $\rho \in]0, \frac{1}{2}[$ et t^* est le point où une approximation quadratique de $\theta(t)$ atteint son minimum. En considérant s_L et s_R les sous-gradients de la fonction $g()$ aux points $\lambda_k - t_L\hat{s}$ et $\lambda_k - t_R\hat{s}$, on détermine t^* par la formule suivante:

$$t^* = \frac{\theta(t_R) - \theta(t_L)}{(s_R - s_L)^T\hat{s}} + \frac{1}{2}(t_L + t_R).$$

4.6 Conclusion

Dans ce chapitre, nous avons présenté les principes de la relaxation lagrangienne et les méthodes d'ajustement des multiplicateurs de Lagrange couramment utilisées. Dans les applications minières, les problèmes lagrangiens consistent à résoudre des problèmes de fermeture maximale sur des graphes de très grande taille. La résolution de ces problèmes est généralement très coûteuse en temps de calcul. Ainsi une bonne méthode d'ajustement des multiplicateurs est celle qui résolvera le moins souvent possible le problème lagrangien pour trouver les valeurs optimales des multiplicateurs. Nous présenterons dans les chapitres suivants les différentes fonctions duales qu'on aura à optimiser. Dans le chapitre 5, nous comparerons la méthode du sous-gradient et la méthode de faisceaux pour l'ajustement des multiplicateurs pour un problème de planification sur une période. À la lumière des résultats de cette comparaison, nous utiliserons la meilleure méthode pour le traitement du problème multi-périodes, le sujet du chapitre 6.

CHAPITRE 5

Fermeture maximale sur un graphe en présence de contraintes de ressource

5.1 Présentation

Dans le cadre de la revue de littérature, la relaxation lagrangienne des contraintes de ressource qui apparaîssent dans la formulation du problème de planification de l'exploitation d'une mine à ciel ouvert, est considérée comme une approche de résolution prometteuse. Le présent article, "*Maximal Closure on a Graph with Resource Constraints*", explore cette approche. Dans cet article, nous avons considéré un problème de planification réduit à une seule période. Ce problème se présente comme le problème de la recherche d'une fermeture maximale satisfaisant un ensemble de contraintes de ressource sur un graphe. L'approche de résolution consiste à traiter les contraintes de ressource par relaxation lagrangienne afin de se ramener à un problème de fermeture maximale classique qu'on sait traiter par un algorithme de flot maximum (voir chapitre 2). Pour l'ajustement des multiplicateurs de Lagrange, nous explorons deux méthodes: la méthode du sous-gradient et la méthode de faisceaux, en vue de sélectionner la plus appropriée à ce problème. Dans cette approche, le problème du lagrangien qui est un problème de fermeture maximale est résolu sur des graphes de très grande taille. Ces graphes présentent des dizaines de milliers de nœuds et des centaines de milliers d'arcs. Ainsi la détermination d'une fermeture maximale sur ce type de graphe est généralement coûteuse en temps de calcul.

L'approche de relaxation lagrangienne ne trouve pas toujours des solutions primales réalisables. Cet article présente une heuristique simple du type glouton permettant de convertir toute solution ne satisfaisant pas les contraintes dualisées en une solution réalisable. Cette heuristique considère une fermeture sur un graphe qui viole les contraintes de ressource et cherche à la rendre réalisable en enlevant des nœuds à sa frontière de façon à dégrader le moins possible la valeur de la fermeture et à diminuer le plus possible les surplus des contraintes.

5.2 Maximal closure on a graph with resource constraints

Beyime Tachefine and François Soumis

GERAD and École Polytechnique de Montréal
5255 avenue Decelles, Montréal, Canada, H3T 1V6

Scope and Purpose

Some scheduling problems can be formulated as a maximal closure problem on a graph. In such graphs, the nodes represent the tasks to perform and the arcs represent the relation of precedence between the tasks. The problem here is to find a set of nodes (set of tasks) with a maximal value and which respect the precedence relations. Such problem is usually solved using a maximal flow algorithm. When constraints related to the resources used to realize the tasks are introduced, the problem becomes more complex to solve optimally. This paper proposes an approach to solve it based on the Lagrangian relaxation of the resource constraints. This approach gives a solution very close to the optimum by solving maximal closure problems in average of ten times.

Abstract

This paper formulates the problem of maximal closure on a graph with resource constraints as an integer programming problem with binary variables, and proposes a solution approach based on Lagrangian relaxation. The subgradient and bundle methods for solving the dual problem are compared. The subproblem resulting from relaxing the resource constraints is the classical maximal closure problem, which is solved using a maximal flow algorithm. This article proposes a heuristic to transform closures that do not satisfy resource constraints into feasible closures. The heuristic finds feasible integer solutions that are close to the upper bound provided by the Lagrangian relaxation.

5.2.1 Introduction

Let $G(N, A)$ be a directed graph in which N is the set of nodes and A the set of arcs. With each node $i \in N$ is associated a value c_i which can take any value. If $(j, k) \in A$, the node k is called the *successor* to node j. A *closure* on G is a set of nodes $L \subset N$ such that any successor to a node in L is also in L. The problem of maximal closure on G consists in finding that closure for which the sum of node values is the greatest. This problem has many important applications, including the design of open pit mines [16], selection of freight handling terminals [20], record segmentation in large shared databases [5], portfolio selection under contingency constraints [25], sequencing of manufacturing and assembly tasks in a flexible shop [7] and storage of mobile repair kits [17]. A number of authors discuss other applications of the problem of maximal closure on a graph [9, 19, 22]. As Picard [18] has demonstrated, the problem of maximal closure on a graph can be solved using a maximal flow algorithm on a modified graph. Other approaches proposed in the literature (for example, Faaland et *al.* [8]) have no major advantages over maximal flow algorithms. This article uses Picard's approach, which is simple and well structured.

Let K be a set of resources consumed at the nodes of the graph G, and a_{ki} the consumption of resource k at node i. Then the maximal closure with resource constraints is defined as the maximal closure such that consumption of each resource k is less than or equal to the resource availability b_k. One important application of this problem is the design of open pit mines. In addition, the problems cited above are excellent applications of this problem when resource constraints are present. A number of authors (such as Dagdeleen & Johnson [2] and Zhao & Kim [26]) have formulated mine planning problems as maximal closure on a graph with resource constraints, and have solved it using Lagrangian relaxation, with the subgradient

method to adjust multipliers. Other heuristic approaches to multiplier adjustment have been proposed by Davis & Williams [4] and Eleveli *et al.* [6].

This article presents an approach to solving the problem of maximal closure on a graph with resource constraints by using Lagrangian relaxation of the resource constraints. The resulting subproblem is handled as a classical maximal closure problem. The objective of this paper is to solve very large scale problems, i.e., with graphs comprising between 10,000 and 100,000 nodes, and with three to ten resource constraints. Since computation time is important in this type of problem, a good solution approach should minimize the number of subproblem evaluations and ensure that a good solution satisfying the resource constraints is found. In addition, the classical subgradient and bundle methods for adjusting multipliers in the dual problem are compared. As these approachs provide excellent upper bounds but poor feasible solutions, a greedy heuristic is presented for finding good feasible solutions.

Section 2 presents a formulation of this problem, emphasizing its application to the mining industry. Section 3 presents the solution method based on Lagrangian relaxation, including solution of the dual and relaxed problems. The heuristic is presented in Section 4. Section 5 presents numerical results. In this section, we compare the bundle and subgradient methods for solving the dual problem and show the contribution of the proposed heuristic for reducing the gap between the solution and the upper bound.

5.2.2 Problem formulation

To formulate the problem of maximal closure on a graph $G(N, A)$ with resource constraints, the following decision variables are defined for $i \in N$:

$$x_i = \begin{cases} 1 & \text{if node } i \text{ is in the closure,} \\ 0 & \text{otherwise.} \end{cases}$$

Given that:

Γ_i is the set of successors of node i,

c_i is the value associated with node i,

K is the set of resources consumed at the nodes of graph G,

a_{ki} is the consumption of resource k at node i,

and b_k is the resource k availability,

the problem is then formulated as follows:

$$\max \sum_{i \in N} c_i x_i \tag{5.1}$$

$$\sum_{i \in N} a_{ki} x_i \leq b_k, \quad k \in K \tag{5.2}$$

$$x_i - x_j \leq 0, \quad j \in \Gamma_i, \quad i \in N \tag{5.3}$$

$$x_i \in \{0,1\}, i \in N. \tag{5.4}$$

The objective (5.1) is to maximize the total value of the closure. Constraints (5.2) are resource constraints that the desired closure must satisfy. Constraints (5.3) identify the precedence arcs on the graph and require that any set of selected nodes must be a closure on the graph. Without constraints (5.2), this problem reduces to a conventional maximal closure problem.

This problem can be applied to the planning of open pit mines. A mineral ore deposit is generally represented using a model of three-dimensional blocks. Each block in the model is characterized by a set of data describing its position in the

deposit, ore content and distance from exploitation facilities. One problem faced by mine operators is how to determine the contours of an open pit mine subject to such resource constraints as transportation and processing capacity. The problem is to determine the blocks that should be part of the open pit mine, in such a way that operating income is maximized while (especially physical) resource constraints are satisfied. For example, a block may not be extracted unless all blocks that physically constrain it have already been extracted. This problem can be formulated as a maximal closure problem with resource constraints on a graph representing the mine. In this graph, each node corresponds to a block and each arc (i, j) specifies that block j must be extracted before block i.

5.2.3 Solution approach

As already mentioned above, without the resource constraints, the formulation reduces to the classical maximum closure problem. To exploit this structure, this article proposes a Lagrangean relaxation of the resource constraints, in which the subproblem amounts to a closure problem. Let $\lambda^k \geq 0, \quad k \in K$, denote the multipliers associated with constraints (5.2). Multiplying each constraint by the corresponding multiplier λ^k and moving it into the objective function (5.1) in the form of a penalty, the problem becomes:

$$\Phi(\lambda) = \max \sum_{i \in N} \bar{c}_i(\lambda) x_i + \sum_{k \in K} \lambda^k b_k \tag{5.5}$$

$$s.t. \quad x_i - x_j \leq 0, \quad j \in \Gamma_i \quad i \in N \tag{5.6}$$

$$x_i \in \{0,1\}, i \in N \tag{5.7}$$

where $\bar{c}_i(\lambda) = c_i - \sum_{k \in K} \lambda^k a_{ki}$

For fixed $\lambda = (\lambda^k)$ the relaxed formulation is that of a maximal closure problem on a graph G in which the updated value at each node i is given by $\bar{c}_i(\lambda)$. For a given $\lambda \geq 0$ the optimal value $\Phi(\lambda)$ of problem (5.5) – (5.7) is an upper bound on the value of any feasible solution of problem (5.1) – (5.4). Therefore, if Z^* is the optimal solution to the initial problem, then

$$Z^* \leq \Phi(\lambda).$$

The best bound, therefore, corresponds to the multipliers λ^* such that

$$\lambda^* \in \text{argmin}\{\Phi(\lambda), \quad \lambda \geq 0\}.$$

The function $\Phi(\lambda)$ is a piecewise linear convex function; furthermore, at each point λ_p there is a subgradient $s_p = (s_p^k)$ with components $s_p^k = b_k - \sum_{i \in N} a_{ki} x_i(\lambda_p)$, where $x_i(\lambda_p)$ is the i^{th} component of the solution vector for problem (5.5) – (5.7).

5.2.3.1 Subgradient method

The subgradient method sets initial multipliers λ_0^k $(k \in K)$, and at iteration p, they are adjusted as follows:

$$\lambda_{p+1}^k = \max(0, \lambda_p^k - t_p s_p^k) \quad \text{where}$$

$$t_p = \rho(\Phi(\lambda_p) - \underline{Z})/\|s_p\|^2 \quad \text{with} \quad 0 < \rho < 2,$$

and where $\underline{Z}$ is a lower estimate of the optimal solution.

Since the optimal solution is not known in advance, it is replaced in the algorithm by $\underline{Z}$, the best known value of the objective function corresponding to a feasible solution. The stopping criterion in calculating the results presented in section 5 is whichever of the following occurs first: $\|s_p\| \leq \delta$ or $|\Phi(\lambda_p) - \Phi(\lambda_{p-1})| \leq \epsilon$ with $\delta = 0.001$ and $\epsilon = 0.01$.

5.2.3.2 Bundle method

One standard method for solving the dual problem $\min\{\Phi(\lambda),\ \lambda \geq 0\}$ is the Kelley's cutting plane method [12] in which constraints are generated and accumulated one after another. In this approach, if λ_q $(q = 0, \ldots, p)$ represent multiplier vectors generated in the p previous iterations, then multiplier vector λ_{p+1} at the next iteration is determined by the solution of the following linear program or its dual:

$$\begin{aligned} & \min \phi \\ s.t.\ \ \Phi(\lambda_q) \ &+\ s_q^T(\lambda - \lambda_q) \leq \phi,\ \ q = 0, 1, \ldots, p \qquad (5.8) \\ \lambda \ &\geq\ 0. \end{aligned}$$

Solution to the dual problem of (5.8) corresponds to the column generation approach (Dantzig-Wolfe decomposition [3]). This cutting plane method suffers from two major weaknesses: slow convergence and instability (in the sense that one iteration close to the optimum may be followed by another fairly far away). The bundle method remedies the weaknesses of this cutting plane method by adding a quadratic penalty to the preceding problem, as follows:

$$\min \phi \ + \ \frac{1}{2t}\|\lambda - \overline{\lambda}_p\|^2 \qquad (5.9)$$

$$s.t.\ \ \Phi(\lambda_q) \ + \ s_q^T(\lambda - \lambda_q) \leq \phi,\ \ q = 0, 1, \ldots, p \qquad (5.10)$$

$$\lambda \ \geq \ 0, \qquad (5.11)$$

where $\overline{\lambda}_p$ is a vector of multipliers called the *center* and defined below, and t is a real-valued parameter. By associating the dual variables α_q with constraints (5.10) and the variables μ with the non-negativity constraint (5.11), the dual of the quadratic problem (5.9) – (5.11), called the *bundle* problem, can be written as:

$$\min \frac{1}{2}\|\sum_{q=0}^{p} \alpha_q s_q - \mu^T\|^2 \ + \ \frac{1}{t}(\sum_{q=0}^{p} \alpha_q e_q + \mu^T \overline{\lambda}_p)$$

$$
\begin{aligned}
s.t. \sum_{q=0}^{p} \alpha_q &= 1 \\
\alpha_q &\geq 0, \quad q = 0, 1, \ldots, p \\
\mu &\geq 0,
\end{aligned}
$$

where $e_q = \Phi(\overline{\lambda}_p) - [\Phi(\lambda_q) + s_q^T(\overline{\lambda}_p - \lambda_q)]$.

The bundle method considers a bundle of information, $B = \{(s_q, e_q) \quad q = 0, 1, \ldots, p\}$ regarding the dual function $\Phi(\lambda)$, accumulated over the course of the iterations. At any iteration, the solution to the preceding bundle problem provides a descent direction $\overline{d} = \overline{\mu} - \sum_{q=0}^{p} \overline{\alpha}_q s_q$ where $\overline{\mu}$ and $\overline{\alpha}$ are vector solutions to the bundle problem. Starting at the center $\overline{\lambda}_p$, a line search is conducted along the search direction $\overline{d}$, to determine a search step t^* such that $\lambda_{p+1} = \overline{\lambda}_p + t^*\overline{d}$ guarantees sufficient decrease in the dual function or generates a new subgradient s_{p+1} that improves the bundle problem. Should there be a sufficient decrease, the center is shifted: $\overline{\lambda}_{p+1} = \lambda_{p+1}$. In the second situation, there is no shift; hence $\overline{\lambda}_{p+1} = \overline{\lambda}_p$. In either situation the bundle problem is improved by enlarging the bundle by one element $B = B \cup \{(s_{p+1}, e_{p+1})\}$. This process repeats until satisfaction of the stopping criterion, which verifies $\|\overline{d}\| \leq \delta$ and $\sum_{i \in I} \overline{\alpha}_i e_i + \overline{\mu}^T \overline{\lambda}_p \leq \epsilon$, where δ and ϵ are tolerances set by the user. For further information regarding updating of linearization gaps e_i, implementation of the line search and management of the size of the bundle, the reader is referred to a number of references by Lemarechal [10, 14, 15].

5.2.3.3 The classical closure problem

A number of approaches exist for solving the maximal closure problem. The approach of Picard [18] is presented here. This method consists in treating the problem as one of maximal flow on a reduced graph derived from a given initial graph. Picard proposes augmenting the graph G by a source s and sink t, placing an arc

(s,i) with capacity $c(s,i) = \bar{c}_i(\lambda)$ between the source s and any node i if $\bar{c}_i(\lambda) > 0$, and placing an arc (j,t) with capacity $c(j,t) = -\bar{c}_j(\lambda)$ between any node j and the sink t if $\bar{c}_j(\lambda) \leq 0$. We are using $\bar{c}_i(\lambda)$ to denote the node values computed every iteration of the dual optimization problem, $\min\{\Phi(\lambda), \quad \lambda \geq 0\}$, as follows $\bar{c}_i(\lambda) = c_i - \sum_{k \in K} \lambda^k a_{ki}$. An infinite capacity $c(i,j) = \infty$ is placed on all other existing arcs (i,j) in the graph G. The maximal closure sought for is determined by all nodes in the subset of nodes containing the source node defined with respect to a minimum cut on the modified graph. A minimal cut is determined by a maximal flow algorithm. Figure 5.1 illustrates this procedure.

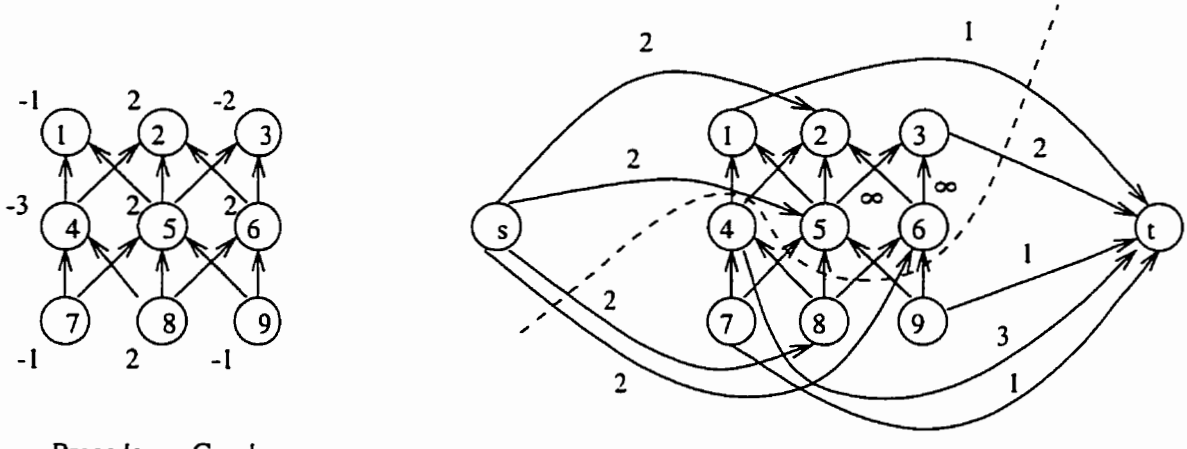


Figure 5.1 Precedence graph and associated Picard's graph

A number of algorithms to solve the maximal flow problem are available in the literature [1, 11]. An initial comparative study of the application of various maximal flow problems to the problem of maximal closure on graphs arising from mining applications has been published in the form of a research report [24]. Preliminary results indicate that a pre-flow algorithm with Highest Label order, with theoretical complexity of $o(n^2 m^{\frac{1}{2}})$, appears to provide good results for the problems at hand, where n is the number of nodes and m is the number of arcs. Numerical experiments

show a computer time growing linearly for problems with 10,000 to 100,000 nodes. The largest problems with 100,000 nodes and 1,000,000 arcs are solved in less than 20 seconds CPU on HP9000/735 machine.

The following definitions are useful for describing this algorithm:

A *pre-flow* f on G is a real-valued function on the arcs of the graph G such that:

$$0 \leq f(i,j) \leq c(i,j) \text{ for all } (i,j) \in A$$

$$e(i) = \sum_{j \in \Gamma_i^{-1}} f(j,i) - \sum_{j \in \Gamma_i} f(i,j) \geq 0 \text{ for all } i \in N - \{s\}$$

A node is said to have a *surplus* $e(i)$ if $i \in N - \{s\}$ and $e(s) = \infty$. A node $i \in N - \{s,t\}$ is said to be *active* if $e(i) > 0$.

A *valid labelling* for a pre-flow f is a function $d()$ of the node set onto the set of natural numbers such that $d(s) = n$, $d(t) = 0$ and $d(i) \leq d(j) + 1$ for any residual arc (i,j). (An arc (i,j) is said to be *residual* if $f(i,j) < c(i,j)$).

The pre-flow algorithm maintains an initial pre-flow f initially set equal to arc capacities on arcs emanating from the source and zero on all other arcs. The algorithm improves the pre-flow by "pushing" node surpluses from the source toward the sink across residual arcs, estimated using valid labelling on the residual shortest path.

We now describe the algorithm. For each node $i \in N$, let $I(i)$ indicate the set of arcs incident to i made up of the set of pairs $\{i,j\}$ such that $(i,j) \in A$ or $(j,i) \in A$. In addition, a *current* arc, initially taken as the first element of $I(i)$, is associated with each node i. The steps of the algorithm are as follows:

While there is an active node **do**:

1. **Select** an active node i, and its associated current arc (i, j).

2. **Push-Relabel**(i).

The **Push-Relabel**(i) step consists in applying one of the following three cases:

- **Push**: if $d(i) > d(j)$ and $f(i,j) < c(i,j)$ then push $\delta = \min\{e(i), c(i,j) - f(i,j)\}$ flow units from i to j by increasing $f(i,j)$ and $e(j)$ by δ and by decreasing $f(j,i)$ and $e(i)$ by δ.

- **Next current arc**: if $d(i) \leq d(j)$ or $f(i,j) = c(i,j)$ and (i,j) is not the last arc in $I(i)$, then replace (i,j) as current arc by the next arc in $I(i)$.

- **Relabel**: If $d(i) \leq d(j)$ or $f(i,j) = c(i,j)$ and (i,j) is the last element of $I(i)$, then replace $d(i)$ by $1 + \min\{d(k)|(i,k) \in I(i) \text{and} f(i,k) < c(i,k)\}$ and take the first element of $I(i)$ as current arc for i.

The performance of this algorithm depends on the order in which the active nodes are examined. This research has adopted the Highest Label order. The approach consists in always selecting the active node which has the highest distance label and continuing to process this node as long as it is active and unlabeled. Using an array of lists, TAB, the element k of which is the list of active nodes with distance label is k, and setting k^* as the highest index value corresponding to a non empty list, the highest label node to examine is choosen arbitrarily from $TAB[k^*]$ list. When the algorithm terminates, source-side nodes in a minimal cut are determined by a breadth-first search on the graph of residual arcs, starting at the source.

The pre-flow algorithm as described could be specialized to determine the minimal cut, without having to determine optimal flows on the graph. The modification consists in considering a node i as active if $e(i) > 0$ and $d(i) < n$. Subproblem solution for the algorithm described in this article uses this specialized minimal cut

algorithm.

5.2.4 Exploration heuristic

While Lagrangian relaxation produces a good upper bound, finding a feasible solution is still a matter of luck. Furthermore, the bundle method to solve the dual problem, while it arrives at the optimal solution after very few iterations, generates solutions which are not feasible for the original problem. The heuristic proposed here remedies this difficulty by constructing a feasible solution to the original problem from an arbitrary unfeasible solution. When required, we apply this heuristic every iteration of the dual optimization process in order to cause the current solution to become feasible.

5.2.4.1 Structure of the heuristic

This heuristic uses a greedy algorithm. It begins with a graph closure that does not satisfy the resource constraints and eliminates nodes until it obtains a new closure that does satisfy these constraints. Nodes are selected for elimination in such a way that the value of the closure decreases as little as possible and the surpluses on constraints not satisfied by the current closure are reduced as much as possible. This selection is guided by criteria described in the next subsection. In addition, a node is not a candidate for elimination unless none of its predecessors in the graph belongs to the current closure. Lets define a *frontier* as the set of nodes belonging to the closure that are possible candidates for elimination. The heuristic consists in sorting the elements of the frontier set according to a given selection criterion and, at each stage, eliminating the best candidate, re-calculating surpluses on resource constraints and updating the set of frontier elements.

The following definitions are required to develop this heuristic:

$X = \{i \ : \ x_i = 1\}$ is the set of nodes making up the original unfeasible closure.

$F = \{i \in X \quad : \quad \Gamma_i^{-1} \cap X = \emptyset\}$ is the set of frontier nodes for the closure defined by X

K is the set of resource constraints

$\alpha^k = \max(0, \sum_{i \in X} a_{ki} - b_k) \quad \forall k \in K$ is the surplus on constraint k

$J = \{k \in K \ / \ \alpha^k > 0\}$ is the set of constraints violated by closure X

The steps of the heuristic are as follows:

- Step 0
  - Initialize X and F
  - Calculate $\alpha^k \quad \forall k \in K$
- Step 1
  - $J = \{k \in K \quad : \quad \alpha^k > 0\}$, the set of violated constraints.
  - if $|J| = \emptyset$ stop, the current solution X is feasible.
  - $\beta^k = \alpha^k \quad \forall k \in J$
  - $\forall i \in F$ calculate the criterion r_i
- Step 2

  While $(\forall k \in J \quad \beta^k > 0)$
  - $i^* = \text{argmin}\{r_i, i \in F\}$
  - $X = X \setminus \{i^*\}$ and $F = F \setminus \{i^*\}$
  - For all $i \in \Gamma_{i^*}$ such that $F \cap \Gamma_i^{-1} = \emptyset$

    $F = F \cup \{i\}$

    calculate r_i
  - $\forall k \in J \quad \beta^k = \max(0, \beta^k - a_{ki^*})$

End while

Set $\alpha^k = \beta^k \forall k \in J$; go to step 1.

With each element of F, the set of closure frontier nodes, is associated a value r_i that serves as a basis for selecting the node to eliminate. The elements of F may be kept in a list sorted in increasing order by r_i. In step 2, therefore, the first element in this list is the candidate element i^*. In addition, new elements of F must be inserted at the right place in the list to maintain this order. Whenever a constraint ceases to be violated by the current closure (step 1), the r_i are recalculated for all elements of F and the list is resorted.

5.2.4.2 Selection criteria

A node in the frontier set is selected for elimination if doing so leads to the smallest reduction in the value of the current closure and contributes to the greatest reduction in surplus on the resource constraints. The surplus on resource constraint k, denoted α^k, is calculated as follows: $\alpha^k = \max(0, \sum_{i \in X} a_{ki} - b_k)$ where X is the set of nodes in the current closure. The contribution of a node element of X to reducing constraint surplus by $\sum_{k \in K} d^k a_{ki}$, where d^k is a weight associated with the constraint k (and a function of the surpluses α^k). The weights d^k may be calculated in two different ways:

$$(a) \qquad d^k = \alpha^k / (\sum_{k \in K} \alpha^k)$$

$$(b) \qquad d^k = \begin{cases} \lambda^k & \text{if } \alpha^k > 0, \\ 0 & \text{if } \alpha^k = 0 \end{cases}$$

The second method for calculating weights assumes knowledge of the dual multipliers associated with the resource constraints. Three criteria for selecting nodes for elimination are proposed:

- criteria 1: $r_i = c_i - \sum_{k \in K} d^k a_{ki}$ where d^k is calculated by using (a).
- criteria 2: $r_i = c_i / \sum_{k \in K} d^k a_{ki}$ where d^k is calculated by using (a).
- criteria 3: $r_i = c_i - \sum_{k \in K} d^k a_{ki}$ where d^k is calculated by using (b).

Criteria 1 and 2 are simple, intuitive criteria based on differences or ratios between reduction in the value of the closure and reduction in resource constraint surpluses. Criterion 3, which is based on the dual multipliers, makes use of reduced costs. Therefore, it measures variation in the upper bound obtained using these multipliers if node i is removed from the current closure. Therefore, using this criterion to eliminate a node from the current closure leads to the smallest decrease in the value of the current closure. The success of the heuristic when this criterion is used depends on the quality of the multipliers and of the initial closure.

5.2.5 Numerical results and conclusion

Numerical tests have been conducted on 15 groups of test problems, each containing ten problems. Problems in the same group have the same number of nodes and the same ratio R_m between the mean resource availability and mean resource consumption per node. This ratio is an estimate of the number of nodes making up the desired closure. Table 5.1 describes the groups with respect to the number of nodes and the ratio R_m. The group corresponding graphs used in test problems and those arising from mineral application adressed in section 5.2.2 are similar. These graphs are acyclic and have on average ten successors per node. That is, such a graph has $10n$ arcs, for n being its number of nodes. Three resource constraints are considered for each problem, with consumption at nodes generated randomly to fall between 0 and 10 with two decimal places. Node values are also generated randomly to lie between -10 and 10, with two decimal places.

The tests that will be presented here attempt to demonstrate the efficiency of the bundle method to adjust the dual multipliers when using lagrangian relaxation on maximal closure problem with resource constraints. We compare it to the commonly used subgradient method. We focus on bound quality and the required number of subproblem evaluations. The subproblem is a classical maximal closure problem and the number of times it is solved yields an excellent indicator of the overall computer time.

Results are divided in three parts:

The first part (see table 5.2) tries to determine the best proposed heuristic criteria. The following strategy was adopted to compare the three criteria. The heuristic was called using each criteria, using the last unfeasible solution obtained during application of the bundle method. For each criterion, the percentage gap from the initial unfeasible solution was calculated. Table 5.2 presents, for each group and by criterion, the mean such gap obtained. These results indicate that criterion 3, based on dual multipliers is the most satisfactory while gaps obtained using criteria 1 and 2 are similar and greater. Criteria 1 and 2 are useful, however, in cases where the dual multipliers are unknown.

In the second part of the results presented in table 5.3, we study the behaviour of the bundle method according to the degree of tolerance of the stopping criteria, on the different groups of problems considered.
In these tables, we use the following notations :
UB : upper bound on the optimal solution
LB : lower bound (best final feasible solution)
$\#eval$: total number of subproblem evaluations

Table 5.3 illustrate the changes in mean upper and lower bounds obtained by the bundle method, as a function of the tolerance $\underline{\epsilon}$ used in the stopping criterion, and present the mean number of subproblem evaluations for each group, as a function of the tolerance $\underline{\epsilon}$. These tables indicates that the gap between the mean upper bound and mean lower bound increases as the tolerance increases, for each problem group. A comparison of results obtained for $\underline{\epsilon} = 0.001$ and $\underline{\epsilon} = 0.1$ reveals that this increase in gaps is generally small and can be accounted for by an increase in the upper bound. The lower bounds obtained for the two tolerance values remain about the same. In addition, use of tolerance $\underline{\epsilon} = 0.1$ guarantees, for each group of problems, a reduction of between three and four in the mean number of subproblem evaluations, compared with tolerance $\underline{\epsilon} = 0.001$. In view of these results, a tolerance of 0.1 has been used in the stopping criterion of bundle method. This value provides lower bounds similar in quality to those of the other tolerance values and reduces the number of subproblem evaluations.

The third part, (see table 5.4) compares the bundle method and the subgradient method for solving maximal closure problem with resource constraints. Also we attempt , in this part, to highlight the contribution of the heuristic proposed to built feasible solutions. Table 5.4 compares the subgradient and bundle methods, by presenting the mean number of subproblem evaluations ($\#eval$) and the mean percentage gap ($\%Gap$) between the dual upper bound and the value of the best known feasible solution, for each group and each approach. Results for the bundle method were obtained by using $\underline{\epsilon} = 0.1$. This table also shows the reduction in gaps achieved by the heuristic based on criterion 3, when combined with the bundle method. These results indicate that solution of the problem of maximal closure with resource constraints on large-scale graphs requires an approach involving few subproblem evaluations. As can be seen from Table 5.4, the bundle method reduces

the geometric mean of the number of subproblem evaluations by 50%, compared with the classical subgradient approach. (The geometric mean is used to accommodate numbers of varying orders of magnitude.) This reduction implies considerable savings in computation time in practice, since the subproblems are very larges. Furthermore, the bundle method guarantees a high quality of dual multipliers and, generally, if a feasible solution is obtained during the resolution process the value of this solution is close to the optimal dual solution value. This behaviour is reflected in Table 5.4, which shows the smaller percentage gaps for the bundle method as opposed to the subgradient method. Further analysis of these results shows that out of the 15 groups of problems tested, the bundle method reduces the gaps of 11 of the tested groups in comparison to the gaps obtained by the subgradient method. As well, the number of subproblem evaluations in all the tested groups are smaller in the bundle method than in the subgradient method.

Gaps obtained by the bundle method are essentially integrality gaps, which may be reduced through heuristics that explore the neighbourhoods of unfeasible solutions to find feasible solutions that improve the lower bound on the optimal solution. As shown in Table 5.4, the simple heuristic proposed here to solve this problem reduces by 24% the geometric mean of integrality gaps found using the bundle method. Importantly, these gaps decrease with problem size and are on average less than 1% for problems of 10,000 nodes and over.

These results contradict the conclusion of the article by Zhao & Kim [26], who claim that Lagrangian relaxation is not suitable for this kind of problem. These authors, however, considered a very small problem with resource constraints expressed as equations. The present study has shown that Lagrangian relaxation is in fact suitable for maximal closure problems with resource constraints, and that

it yields very small gaps for large-scale problems. While it is possible to construct small problems demonstrating large gaps, most problems in practice (especially in mining industry applications) are quite large. As this article has shown, Lagrangian relaxation is highly appropriate for such problems and yields very small gaps.

While the Lagrangian relaxation approach using the bundle method to adjust multipliers may produce few feasible solutions, any unfeasible solutions obtained in later iterations violate the relaxed constraints only slightly. The heuristic can then adjust them to be feasible without causing their value to decline too much. The bundle method guarantees high quality dual multipliers and excellent solutions very close to the desired feasible solution, thus ensuring that the heuristic will be effective.

Table 5.1 Problem classification

	Average Ratio R_m					
No. nodes	10	30	100	300	1000	3000
100	G1	G2				
300	G3	G4	G5			
1000	G6	G7	G8	G9		
3000			G10	G11	G12	
10800				G13	G14	G15

Table 5.2 Comparison of criteria in the heuristic

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14
C1	17.20	17.10	14.38	8.30	8.55	15.26	15.25	30.42	29.80	3.59	3.52	33.56	1.06	11.41
C2	17.20	17.10	14.38	8.30	8.55	15.26	15.25	27.59	29.80	3.59	3.52	32.60	1.06	11.41
C3	9.33	10.32	12.36	4.93	3.29	18.26	12.61	23.14	21.52	2.61	0.90	28.77	0.78	7.03

Table 5.3 Behavior of the bundle method as a function of tolerance ϵ.

epsilon	Group G1			Group G2			Group G3		
ϵ	# eval	UB	LB	# eval	UB	LB	# eval	UB	LB
0.001	9	14.5302	13.7360	17	23.7708	21.1580	14	19.7188	18.1240
0.01	8	14.5330	13.7360	15	23.7727	21.1580	13	19.7207	18.1240
0.1	7	14.5431	13.7360	14	23.8459	21.1580	9	19.7525	18.1240
1.0	6	14.8631	13.7360	6	24.2776	21.1040	8	20.0724	17.5720
	Group G4			Group G5			Group G6		
0.001	12	47.7343	47.2380	8	79.2151	78.6860	11	19.3489	18.6840
0.01	11	47.7359	47.2380	6	79.2175	78.6860	12	19.3489	18.6840
0.1	9	47.7556	46.9180	6	79.2216	78.5960	9	19.3792	18.2720
1.0	8	47.9441	46.5000	5	79.5011	78.4160	8	19.5775	17.9041
	Group G7			Group G8			Group G9		
0.001	12	46.4566	45.6960	10	78.5695	76.4220	13	88.4046	85.4680
0.01	13	46.4567	45.6960	10	78.5696	76.4220	13	88.4046	85.4680
0.1	9	46.4738	45.5480	8	78.5737	76.3780	12	88.4237	85.4680
1.0	7	46.4341	45.4000	8	78.9841	76.2600	4	88.4299	85.4680
	Group G10			Group G11			Group G12		
0.001	11	155.1309	154.6280	12	250.4298	249.5320	13	292.5827	280.2480
0.01	9	155.1323	154.6280	11	250.4348	249.5320	13	292.5827	280.2480
0.1	8	155.1332	154.1560	9	251.8962	250.9020	9	292.8652	280.2480
1.0	6	155.4562	153.4040	8	250.8700	247.8580	8	292.7604	280.2480
	Group G13			Group G14			Group G15		
0.001	14	497.4017	496.8620	12	804.1505	800.5620	14	926.4075	900.5660
0.01	14	497.4024	496.6480	11	804.1524	800.5180	14	926.4075	900.5660
0.1	10	497.4076	496.3740	9	804.1580	799.3760	11	926.4089	900.5660
1.0	10	497.5681	494.2000	9	804.3284	795.4300	8	926.6805	900.5660

Table 5.4 Comparison of solution approaches

Group	Subgradient Method		Bundle Method		Heuristic and Bundle Method	
	# eval.	% Gap	# eval.	% Gap	# eval.	% Gap
1	15	9.11	7	6.83	7	5.43
2	31	10.47	14	10.44	14	10.44
3	13	23.32	9	10.25	9	8.30
4	12	3.17	9	4.55	9	1.72
5	12	0.71	6	0.73	6	0.63
6	12	10.83	9	3.53	9	3.40
7	11	5.28	9	1.99	9	1.99
8	26	3.24	8	3.20	8	2.77
9	29	3.02	12	3.14	12	3.14
10	12	0.80	8	0.80	8	0.64
11	21	1.29	9	1.16	9	0.39
12	44	4.00	11	3.98	9	2.21
13	12	1.57	10	0.31	10	0.21
14	25	0.69	9	0.49	9	0.43
15	45	2.76	11	2.70	11	2.70
Geometric Mean	18	3.18	9	2.29	9	1.74

References

[1] AHUJA, R.K., MAGNANTI, T.L., & ORLIN, J.B. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall.

[2] DAGDELEEN, K. & JOHNSON, T.B. (1986). Optimum Open Pit Mine Production Scheduling by Lagrangian Parameterization. *Proceedings of the 19th APCOM Symposium*, 127–141.

[3] DANTZIG, G.B. & WOLFE, P. (1960). The Decomposition Algorithm for Linear Programming. *Operations Research*, 8.

[4] DAVIS, R.E. & WILLIAMS, C.E. (1973). Optimization Procedures for Open Pit Mine Scheduling. *11th APCOM Symposium*, University of Arizona, Tucson, USA.

[5] EISNER, M.J. & SEVERANCE, D.G. (1976). Mathematical Techniques for Efficient Record Segmentation in Large Shared Databases. *J. Assoc. Comput. Mach.*, **23**, 619–635.

[6] ELEVELI, E., DAGDELEEN, K. & SALAMON, D.G. (1989). Single Time Period Production Scheduling of Open Pit Mines. *SME-AIME Meetings*, Las Vegas, Nevada, USA.

[7] FAALAND, B. & SCHMITT, T. (1987). Scheduling Tasks with Due Dates in a Fabrication/Assembly Process. *Oper. Res.*, **35**, 378–388.

[8] FAALAND, B., KIM, K. & SCHMITT, T. (1990). A New Algorithm for Computing the Maximal Closure of a Graph. *Management Science*, **36(3)**, 315–331.

[9] GALLO, G., GRIGORIADIS, M.D. & TARJAN, R.E. (1989). A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM J. Comput.*, **18**, 30–55.

[10] HIRIART-URRUTY, J. & LEMARECHAL, C. (1993). *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods.* A series of Comprehensive Studies in Mathematics, Springer-Verlag.

[11] KARZANOV, A.V. (1974). Determining the Maximal Flow in a Network by the Method of Preflows. *Soviet Mathematics Doklady* , **15**, 434–437.

[12] KELLEY, J.E. (1960). The Cutting Plane Method for Solving Convex Programs. *SIAM Journal*, **8**, 703–712.

[13] KIWIEL, K.C. (1990). Proximity Control in Bundle Methods for Convex Nondifferentiable minimization. *Math. Programming*, **46(1)**, 105–122.

[14] LEMARECHAL, C., NEMIROVSKI, A.S.& NESTEROV, Y. (1991). *New Variant of Bundle Methods and Applications.* Report RR, INRIA, France.

[15] LEMARECHAL, C., STRODIOT, J.J. & BIHAIN, A. (1981). On a Bundle Algorithm for Nonsmooth Optimization. *Nonlinear Programming J.* O.L. Mangasarian, R.R. Meyer and S.M. Robinson (Eds), Academic Press, 245–282.

[16] LERCHS, & GROSSMAN, (1965). Optimum Design of Open Pit Mines. *C.M.I.Bulletin*, **68** (no. 633), 47–54.

[17] MAMER, J.W. & SMITH, S.A. (1982). Optimizing Field Repair Kits Based on Job Completion Rate. *Management Science*, **28**, 1328–1333.

[18] PICARD, J.C. (1976). Maximal Closure of a Graph & Applications to Combinatorial Problem. *Management Science*, **22**.

[19] PICARD, J.C. & QUEYRANNE, M. (1982). Selected Applications of Minimum Cuts in Networks. *INFOR*, **20**, 394–422.

[20] RHYS, J.M.W. (1970). A Selection Problem of Shared Fixed Costs and Network Flows. *Management Science*, **17**, 200–207.

[21] SCHRAMM, H. & ZOWE, J. (1992). A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results. *SIAM Journal on Optimization* **2(1)**, 121–152.

[22] STEINER, G. (1986). An Algorithm to Generate the Ideals of a Partial Order. *Oper. Res. Lett.* **5**, 317–320.

[23] TACHEFINE, B. (1991). Détermination du plan d'extraction d'une mine à ciel ouvert. M.Sc.A. Thesis, École Polytechnique de Montréal, Canada.

[24] TACHEFINE, B. & SOUMIS, F. (1996). Étude comparative des algorithmes de flot maximum pour le problème des contours finaux d'une mine à ciel ouvert. *Cahiers du GERAD*, **G-96-30**

[25] WEINGARTNER, H.M. (1966). Capital Budgeting of Interrelated Projects: Survey and Synthesis. *Management Science*, **12**, 485–516.

[26] ZHAO, Y. & KIM, Y.C. (1993). Optimum Mine Production Sequencing Using Lagrangian Parameterization Approch. *Proceeding 24^{th} APCOM Symposium*, **2**, 176–189.

5.3 Conclusion

L'article présenté dans ce chapitre traite le problème de la fermeture maximale avec contraintes de ressource sur un graphe. Il présente une approche de résolution à ce problème basée sur la relaxation lagrangienne des contraintes de ressource. Dans cette approche, l'ajustement des multiplicateurs est effectué par la méthode de faisceaux et le problème lagrangien est résolu comme une fermeture maximale classique par un algorithme de flot maximum. Une heuristique simple est utilisée pour rendre les solutions du lagrangien non admissibles, réalisables aux contraintes de ressource dualisées.

Cet article montre que pour des problèmes tests générés aléatoirement, la méthode de faisceaux, comparativement à la méthode du sous-gradient, permet d'ajuster à l'optimalité les multiplicateurs en résolvant en moyenne 10 fois le problème de fermeture maximale. Il montre aussi que le gap entre la valeur de la solution primale réalisable déterminée par l'heuristique et la borne supérieure obtenue par la relaxation lagrangienne, est de l'ordre de 1.5% pour les problèmes tests.

Cet article qui considère le problème de planification réduit à une seule période montre que la relaxation lagrangienne utilisant une méthode d'ajustement des multiplicateurs efficace est une approche prometteuse pour le développement d'une méthode de résolution au problème complexe de la planification de l'exploitation dans une mine à ciel ouvert. L'extension de cette approche au problème global sera abordée au chapitre suivant en se basant sur le contenu de cet article.

CHAPITRE 6

Fermeture maximale sur un réseau multi-périodes avec contraintes de ressource: Application à la planification minière

6.1 Introduction

Considérons un graphe orienté dont les nœuds sont valorisés. Ces valeurs peuvent être positives, négatives ou nulles. On appelle fermeture sur ce graphe un ensemble de nœuds L du graphe tel que si un nœud appartient à L, tous ses successeurs appartiennent aussi à L. Une fermeture maximale est une fermeture sur le graphe dont la somme des valeurs des nœuds est la plus grande possible. Le problème de fermeture maximale a été largement abordé dans la littérature[37, 38] et plusieurs approches de résolution lui ont été proposées. Nous utilisons l'approche de Picard [37] qui consiste à le résoudre comme un problème de flot maximum sur un graphe étendu formé à partir du graphe sur lequel la fermeture est recherchée. Cette approche est simple et systématique, comparativement aux autres approches proposées dans la littérature.

Le problème de la fermeture maximale avec contraintes de ressource est un problème de fermeture maximale classique sur un graphe auquel sont ajoutées des contraintes sur la consommation des ressources au niveau des nœuds. La fermeture recherchée doit être de valeur maximale par rapport à toutes les fermetures qui respectent les disponibilités des ressources. Pour traiter les problèmes de fermeture maximale avec contraintes de ressource, nous avons proposé une approche

de résolution basée sur la relaxation lagrangienne des contraintes de ressource [43]. Dans cette approche, l'ajustement des multiplicateurs de Lagrange est effectué par la méthode de faisceaux et le problème de fermeture maximale classique est résolu par un algorithme de flot maximum. Pour rendre les solutions obtenues par cette approche réalisables, nous avons incorporé au processus de résolution précédent une heuristique d'exploration qui permet d'obtenir des solutions satisfaisant les contraintes de ressource à partir de solutions du problème relaxé. La relaxation lagrangienne accompagnée de l'heuristique proposée présentent d'excellentes solutions avec des gaps entre les bornes supérieure et inférieure très petits.

Dans ce chapitre, nous considérons un graphe dont les nœuds sont valorisés sur plusieurs périodes d'un horizon de temps. Nous supposons qu'à chaque période est associé un ensemble de contraintes de ressource. Nous cherchons à sélectionner un ensemble de nœuds pour chaque période de façon à maximiser la valeur totale des ensembles sélectionnés. L'ensemble des nœuds sélectionnés pour une période doit satisfaire les contraintes de ressource de la période. En plus, si un nœud est sélectionné à une période, tous ses successeurs dans le graphe devraient être sélectionnés à la même période ou déjà sélectionnés aux périodes antérieures. Nous imposons dans ce problème qu'un nœud du graphe ne peut être sélectionné qu'une seule fois durant toutes les périodes. Ce problème est un problème de fermeture maximale sur un réseau multi-périodes en présence de contraintes de ressource.

Ce problème de fermeture maximale sur un réseau multi-périodes en présence de contraintes de ressource est formulé à la section 6.2. L'approche de résolution sera présentée à la section 6.3. La section 6.4 présente les résultats obtenus sur des problèmes tests générés aléatoirement et, à la section 6.5, nous présentons une application de ce problème en planification minière et donnons les résultats obtenus

sur un jeu de données tirées de la vie réelle. Les conclusions sont présentées à la dernière section de ce chapitre.

6.2 Description du Problème

Considérons un graphe orienté $G(N, A)$ où N est l'ensemble des nœuds et A est l'ensemble des arcs. En considérant K ressources consommées au niveau des nœuds de ce graphe, soit a_{ki} $k = 1, \ldots, K$ la consommation de la ressource k au niveau du nœud i, et b_k^p la disponibilité de cette ressource k à la période p $(p = 1, \ldots, P)$. En associant à chaque nœud i du graphe une valeur c_i^p, pour chaque période p, qui peut être positive, négative ou nulle, le problème traité dans ce chapitre cherche à déterminer pour chaque période p un ensemble de nœuds E^p tel que $\cup_{q=1}^{p} E^q$ forme une fermeture sur G et tel que la consommation de chaque ressource k par l'ensemble des nœuds E^p respecte la disponibilité de la ressource b_k^p à la période p. La valeur totale des ensembles à déterminer $\sum_{p=1}^{P} \sum_{i \in E^p} c_i^p$ doit être maximale.

Ce problème se présente comme un problème de fermeture maximale en présence de contraintes de ressource sur un réseau multi-périodes constitué de P copies du graphe G, identifiées pour chaque période $p = 1, ..., P$ par $G^p(N^p, A^p)$ où :

$$N^p = \{i/p \quad : \quad i \in N\} \text{ et } A^p = \{(i/p, j/p) \quad : \quad (i,j) \in A\}.$$

Les éléments de N^p sont notés i/p plutôt que d'utiliser (i, p), la notation classique du produit d'ensemble, pour éviter la confusion avec les arcs de l'ensemble A. D'autre part, le réseau G comprend l'ensemble des arcs $(i/p, i/p + 1)$ où $i \in N$ et $p = 1, \ldots, P - 1$ qui relient tout nœud i/p du graphe G^p à son semblable $i/p + 1$ dans le graphe G^{p+1}.

La consommation d'une ressource k au niveau d'un nœud i du graphe G est considérée comme la consommation associée à chacun des nœuds i/p $(p = 1, \ldots, P)$ du réseau. De même la valeur c_i^p associée au nœud i de G à la période p devient la valeur associée au nœud i/p du réseau.

À cause de la présence des arcs inter-périodes dans le réseau, l'ensemble des nœuds S^p sélectionnés dans la fermeture du graphe G^p est inclus dans l'ensemble des nœuds S^{p+1} sélectionnés dans la fermeture du graphe G^{p+1}. Les ensembles recherchés sur le graphe G, E^p $(p = 1, \ldots, P)$ sont déterminés par différence des ensembles S^p comme suit: $E^p = S^p \setminus S^{p-1}$ pour $p = 2, \ldots, P$ et $E^1 = S^1$. Les deux figures 6.1 et 6.2 illustrent ces relations.

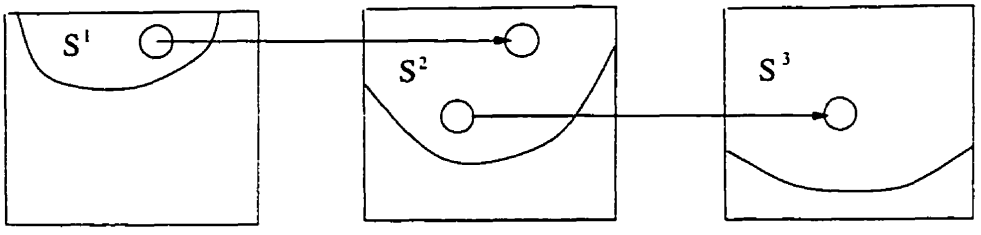


Figure 6.1 Exemple de fermeture sur un réseau multi-périodes

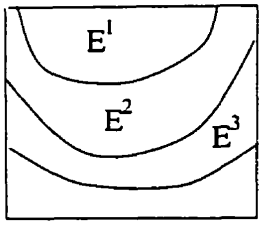


Figure 6.2 Équivalence de la fermeture sur le réseau avec les ensembles E^p recherchés sur le graphe G

Ce problème de fermeture sur le réseau multi-périodes se formule en utilisant les variables suivantes:

$$x_i^p = \begin{cases} 1 & \text{si le nœud } i/p \text{ appartient à la fermeture,} \\ 0 & \text{sinon.} \end{cases}$$

comme suit:

$$(P) \quad \max \sum_{i \in N} c_i^1 x_i^1 + \sum_{p=2}^{P} \sum_{i \in N} c_i^p (x_i^p - x_i^{p-1}) \tag{6.1}$$

sujet à:

$$\sum_{i \in N} a_{ki} x_i^1 \leq b_k^1, \quad k = 1, ..., K$$

$$\sum_{i \in N} a_{ki} (x_i^p - x_i^{p-1}) \leq b_k^p, \quad p = 2, ..., P, \quad k = 1, ..., K \tag{6.2}$$

$$x_i^p - x_j^p \leq 0 \quad \text{pour} \quad (i/p, j/p) \in A^p, \quad p = 1, ..., P \tag{6.3}$$

$$x_i^p - x_i^{p+1} \leq 0 \quad \text{pour} \quad i/p \in N^p \quad \text{et} \quad i/p+1 \in N^{p+1}, \quad p = 1, ..., P-1 \tag{6.4}$$

$$x_i^p \in \{0,1\}, \quad i \in N, \quad p = 1, ..., P. \tag{6.5}$$

Dans cette formulation, les coûts et les contraintes pour une période p portent sur les ensembles E^p. L'objectif (6.1) de cette formulation se réécrit, en définissant $\overline{c}_i^p = c_i^p - c_i^{p+1}$ pour $p = 1, \ldots, P-1$ et $\overline{c}_i^P = c_i^P$, comme suit: $\max \sum_{p=1}^{P} \sum_{i \in N} \overline{c}_i^p x_i^p$. Cet objectif maximise la valeur totale de la fermeture. Les contraintes (6.2) représentent les contraintes de ressource pour les différentes périodes. L'ensemble des contraintes (6.3) traduit les relations de précédence entre les nœuds des graphes G^p. Ces contraintes sont représentées, pour chaque graphe G^p correspondant à la période p, par l'ensemble des arcs A^p. Les contraintes (6.4) présentent les relations de précédence entre les périodes. Ces contraintes (6.4) imposent que la sélection d'un nœud i/p du graphe G^p dans la fermeture sur le réseau multi-périodes soit au préalable précédé par la sélection de ses semblables aux périodes antérieures.

Ce problème trouve une importante application en planification minière. Un gisement minier est habituellement subdivisé en blocs parallélipédiques où chaque bloc est accompagné de l'ensemble des informations pertinentes pour la prise de décision de son exploitation. De même, à chaque bloc est associée une valeur qui peut être positive, négative ou nulle qui représente le profit tiré ou la perte encourue s'il est exploité. La planification de l'exploitation d'un tel gisement sur un horizon de plusieurs périodes consiste à déterminer l'ensemble des blocs à exploiter durant chacune des périodes de façon à maximiser les revenus générés à la fin de l'horizon. Dans cette application les graphes en jeu sont de très grande taille, soit une centaine de milliers de nœuds et un million d'arcs. La taille du réseau de planification est P fois la taille d'un graphe d'une seule période.

Cette planification doit tenir compte de deux ensembles de contraintes:

- L'ensemble des contraintes physiques qui imposent qu'un bloc ne doit être exploité que s'il est à découvert (généralement on considère qu'un bloc est contraint par ses neuf voisins du niveau supérieur). Ces contraintes sont traduites par les arcs du réseau définissant les relations de précédence entre les blocs représentés par les nœuds.

- L'ensemble des contraintes de ressource qui définissent les capacités des installations pour chaque période de l'horizon de planification.

6.3 Approche de Résolution

La formulation (P) précédente est celle d'un problème de fermeture maximale sur un réseau multi-périodes de grande taille en présence de contraintes de ressource. L'approche que nous avons déjà utilisée pour traiter ce type de problème sur un graphe de taille raisonnable [43] pourraît être employée mais en pratique la taille du réseau représente une difficulté majeure. Il est quasiment impossible de prendre en mémoire d'une machine un réseau de cette taille pour y déterminer une fermeture par un algorithme de flot maximum.

L'approche de résolution qui sera présentée dans cette section utilise la relaxation lagrangienne des contraintes de ressource qui conduit à un problème de fermeture maximale classique sur un réseau multi-périodes. Afin de contourner la difficulté liée à la taille du réseau, nous relaxons l'ensemble des contraintes inter-périodes pour se ramener à P graphes indépendants (P est le nombre de périodes) où sur chaque graphe qui est de taille raisonnable, il est possible de déterminer une fermeture maximale. Cette approche combinant la relaxation lagrangienne des contraintes de ressource et l'élimination des contraintes inter-périodes, fournie une borne supérieure sur la solution optimale du problème (P). Pour déterminer une borne inférieure représentant la valeur d'une solution réalisable, on utilise une heuristique qui en cherchant à satisfaire les contraintes de ressource relaxées, tient compte des contraintes inter-périodes.

6.3.1 Borne Supérieure

Pour calculer une borne supérieure au problème (P), nous relaxons les contraintes inter-périodes (6.4). Ainsi on se retrouve avec un problème ($P1$) constitué de l'objectif (6.1) et des contraintes (6.2), (6.3) et (6.5). En définissant les termes

suivants:

A : matrice (K x n) des coefficients des contraintes de ressource;

X^p : vecteur des variables, $X^p = (x_1^p, x_2^p, \ldots, x_n^p)^T$;

$\overline{C}^p$: vecteur des coûts à la période p, $\overline{C}^p = (\overline{c}_1^p, \overline{c}_2^p, \ldots, \overline{c}_n^p)$;

b^p : vecteur des disponibilités des ressources, $b^p = (b_1^p, b_2^p, \ldots, b_K^p)^T$;

D : domaine des vecteurs X^p, $D = \{X^p \quad : \quad x_i^p \leq x_j^p \ \forall\ (i/p, j/p) \in A^p$ et $x_i^p \in \{0,1\}\ \ i \in N\}$.

Le problème ($P1$) s'écrit de façon plus compacte:

$$
\begin{array}{lccccccccccc}
 & \max & \overline{C}^1X^1 & + & \overline{C}^2X^2 & + & \ldots & + & \overline{C}^PX^P & & \\
 & & AX^1 & & & & & & & \leq & b^1 \\
\text{(P1)} & - & AX^1 & + & AX^2 & & & & & \leq & b^2 \\
 & & & - & AX^2 & + & AX^3 & & & \leq & b^3 \\
 & & & & & & - \quad AX^{P-1} & + & AX^P & \leq & b^P
\end{array}
$$

$$X^p \in D \quad \forall p = 1, \ldots, P.$$

Pour déterminer une borne supérieure pour le problème ($P1$), qui est aussi une borne supérieure au problème (P), on procède par relaxation lagrangienne des contraintes. Ainsi, en associant à chaque groupe de contraintes p le vecteur de multiplicateurs $\lambda^p \geq 0$, le dual du problème ($P1$) s'écrit:

$$(DP1) \quad \min_{\lambda \geq 0} f(\lambda^1, \lambda^2, \ldots, \lambda^P) \text{ où}$$

$$f(\lambda^1, \lambda^2, \ldots, \lambda^P) = \max_{X \in D} \sum_{p=1}^{P} \overline{C}^p X^p + \lambda^1(b^1 - AX^1) + \sum_{p=2}^{P} \lambda^p(b^p - AX^p + AX^{p-1}).$$

L'optimisation de la fonction duale $f()$ par la méthode de faisceaux ou par une méthode équivalente prend beaucoup de temps de calcul (voir section 6.4). Pour remédier à cette difficulté et afin de pouvoir traiter des problèmes de plusieurs

périodes en un temps raisonnable, nous résolvons le dual d'une relaxation de (P1).

Considérons la relaxation $(RP1)$ du problème $(P1)$ obtenue en remplaçant le bloc de contraintes de second membre b^p par l'addition des p premiers blocs de contraintes.

$$
\begin{array}{lrcl}
\max & \overline{C}^1X^1 + \overline{C}^2X^2 + \ldots + \overline{C}^PX^P & & \\
 & AX^1 & \leq & b^1 \\
\text{(RP1)} & AX^2 & \leq & b^1 + b^2 \\
 & AX^P & \leq & b^1 + b^2 + \ldots + b^P \\
 & X^p \in D \quad \forall p = 1, \ldots, P. & &
\end{array}
$$

En associant à chaque groupe de contraintes p, le vecteur de multiplicateurs $\alpha^p \geq 0$, le dual de $(RP1)$ s'écrit:

$$(DRP1) \quad \min_{\alpha \geq 0} g(\alpha^1, \alpha^2, \ldots, \alpha^P) \quad \text{où}$$

$$g(\alpha^1, \alpha^2, \ldots, \alpha^P) = \max_{X \in D} \sum_{p=1}^{P} \overline{C}^p X^p + \sum_{p=1}^{P} \alpha^p (\sum_{q=1}^{p} b^q - AX^p).$$

Notons que le dual de $(P1)$, $\min_{\lambda \geq 0} f(\lambda^1, \lambda^2, \ldots, \lambda^P)$ est équivalent au problème:

$$(DP1) \quad \min\{g(\alpha^1, \alpha^2, \ldots, \alpha^P) \ : \ \sum_{q=p}^{P} \alpha^q \geq 0 \ ; \ p = 1, \ldots, P\},$$

qui est obtenu en posant le changement de variables $\lambda^p = \sum_{q=p}^{P} \alpha^q$ pour $p = 1, \ldots, P$ dans le dual de $(P1)$. La résolution de ce nouveau problème dual (DP1 sous sa forme $g()$) par la méthode de faisceaux prend moins de temps que celle de son équivalent (DP1 sous sa forme $f()$), mais elle reste très lente pour le traitement des problèmes de grande taille. Dans ce nouveau problème, la fonction $g()$ est séparable par période mais la présence de contraintes couplantes sur les variables vient diminuer l'avantage de cette propriété.

Nous avons opté pour la résolution du dual de la relaxation ($RP1$) qui n'est autre que le problème ($DP1$) dans lequel les contraintes couplantes sur les variables α sont remplacées par des contraintes de non négativité. Nous montrons dans le développement qui suit, qu'à toute solution duale de la relaxation on peut construire une solution duale de même coût au problème initial ($P1$). En plus, nous déterminons des conditions permettant de vérifier si une solution duale optimale du problème relaxé est duale optimale pour (P1). Si ces conditions ne sont pas vérifiées, on montre que l'optimisation de la fonction duale $f()$ peut être initialisée avec la solution optimale duale de la relaxation et que l'ensemble des sous-gradients accumulés lors de l'optimisation de la relaxation se convertit en un ensemble de sous-gradients pour la fonction duale $f()$.

Pour faire ressortir les liens entre les deux fonctions duales f et g, nous montrons les propriétés suivantes:

- **propriété 1:**

$$\min_{\lambda \geq 0} f(\lambda^1, \lambda^2, \ldots, \lambda^P) \leq \min_{\alpha \geq 0} g(\alpha^1, \alpha^2, \ldots, \alpha^P)$$

- **propriété 2:** À tout point $(X, \alpha^1, \alpha^2, \ldots, \alpha^P)$ où la valeur de la fonction duale $g()$ est $\overline{g}$, correspond le point $(X, \lambda^1, \lambda^2, \ldots, \lambda^P)$ défini par:

$$\lambda^p = \sum_{q=p}^{P} \alpha^q \quad \forall p = 1, \ldots, P$$

et tel que la valeur de la fonction duale $f()$ en ce point est $\overline{g}$.

- **propriété 3:** Soit s_g un sous-gradient de $g()$ au point $(\overline{\alpha}^1, \overline{\alpha}^2, \ldots, \overline{\alpha}^P)$, défini par $s_g^p = \sum_{q=1}^{p} b^q - AX^p \quad \forall\, p = 1, \ldots, P$, alors la fonction duale $f()$ admet un

sous-gradient s_f défini par $s_f^1 = s_g^1$ et $s_f^p = s_g^p - s_g^{p-1}$ pour $p = 2, \ldots, P$ au point $(\overline{\lambda}^1, \overline{\lambda}^2, \ldots, \overline{\lambda}^P)$ satisfaisant les relations de la propriété 2.

Les propriétés 1,2 et 3 se déduisent du fait que la fonction duale $g()$ est obtenue de la fonction duale $f()$ par changement de variables.

- **propriété 4:** Soient $(\overline{\alpha}^1, \overline{\alpha}^2, \ldots, \overline{\alpha}^P)$ les multiplicateurs duaux optimaux associés à $(RP1)$. Les multiplicateurs correspondant de $(P1)$, $(\overline{\lambda}^1, \overline{\lambda}^2, \ldots, \overline{\lambda}^P)$ déterminés par les relations de la propriété 2 vérifient la relation suivante: $\overline{\lambda}^1 \geq \overline{\lambda}^2 \geq \ldots \geq \overline{\lambda}^P \geq 0$. Ces multiplicateurs sont optimaux pour $(P1)$ sauf s'il existe un q tel que $\overline{\lambda}^q = \overline{\lambda}^{q+1}$ et $\overline{\lambda}^P > 0$.
  preuve: La relation $\overline{\lambda}^1 \geq \overline{\lambda}^2 \geq \ldots \geq \overline{\lambda}^P \geq 0$ se déduit des relations de la propriété 2 car les $\overline{\alpha}^p$ sont non négatifs. Cette relation se vérifie suivant 4 cas de figures principaux:
  cas 1 : $\overline{\lambda}^1 > \overline{\lambda}^2 > \ldots > \overline{\lambda}^P > 0$
  cas 2 : $\overline{\lambda}^1 = \overline{\lambda}^2 = \ldots = \overline{\lambda}^P = 0$
  cas 3 : $\overline{\lambda}^1 > \overline{\lambda}^2 > \ldots > \overline{\lambda}^q = \ldots = \overline{\lambda}^P = 0$
  cas 4 : $\overline{\lambda}^1 > \overline{\lambda}^2 > \ldots > \overline{\lambda}^q = \overline{\lambda}^{q+1} > \ldots > \overline{\lambda}^P > 0$
  Pour les 3 premiers cas, les multiplicateurs vérifient les conditions de complémentarité du problème $(P1)$. Dans le dernier cas, les conditions peuvent ne pas être satisfaites, donc les multiplicateurs peuvent ne pas être optimaux pour le dual de $(P1)$. Dans les 3 premiers cas, la résolution du dual $(DRP1)$ est équivalente à celle de $(DP1)$. Dans le dernier cas, une ré-optimisation du dual de $(DP1)$ à partir de la solution optimale de $(DRP1)$ est nécessaire pour déterminer les multiplicateurs optimaux.

La propriété 4 suggère qu'à la suite de l'optimisation de la fonction duale $g()$ du problème relaxé, on peut facilement vérifier si la solution optimale obtenue est

optimale pour la fonction duale $f()$ du problème $(P1)$. Si pour un type de contraintes k, la situation suivante se présente:

$$\overline{\alpha}_k^P > 0 \text{ et il existe } q < P \text{ tel que } \overline{\alpha}_k^q = 0$$

, alors l'optimum de $f(.)$ n'est pas atteint. Dans ce cas, les propriétés 2 et 3 nous permettent d'utiliser toute l'information accumulée lors de l'optimisation de $g()$ pour initialiser celle de $f()$ dont la solution optimale ne devrait pas être éloignée de celle de $g()$ déjà obtenue.

6.3.1.1 Optimisation de la fonction duale $g(.)$

L'optimisation de la fonction duale $g()$ associée au problème $(RP1)$ est effectuée par la méthode de faisceaux. La fonction $g()$ est une fonction décomposable par période. Le problème correspondant à une période p s'écrit comme suit:

$$\min_{\alpha^p \geq 0} Z^p(\alpha^p) \quad \text{où}$$

$$Z^p(\alpha^p) = \max_{X^p \in D} \sum_{i \in N} (\overline{c}_i^p - \sum_{k=1}^{K} \alpha_k^p a_{ki}) x_i^p + \sum_{k=1}^{K} \alpha_k^p (\sum_{q=1}^{p} b_k^q).$$

La fonction $Z^p(\alpha^p)$ est une fonction convexe linéaire par morceaux qui admet en tout point α_j un sous-gradient s_j dont la composante k $(k = 1, ..., K)$ est donnée par : $s_j^k = \sum_{q=1}^{p} b_k^q - \sum_{i \in N} a_{ki} \overline{x}_i^p$, où $\overline{x}_i^p$ $(i \in N)$ est la composante correspondant au nœud i du vecteur solution d'un problème de fermeture maximale classique sur le graphe G^p correspondant à la période p. Dans ce graphe, la valeur associée à un nœud i/p est donnée par $\overline{c}_i^p - \sum_{k=1}^{K} \alpha_k^p a_{ki}$. Nous résolvons le problème de fermeture maximale classique sur le graphe G^p par un algorithme de flot maximum.

Pour l'optimisation de la fonction $Z^p(\alpha^p)$, nous avons proposé dans l'article [43] la méthode de faisceaux pour la détermination des multiplicateurs optimaux. Nous y avons comparé la méthode de faisceaux et celle du sous-gradient. La méthode de faisceaux est une méthode itérative qui utilise un faisceau d'informations accumulées au cours des itérations; elle présente une réduction considérable du nombre d'évaluations de la fonction duale nécessaire comparativement à la méthode du sous-gradient généralement utilisée. Les résultats numériques présentés dans [43] et qui ont porté sur des graphes présentant moins de 10,000 nœuds montrent la supériorité de cette méthode. Nous utilisons ici la même méthode sur des problèmes avec des graphes dont la taille peut être supérieure à la centaine de milliers de nœuds.

L'optimisation de la fonction duale $g()$ période par période nous donne une borne supérieure sur la solution optimale de $(P1)$. Cette borne est calculée comme suit:

$$Z_{sup} = \sum_{p=1}^{P} Z^p(\overline{\alpha}^p) \text{ où } \overline{\alpha}^p \in argmin\{Z^p(\alpha^p) \ : \ \alpha^p \geq 0\}.$$

6.3.2 Heuristique pour obtenir une solution réalisable

Pour déterminer une solution réalisable au problème initial contenant les contraintes de ressource et les contraintes inter-périodes, nous utilisons une heuristique en deux phases. La première phase, qui est une généralisation de l'heuristique glouton présentée dans [43], détermine à partir d'une fermeture sur un graphe, une nouvelle fermeture réalisable pour les contraintes de ressource. Elle procède par élimination de nœuds à la frontière de la fermeture en se basant sur les coûts réduits. La deuxième phase utilise une recherche tabu simple pour améliorer la valeur de la solution réalisable obtenue lors de la première phase de l'heuristique.

Nous utilisons cette heuristique à chaque itération du processus de résolution

du problème décomposable, qui est une relaxation du dual du problème initial dans lequel des contraintes inter-périodes ont été supprimées. Ainsi, lors du traitement de la période p $(p = 1, \ldots, P)$ et à chaque itération de la méthode de faisceaux qui détermine des multiplicateurs aux contraintes de ressource, l'heuristique détermine pour cette période une solution qui satisfait en même temps les contraintes de ressource de la période p et les contraintes inter-périodes entre $p-1$ et la période p. Ceci est obtenu en considérant que la solution réalisable pour la période $p-1$ est fixée quand on traite la période p. Ainsi, l'élimination d'un nœud à la période p ne peut donc se faire que s'il ne fait pas partie de la solution de la période $p-1$ qui est fixée.

Le fait de considérer les solutions des périodes précédentes comme des solutions fixées quand on traite une période p permet à l'heuristique d'évaluer les disponibilités des ressources à la période p en vue d'obtenir une solution satisfaisant les contraintes de ressource à cette période. En plus, cette approche exploite correctement la structure des coûts dans les problèmes de planification minière qui nous intéressent. Cette structure favorise l'extraction d'un bloc rentable le plus tôt possible, plutôt que de retarder son extraction à des périodes futures. L'utilisation de cette heuristique nous amène à construire une solution globale réalisable en construisant progressivement pour chaque période une solution réalisable pour les contraintes de ressource et pour les contraintes inter-périodes.

Pour présenter cette heuristique, soit α^p un vecteur de multiplicateurs associés aux contraintes de ressource de la période p et X^p le vecteur solution du problème :

$$\max_{X^p \in D} \overline{C}^p X^p + \alpha^p (\sum_{q=1}^{p} b^q - A X^p).$$

Appelons S^p l'ensemble des nœuds du graphe défini par la solution X^p, $S^p = \{i \ : \ /x_i^p = 1\}$ et $\overline{S}^{p-1}$ l'ensemble des nœuds du graphe sélectionnés dans une solution réalisable déterminée par l'heuristique à la période $p-1$. De même nous notons $\overline{X}^{p-1}$ le vecteur solution associé à l'ensemble $\overline{S}^{p-1}$

Pour présenter les deux phases de cette heuristique on définit les ensembles suivants:

$$F^- = \{i \ : \ i \in S^p \ et \ \forall \ j \in \Gamma_i^{-1}, \ j \notin S^p \ \}.$$

Cet ensemble désigne l'ensemble des nœuds à la frontière intérieure de la fermeture définie par S^p. Ces nœuds sont candidats à être retirés.

$$F^+ = \{i \ : \ i \notin S^p \ et \ \forall \ j \in \Gamma_i, \ j \in S^p \ \}.$$

Cet ensemble désigne l'ensemble des nœuds à la frontière extérieure de la fermeture définie par S^p. Les nœuds de cet ensemble peuvent être ajoutés à la fermeture définie par S^p sans violer les contraintes de précédence entre les nœuds.

L'heuristique est composée de deux phases. La première convertit la fermeture définie par l'ensemble des nœuds S^p en une fermeture satisfaisant les contraintes de ressource et les contraintes inter-périodes liant la période $p-1$ et la période p. Cette phase fonctionne suivant une approche gloutonne qui retire des nœuds, en se basant sur un critère de sélection, de la fermeture courante jusqu'à satisfaction des contraintes. Les étapes de cette phase sont:

- **Étape 0 :**

  $S^p = S^p \cup \overline{S}^{p-1}$

  $\overline{b}^p = b^p + A\overline{X}^{p-1}$

  Former F^-

- **Étape 1 :**

  $J = \{k \ : \ \sum_{i \in S^p} a_{ki} > \overline{b}_k^p\}$, ensemble des contraintes violées par la fermeture courante.

  Si $|J| = 0$ stop, la fermeture courante est réalisable.

  Calculer $r_i = \overline{c}_i^p - \sum_{k \in J} a_{ki} \alpha_k^p$ pour tout $i \in F^-$

- **Étape 2 :**

  Tant que ($|J|$ n'a pas changé) faire:

  $i^* \in argmin\{r_i \ : \ i \in F^- \ et \ i \notin \overline{S}^{p-1}\}$

  $S^p = S^p \setminus \{i^*\}$ et $x_{i^*}^p = 0$

  Pour tout $i \in \Gamma_{i^*}$ tel que $F^- \cap \Gamma_i^{-1} = \emptyset$

  $F^- = F^- \cup \{i\}$

  calculer r_i

  Mettre à jour J

  Aller à l'étape 1.

La deuxième phase utilise une recherche tabu simple qui permet d'enlever ou d'ajouter des nœuds à la fermeture courante en vue d'améliorer sa valeur. L'idée de base d'une recherche tabu telle que décrite dans [16, 17] est de définir un ensemble de solutions possibles, et en partant d'une solution courante de trouver une autre meilleure par exploration de son voisinage. Dans notre cas une solution de départ est définie par la fermeture trouvée durant la phase gloutonne. Cette solution est définie par l'ensemble des nœuds du graphe S^p de valeur $val(S^p) = \sum_{i \in S^p} \overline{c}_i^p$. Cette solution de départ satisfait les contraintes de ressource du problème et définit deux ensembles frontières F^+ et F^-. Nous définirons le voisinage de cette solution comme l'ensemble des fermetures qui peuvent être obtenues par ajout d'un nœud appartenant à l'ensemble F^+ ou par retrait de la fermeture d'un nœud appartenant

à F^-. L'évaluation d'un élément du voisinage d'une solution courante définie par l'ensemble S^p est faite comme suit:

si $j \in F^-$ $v(S^p \setminus \{j\}) = \sum_{i \in S^p \setminus \{j\}} \overline{c}_i^p - M \sum_{k=1}^K \max(0, \sum_{i \in S^p \setminus \{j\}} a_{ki} - b_k^p)$

si $j \in F^+$ $v(S^p \cup \{j\}) = \sum_{i \in S^p \cup \{j\}} \overline{c}_i^p - M \sum_{k=1}^K \max(0, \sum_{i \in S^p \cup \{j\}} a_{ki} - b_k^p)$

où le paramètre M désigne une grande valeur positive.

La clé d'une recherche tabu est qu'elle permet de dégrader l'objectif afin d'éviter d'être pris dans un optimum local. Pour éviter les recherches cycliques, une solution récemment obtenue est considérée tabu pendant un certain nombre d'itérations.

L'utilisation de cette heuristique constituée de la phase gloutonne et de la phase tabu dans le cadre de l'optimisation du problème dual décomposable se fait suivant le schéma suivant:

1. Initialisation

   $\overline{S}^0 = \emptyset$ et $Z_{inf}^p = -\infty$ $(p = 1, \ldots, P)$ où P est le nombre de périodes.

   Poser $p = 1$

2. A chaque itération de la méthode de faisceaux utilisée pour la résolution de :

$$\min_{\alpha^p} \max_{X^p \in D} \overline{C}^p X^p + \alpha^p (\sum_{q=1}^{p} b^q - A X^p)$$

   Soit α^p le vecteur de multiplicateurs courant et X^p la solution correspondante:

   - Appliquer les deux phases de l'heuristique, pour déterminer une solution réalisable $\underline{S}^p$ de coût $Z = \sum_{i \in \underline{S}^p} \overline{c}_i^p$.
   - si $Z > Z_{inf}^p$ alors $Z_{inf}^p = Z$ et $\overline{S}^p = \underline{S}^p$

3. si $p < P$ poser $p = p+1$ et aller à l'étape 1, sinon stop. La valeur de la solution réalisable globale est $Z_{inf} = \sum_{p=1}^{P} Z_{inf}^p$ et les ensembles solutions sont définis par $\overline{S}^p$ $(p = 1, \ldots, P)$.

Les ensembles solutions $\overline{S}^p$ satisfont les contraintes de ressource pour chaque période p et les contraintes inter-périodes car ils vérifient $\overline{S}^p \subseteq \overline{S}^{p+1}$ pour $p = 1, \ldots, P-1$.

6.4 Résultats Numériques

Pour valider l'approche de résolution que nous proposons dans ce chapitre pour les problèmes de fermeture maximale sur un réseau multi-périodes en présence de contraintes de ressource, nous avons traité des graphes générés aléatoirement. Comme l'objectif principal du développement de cette approche est de pouvoir traiter des problèmes provenant des applications minières, nous avons considéré des graphes de même nature que ceux qu'on rencontre dans ces applications minières. Ces graphes orientés présentent en moyenne dix successeurs par nœud soit un nombre total d'arcs de $10n$ où n est le nombre de nœuds du graphe.

Les facteurs importants que nous avons considérés pour classifier les problèmes générés sont la taille en terme de nombres de nœuds, le nombre de périodes et un ratio r_m mesurant le rapport de la disponibilité moyenne des ressources sur la contribution moyenne d'un nœud à ces contraintes de ressource. Ce ratio mesure également de façon approximative le nombre de nøeuds du graphe devant faire partie de l'ensemble de nœuds recherché durant chacune des périodes. Nous générons aléatoirement les contributions des nœuds aux contraintes de ressource a_{ki} entre 0 et 10 avec deux chiffres significatifs et les valeurs associés aux nœuds à la première période c_i^1 entre -1.0 et 1.0 avec deux chiffres significatifs. Ces valeurs sont générées de façon à avoir un certain pourcentage des nœuds du graphe munies de valeurs négatives. A la lumière des données sur des gisements en exploitation, nous avons considéré un pourcentage de 33% pour les problèmes tests. Les valeurs pour les autres périodes sont calculés par la formule $c_i^p = c_i^1/(1+d)^p$ où $d = 0.1$ qui simule

la dépréciation des revenus d'exploitation en fonction du temps généralement considérée dans les modèles de calcul de revenus en usage dans les exploitations minières. Pour chaque problème, nous avons considéré 5 contraintes de ressource par période.

L'expérimentation est subdivisée en trois parties. Dans un premier temps, nous comparons la qualité de la borne supérieure obtenue en résolvant le problème dual de la relaxation ($RP1$) à la borne supérieure qu'on pourrait obtenir en résolvant le dual du problème original (P). Le tableau 6.1 présente les bornes (Z_{sup}), le nombre de résolutions du problème de fermeture nécessaire (# eval.) pour l'obtention de cette borne et les temps de calcul observés en secondes sur un ordinateur HP9000/735 (CPU) sur 10 instances d'un problème de 18 000 nœuds et 5 périodes.

Tableau 6.1 Comparaison des bornes supérieures obtenues par le modèle relaxé $RP1$ et le modèle original P sur des instances d'un problème de 18 000 nœuds et 5 périodes.

	Modèle relaxé ($RP1$)			Modèle non relaxé (P)		
Problèmes	Z_{sup}	# éval.	CPU	Z_{sup}	# éval.	CPU
Prob. 1	1821.72	77	227	1821.61	145	4113
Prob. 2	1859.19	69	226	1859.10	188	5443
Prob. 3	1840.99	70	219	1840.95	141	3483
Prob. 4	1872.66	77	256	1872.59	117	3416
Prob. 5	1853.04	70	221	1852.99	160	6222
Prob. 6	1870.69	69	238	1870.62	161	4565
Prob. 7	1828.07	70	217	1828.01	112	2996
Prob. 8	1849.78	67	200	1849.68	136	3746
Prob. 9	1827.64	74	251	1827.54	117	4100
Prob. 10	1861.11	74	266	1860.99	147	3992

Ce tableau montre que la qualité des bornes supérieures obtenues par la résolution du dual de la relaxation du problème ($RP1$) est quasiment la même que celle des bornes obtenues en résolvant le dual du problème original (P). L'utilisation de

la relaxation ($RP1$) est accompagné d'une économie considérable en temps de calcul. Pour ces problèmes ayant une petite taille, le tableau montre un temps moyen de traitement de 3 à 4 minutes pour ($RP1$) comparativement à un temps CPU de plus d'une heure pour le modèle original. Notons que le nombre d'évaluations (*# eval.*) représente pour le modèle relaxé ($RP1$) la somme du nombre d'évaluations des problèmes de fermeture maximale sur un graphe correspondant à une seule période et représente pour le modèle original (P), le nombre d'évaluations du problème de fermeture maximale sur un réseau dont la taille est 5 fois celle d'un graphe correspondant à une période.

À la lumière de ces résultats, il est prévisible que le calcul de la borne supérieure par la résolution du dual de (P) nécessitera de très grand temps de calcul pour les problèmes ayant une taille de 100 000 nœuds et plus. Ainsi nous avons opté pour la résolution de la relaxation ($RP1$) même si parfois elle peut fournir une borne de qualité moindre que celle qui aurait été trouvée via la résolution du dual complet. Notons que le dual de ($RP1$) présente deux niveaux de relaxation par rapport au dual du problème original (P). Les contraintes inter-périodes présentes dans (P) sont absentes dans ($RP1$) et le domaine des variables duales a été restreint dans le dual de ($RP1$).

La deuxième étape de cette expérimentation porte sur les résultats obtenus par l'approche de résolution proposée qui résout le dual de la relaxation ($RP1$) pour calculer une borne supérieure et utilise une heuristique pour déterminer une solution réalisable. Les tableaux 6.2 et 6.3 présentent ces résultats pour un ensemble de problèmes tests. Chaque tableau est subdivisé en trois parties. Chaque partie correspond à un groupe de problèmes obtenus en variant l'un des trois facteurs caractérisant les problèmes (nombre de nœuds, nombre de périodes et ratio r_m) et en

fixant les deux autres. Les résultats présentés sont des moyennes de 10 instances du même type de problème générées aléatoirement. Pour chaque type de problème, les tableaux présentent une moyenne sur la valeur de la borne supérieure Z_{sup} obtenue par la résolution de la relaxation ($RP1$), une valeur moyenne de la borne inférieure Z_{inf} qui représente la valeur de la meilleure solution réalisable obtenue par l'heuristique combinant la phase gloutonne et la phase tabu. Ils présentent aussi les gaps relatifs entre les deux bornes et les temps CPU moyens en secondes nécessaires à la résolution de chaque instance. Le tableau 6.2 présente ces résultats pour des problèmes où les valeurs sur les nœuds à la première période sont générées aléatoirement sans aucune structure tel que décrit au début de cette section. Nous désignerons cette catégorie de problèmes par la catégorie A dans la suite du texte. Le tableau 6.3 présente des résultats semblables pour un ensemble de problèmes tests présentant une structure sur les valeurs des nœuds de façon à rendre moins attrayants la sélection des nœuds plus profonds du graphe. En effet les graphes générés sont de même nature que ceux rencontrés dans les applications minières, chaque nœud est caractérisé par son niveau dans le graphe, les nœuds de niveau 1 correspondent aux nœuds sans successeur dans le graphe. Cette structure est mise en œuvre en multipliant les valeurs générées aléatoirement pour chaque nœud par α si la valeur est négative et par $1-\alpha$ si la valeur est positive. Le paramètre α désigne le rapport du niveau du nœud dans le graphe par le nombre total de niveaux dans le graphe. La présence de cette structure sur les valeurs des nœuds d'un graphe le rend plus conforme aux graphes représentant un gisement minier dont les valeurs sur les nœuds dépendent fortement des coûts de transport qui sont fonction du niveau du bloc dans le gisement. Cette catégorie de problèmes sera désignée par la catégorie B dans la suite du texte.

Ces deux catégories de problèmes présentent des caractéristiques différentes qui

conditionnent le comportement de l'heuristique proposée pour la recherche des solutions réalisables. Cette heuristique cherche à rendre admissible les solutions fournies par le problème du lagrangien lors de l'optimisation duale. Son comportement est fortement conditionné par la qualité de ces solutions. Dans un processus de relaxation lagrangienne, la qualité des solutions fournies par le problème lagrangien (la qualité est mesurée en terme de la satisfaction des contraintes dualisées) dépend de deux facteurs principaux: (1) la densité des solutions possibles dans un voisinage de la solution optimale recherchée, et (2) la courbure de la fonction objectif en fonction des seconds membres des contraintes dualisées. La présence de ces facteurs signifie que l'objectif en fonction des seconds membres des contraintes dualisées approche une fonction strictement concave. Cette fonction est par nature discrète donc ne peut être définie strictement concave mais en présence d'une bonne densité de solutions et une certaine courbure, elle peut approcher une fonction strictement concave.

Le premier facteur augmente les chances de l'obtention d'une solution du lagrangien, une fois les multiplicateurs bien ajustés, proche de la solution optimale recherchée. Le deuxième facteur contribue à la stabilité du comportement du lagrangien. Dans une situation où la notion de proximité de la stricte concavité est absente, une petite modification des multiplicateurs de Lagrange peut changer radicalement la solution du lagrangien.

Sur des problèmes dont l'objectif en fonction des seconds membres des contraintes dualisées approche une fonction strictement concave, l'heuristique que nous utilisons est efficace car elle agit sur des vecteurs solutions proches du vecteur solution optimal recherché. Elle prend ainsi moins de risque de s'en éloigner en cherchant à rendre la solution admissible.

La présence du premier facteur sur notre type de problème est difficile à mettre en évidence, néanmoins leur structure laisse croire qu'aux alentours d'une solution optimale il devrait y avoir une bonne densité de solutions. Pour mettre en évidence le deuxième facteur, nous avons considéré un problème de 18 000 nœuds et 5 périodes en présence de 3 contraintes de ressource par période. Nous considérons que les disponibilités des ressource sont identiques pour tous les types de ressources et pour toutes les périodes. En représentant l'évolution de la valeur de la borne lagrangienne en fonction de la disponibilité des ressources (second membre des contraintes dualisées) pour un problème de chacune des catégories A et B, on obtient respectivement les figures 6.3 et 6.4. On note que pour la catégorie B, la courbe de la figure 6.4 approche une courbe strictement concave; par contre la courbe de la figure 6.3 qui correspond aux problèmes de la catégorie A présente un tronçon linéaire couvrant une grande plage des disponibilités. Les résultats présentés dans les tableaux 6.2 et 6.3 respectivement pour les problèmes de la catégorie A et ceux de la catégorie B reflètent les conséquences de l'observation précédente.

Le tableau 6.3 montre que pour les différents types de problèmes de la catégorie B, les gaps entre la borne lagrangienne et la valeur de la solution réalisable obtenue par l'heuristique combinant la phase gloutonne et la phase tabu sont relativement petits. Ces gaps moyens se situent entre 0.2 et 0.3%, témoignant ainsi de l'efficacité de l'heuristique sur cette catégorie de problèmes. Le tableau 6.2 montre des gaps moyens relativement élevés, de l'ordre de 2% à 3%, pour les différents types de problème de la catégorie A, ce qui montre la difficulté pour l'heuristique de trouver de bonnes solutions réalisables pour cette catégorie de problèmes.

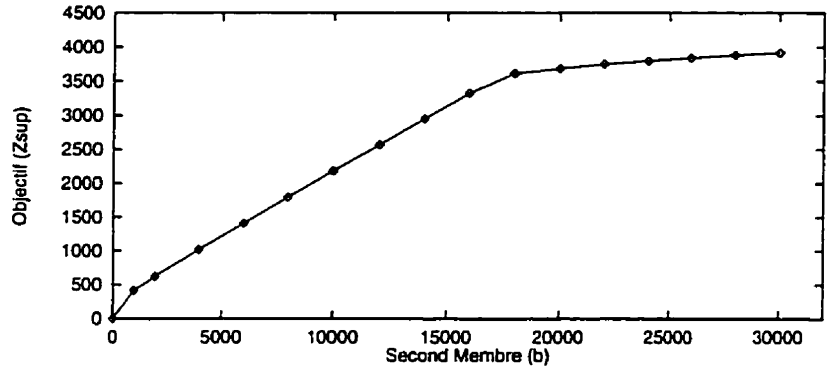


Figure 6.3 Caractéristique des problèmes de la catégorie A.

Tableau 6.2 Résultats obtenus sur des problèmes tests de la catégorie A

	Z_{sup}	Z_{inf}	Gap(%)	CPU(secs)		
				Flot Max.	Heur.	Total
Nbr. Nœuds	Nbr. Périodes = 5 , r_m = 1500					
20 000	1992.43	1960.46	1.60	553	120	714
50 000	1992.07	1922.25	3.51	1379	547	1975
80 000	1999.87	1947.33	2.65	1960	742	2755
100 000	2136.16	2092.65	2.04	2135	704	2900
r_m	Nbr.Périodes = 5 , Nbr. Nœuds = 100 000					
1500	2153.32	2109.07	2.06	2173	729	2959
3000	3591.57	3474.66	3.25	7865	733	8660
4500	5051.10	4874.68	3.49	10217	803	11088
6000	6467.91	6246.76	3.42	9903	702	10673
Nbr. Périodes	r_m = 1500 , Nbr. Nœuds = 100 000					
5	2147.02	2101.52	2.12	1987	713	3756
8	2720.39	2619.87	3.69	7097	1019	8209
10	3014.35	2910.16	3.45	13166	1008	14283
13	3255.45	3149.12	3.27	15122	1205	16448

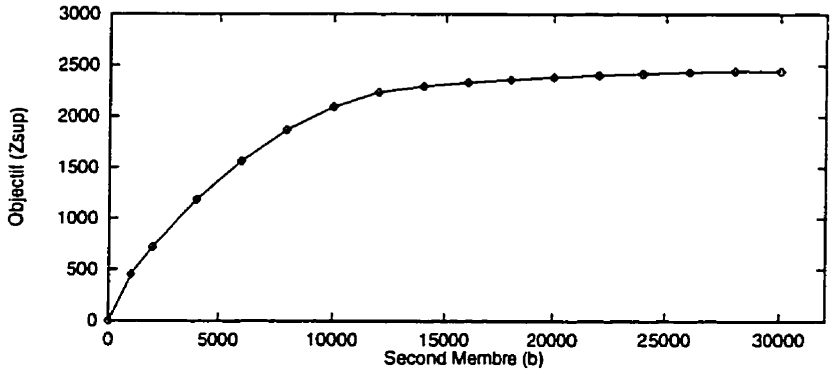


Figure 6.4 Caractéristique des problèmes de la catégorie B.

Tableau 6.3 Résultats obtenus sur des problèmes tests de la catégorie B

	Z_{sup}	Z_{inf}	Gap(%)	CPU(secs)		
				Flot Max.	Heur.	Total
Nbr. Nœuds	Nbr. Périodes = 5 , r_m = 1500					
20 000	2094.15	2092.28	0.09	222	106	342
50 000	2412.65	2406.11	0.27	661	310	1005
80 000	2505.08	2497.39	0.31	1203	661	1921
100 000	2630.57	2616.06	0.55	1148	596	1811
r_m	Nbr.Périodes = 5 , Nbr. Nœuds = 100 000					
1500	2641.32	2634.43	0.26	1191	489	1765
3000	4603.38	4594.42	0.20	1527	794	2403
4500	6288.36	6278.70	0.15	1833	916	2832
6000	7813.81	7804.50	0.12	2065	805	2957
Nbr. Périodes	r_m = 1500 , Nbr. Nœuds = 100 000					
5	2651.99	2644.26	0.29	1188	570	1826
8	3441.02	3428.49	0.36	2264	1154	3521
10	3838.42	3823.55	0.39	2936	1438	4494
13	4293.07	4274.53	0.43	4087	1880	6112

Étant donné que l'heuristique proposée n'a pas été efficace sur les problèmes de la catégorie A, nous avons, dans cette dernière partie des tests, comparé les valeurs des solutions réalisables obtenues en utilisant deux versions de la phase gloutonne de l'heuristique. La phase gloutonne *Heur. Greedy(1)* est celle présentée dans le texte et qui retire un à la fois les nœuds de la fermeture à rendre réalisable. La phase gloutonne *Heur. Greedy(2)* est une version qui permet de retirer un groupe de nœuds à la fois. Si on considère S l'ensemble des nœuds de la fermeture de départ et $Front(S) = \{i \in S \ : \ \forall i \in \Gamma_i^{-1} \ i \notin S\}$, en posant $F^1 = FRONT(S)$ et $F^2 = FRONT(S \setminus F^1)$, nous considérons les groupes de nœuds constitué de un nœud dans F^2 et de ses prédécesseurs qui sont dans F^1. Il s'agit de la plus simple des stratégies de sélection du groupe de nœuds à enlever de la fermeture. Cette stratégie pourrait être sophistiquée en considérant plus de deux niveaux dans le graphe pour choisir le groupe de nœuds à éliminer. À cause de la combinatoire, le nombre de groupes de nœuds possibles devient très élevé. Ainsi l'élaboration d'un algorithme de sélection est nécessaire car ils ne peuvent être tous essayés. Pour les tests de cette stratégie simple, nous considérons 10 instances d'un problème de la catégorie A présentant 18 000 nœuds, 5 périodes et un ratio r_m de 1500. Le tableau 6.4 présente la valeur des meilleures solutions réalisables Z_{inf} et les temps CPU en secondes obtenus sur les 10 instances par chacune des approches.

Ce tableau montre que *Heur. Greedy (2)* obtient des meilleures bornes que *Heur. Greedy (1)* mais elle nécessite un temps de calcul nettement plus élevé. Les temps de calcul élevés sont dûs au fait qu'à chaque élimination d'un groupe de nœuds, les ensembles F^1 et F^2 sont reconstruits. Une mise à jour dynamique de ses ensembles devrait diminuer ces temps de calcul. Nous n'avons pas réalisé cette implémentation ni ajouté la ré-optimisation avec la méthode tabu car la traitement des problèmes de la catégorie A n'est pas l'objectif principal de cette recherche.

Tableau 6.4 Comparaison des stratégies de l'heuristique sur des instances d'un problème de la catégorie A de 18 000 nœuds et 5 périodes.

Problèmes	Heur. Greedy(1)		Heur. Greedy(2)		
	Z_{inf}	CPU(s)	Z_{inf}	CPU(s)	Z_{sup}
Prob. 1	1617.99	10	1659.93	2106	1716.87
Prob. 2	1499.96	10	1610.08	2260	1693.88
Prob. 3	1472.31	10	1639.55	2086	1692.38
Prob. 4	1341.04	10	1619.17	2168	1684.27
Prob. 5	1606.74	9	1612.21	2192	1754.75
Prob. 6	1470.03	13	1492.72	3560	1655.18
Prob. 7	1541.73	9	1559.44	2350	1708.79
Prob. 8	1515.96	10	1531.38	2706	1670.70
Prob. 9	1553.95	8	1653.84	2056	1746.10
Prob. 10	1533.72	11	1601.36	2779	1681.89

Ce tableau montre toutefois qu'une bonne implémentation de l'heuristique *Heur. Greedy (2)* pourrait donner de bons résultats sur les problèmes de la catégorie A.

6.5 Application à la Planification minière

Nous avons considéré le gisement de fer du Mont Wright exploité par la compagnie Québec Cartier Mining. Ce gisement est étalé sur une étendue de $4.8par2.7km$ avec une profondeur de $600m$. Le modèle de blocs représentant ce gisement est composé de 44 niveaux de 14 m chacun. Chaque niveau est discrétisé en blocs de dimension $10m$ par $10m$ par $14m$.

En tenant compte de la topographie en date du 03 mai 1995, le modèle de blocs comprend 132 525 blocs minéralisés et 4 256 504 blocs non minéralisés. Tous les blocs avec une teneur en fer non nulle ont été considérés comme des blocs minéralisés. Les blocs les plus pauvres auront une valeur négative avec la structure de coût utilisée.

Ce modèle ne peut être traduit sous forme de graphe à cause du nombre élevé de blocs de stérile. Ainsi, nous avons réduit le nombre de blocs dans le modèle en prenant en considération uniquement les blocs minéralisés et les blocs non minéralisés qui contraignent l'extraction des blocs minéralisés. Nous supposons qu'un bloc est contraint par le bloc au dessus et les huit voisins immédiats de ce dernier du même niveau. Les blocs non minéralisés qui contraignent par transitivité l'extraction de blocs minéralisés sont aussi inclus. Cette réduction nous amène à considérer pour fin d'optimisation un modèle constitué de 132 525 blocs minéralisés et 14 834 blocs non minéralisés. Le modèle renseigne, pour chaque bloc, sur les coordonnées, la teneur en fer et le pourcentage de la récupération en poids. Ces informations nous permettent de déterminer, pour chaque bloc, le tonnage de minerai et le tonnage en concentré. Nous associons à chaque bloc dans ce modèle le revenu qui peut être positif ou négatif généré à la suite de son exploitation à la première année de l'horizon de planification:

$$c_i^1 = vQ_i^c - c_tQ_i^t - c_mQ_i^m - c_rQ_i^r$$

où :

Q_i^c : tonnage en concentré du bloc i;

Q_i^t : tonnage traité du bloc i qui vaut Q_m si le bloc est minéralisé et 0 sinon;

Q_i^m : tonnage en minerai du bloc i, fonction de sa teneur t_i;

Q_i^r : tonnage en minerai du bloc i multiplié par la distance de transport;

v : prix de vente d'une tonne de concentré en retranchant les coûts de commercialisation et de livraison;

c_t : coût de traitement d'une tonne de minerai;

c_m : coût d'exploitation d'une tonne de matériel;

c_r : coût de transport d'une tonne de matériel sur un km.

Nous déterminons les revenus des blocs pour les périodes suivantes de l'horizon de

planification en considérant la dépréciation financière comme suit:

$$c_i^p = c_i^1/(1+d)^p$$

où : p est l'indice de la période et d est le taux d'intérêt.

Dans cette application, nous avons considéré 3 types de contraintes de ressource pour chaque période.

- Le tonnage traité doit respecter la capacité de traitement des installations; ainsi, la quantité traitée à l'année p doit être inférieure à la capacité de traitement annuel des installations, $Q^{t,p}$.
- Le transport de tout le matériel à l'année p doit respecter la disponibilité du matériel de transport à la même année. $Q^{r,p}$ représente la borne sur la capacité de transport durant l'année p.
- La teneur moyenne des blocs minéralisés extraits à la période p doit être inférieure ou égale à une valeur prédéterminée t_f^p. Une borne inférieure sur la teneur doit être considérée, mais en pratique l'objectif du maximum de bénéfice porte à extraire d'abord les blocs les plus riches, ainsi seule la contrainte de borne supérieure sur la teneur est active.

Le problème de planification de l'exploitation dans cette mine consiste à déterminer, pour chaque période de l'horizon de planification, l'ensemble des blocs à exploiter de façon à maximiser le revenu total sur l'horizon et à respecter pour chaque période, l'ensemble des contraintes de ressource et les contraintes physiques qui imposent qu'un bloc doit être à découvert avant d'être exploité. Les contraintes physiques pour ce gisement se traduisent par un réseau multi-périodes. À chaque

période de ce réseau correspond un graphe de 147 359 nœuds.

En définissant les variables de décision suivantes:

$$x_i^p = \begin{cases} 1 & \text{si le bloc } i \text{ est extrait à la période } p \text{ ou avant,} \\ 0 & \text{sinon,} \end{cases}$$

le problème de la planification de l'exploitation se formule comme le problème (P) de la section 6.2 où les contributions des nœuds a_{ki} $(k = 1, 2, 3)$ aux contraintes sont représentées par respectivement Q_i^t, Q_i^r et $t_i - t_f^p$ alors que les bornes b_k^p $(k = 1, 2, 3)$ des contraintes sont représentées par respectivement Q_p^t, Q_p^r et 0.

Pour effectuer les tests, nous avons considéré les paramètres du modèle de coût suivants: $v = 1.8$, $c_m = 0.1$, $c_t = 0.3$ et $c_r = 0.2$.

Tableau 6.5 Problème QCM en fonction du nombre de périodes

Nbr. Périodes	Z_{sup}	Z_{inf}	$Gap(\%)$	CPU
6	94 069.6	92 775.1	1.38	3h28min
8	112 673.7	111 640.8	0.92	4h37min
10	124 029.5	122 462.9	1.26	5h08min

Nous présentons dans le tableau 6.5 les résultats obtenus en considérant des horizons allant de 6 à 10 périodes. Ce tableau donne la borne supérieure obtenue par la résolution de la relaxation $(RP1)$ du duale et la borne inférieure qui correspond à la solution réalisable trouvée par l'heuristique. Il montre aussi le gap entre ces deux bornes et le temps CPU d'exécution sur une machine HP9000/735. Ce tableau montre un gap de moins de 1.5%, ce qui signifie une erreur du même ordre sur l'optimalité. Vu l'incertitude sur les informations concernant les contenus des blocs et leur modèle de valorisation, les données des problèmes miniers présentent

Tableau 6.6 Contributions aux contraintes dans le problème QCM 10 périodes

Période p	Traitement 10^3 Tonne	Transport 10^3 Tonne*Km	Teneur moyenne (%)	Nbr. Blocs exploités
1	24 991.79	10 362.26	34.00	12120
2	23 764.15	12 148.62	34.00	12104
3	23 193.42	13 299.66	33.97	11861
4	23 869.50	14 449.96	34.00	11917
5	24 404.01	15 599.99	34.00	12610
6	24 592.61	16 749.79	34.00	12487
7	24 088.96	17 899.99	34.00	12180
8	21 850.99	18 049.95	34.00	10932
9	18 208.03	19 199.99	32.39	12062
10	20 471.74	20 349.94	33.37	12121
Disponibilité	24 991.80	$11000 + 1150(p-1)$	34.00	-

des erreurs. Tenant compte de l'existence de ces erreurs, le résultat précédent est très satisfaisant en pratique pour ces problèmes. Les temps d'exécution pour ce problème varient de 3 à moins de 6 heures pour un nombre de périodes allant de 6 à 10. Vu la taille de ces problèmes qui est de l'ordre de 1,400,000 variables et de 14,000,000 de contraintes pour le problème de 10 périodes, et le fait que ces problèmes sont des problèmes de planification qui se résolvent habituellement peu de fois durant une période d'une année, l'ordre de ces temps de calcul est très satisfaisant en pratique.

Le tableau 6.6 présente la contribution aux contraintes de ressource dans la solution réalisable retenue pour le problème de 10 périodes. Ce tableau montre que sur trois contraintes de ressource par période, la solution obtenue satisfait à égalité deux contraintes par période. Pour la première période, elle sature la contrainte de traitement et la contrainte de teneur et pour les autres périodes, elle sature la contrainte de transport et la contrainte de teneur.

Pour ce problème, nous avons considéré une disponibilité en transport croissante avec les périodes et une même disponibilité en traitement durant toutes les périodes. L'essai d'autres scénarios est d'une grande importance pour la prise de décisions stratégiques telles que la disponibilité du matériel de transport, l'extension des installations de traitement, etc.

6.6 Conclusion

Le problème traité dans ce chapitre est un problème combinatoire de très grande taille. Vu sa structure, la seule approche qui semble appropriée à sa résolution est la relaxation lagrangienne des contraintes de ressource qui le ramène à un problème de fermeture maximale classique sur un réseau multi-périodes. En pratique, ce réseau ne peut être traité directement par un algorithme de détermination d'une fermeture à cause de sa taille. Nous avons ainsi relaxé toutes les contraintes de préséance inter-périodes afin d'avoir à résoudre plusieurs problèmes de fermeture sur des graphes de taille acceptable correspondant aux graphes de chacune des périodes.

Ce schéma de relaxation lagrangienne nous permet de calculer une borne supérieure sur la solution optimale du problème original. A ce schéma, nous avons incorporé une heuristique simple qui permet de convertir les solutions du lagrangien en des solutions réalisables au problème original. Les résultats obtenus sur une application en planification minière montrent des gaps relatifs entre la solution réalisable retenue et la borne supérieure obtenue par relaxation lagrangienne de moins de 1.5%. Ce résultat est très satisfaisant en pratique, étant donné l'ordre de grandeur des erreurs sur les données des gisements miniers.

Actuellement, aucun outil basé sur une approche d'optimisation n'est disponible

dans les industries minières pour le traitement complet de ce problème. L'approche que nous proposons dans cette thèse constitue une première méthode à erreur mesurable qui réussit à considérer le problème dans sa globalité et qui permet de déterminer de bonnes solutions réalisables en un temps de calcul raisonnable.

CONCLUSION

Dans cette thèse, une approche de résolution efficace est développée pour le problème de planification multi-périodes dans une exploitation minière à ciel ouvert. Cette approche est la première basée sur l'optimisation permettant de traiter le problème dans toute sa complexité. Ce développement permettra aux exploitants miniers de se doter d'un outil de planification qui peut être utilisé à long, moyen ou court terme dépendemment de l'horizon considéré et du type de contraintes de ressource modélisées. Cet outil sera capable de construire des plans d'exploitation qui prennent en considération les contraintes de fonctionnement et d'exploitation et surtout de la variation des revenus des blocs en fonction du temps. Ce fait constitue une innovation majeure dans le domaine sachant que les méthodes en utilisation dans les industries minières ne tiennent compte ni des contraintes de ressource, ni de la variation des revenus en fonction du temps.

L'approche de résolution proposée permet de quantifier la qualité des solutions obtenues. Une mesure du gap entre la valeur de la borne supérieure obtenue par relaxation lagrangienne et la valeur de la solution primale retenue permet de quantifier de combien la solution proposée peut être éloignée de l'optimum recherché. Cette approche de résolution a été testée sur les données d'une exploitation minière qui présente 150 000 blocs. En considérant un horizon de planification de 10 périodes et 3 contraintes de ressource (transport, traitement et teneur) par période, les solutions obtenues présentent des gaps de l'ordre de 1% par rapport à l'optimalité.

Le développement de cette approche de résolution pour le problème de planification multi-périodes est basé sur les résultats obtenus dans cette thèse pour le problème des contours et celui du design de fosses avec contraintes de ressource.

Ces deux problèmes sont des cas particuliers du problème global de planification multi-périodes.

Le problème des contours est un problème de fermeture maximale sur un graphe et il se résout par un algorithme de flot maximum. Une étude approfondie des différents algorithmes de flot maximum existants est présentée dans cette thèse. Des améliorations pratiques tenant compte de la structure des graphes rencontrés dans les applications minières sont proposées. Ces améliorations nous ont permis d'arriver à une implémentation permettant de traiter ce type de problème sur des graphes de plus de 100,000 nœuds et 1 million d'arcs en moins de 15 secondes CPU sur un ordinateur HP9000/735.

Le problème du design de fosses avec contraintes de ressource est vu dans cette thèse comme un problème de fermeture maximale sur un graphe en présence de contraintes de ressource. Pour sa résolution, nous avons adopté la relaxation lagrangienne des contraintes de ressource pour se ramener à un problème de fermeture maximale classique qu'on sait résoudre efficacement. L'ajustement des multiplicateurs de Lagrange est effectué par la méthode de faisceaux qui s'est avérée la plus appropriée comparativement à la méthode du sous-gradient et la méthode des plans sécants. Cette approche nous fournit une borne supérieure sur la solution optimale. Une borne inférieure est déterminée par l'utilisation d'une heuristique simple qui a été développée pour la recherche de solutions réalisables. Le gap entre ces deux bornes constitue une mesure de la qualité des solutions réalisables retenues. L'innovation de cette approche par rapport aux tentatives connues dans la littérature réside dans l'utilisation d'une méthode performante d'ajustement des multiplicateurs et l'utilisation d'un algorithme adapté pour la détermination d'une fermeture maximale sur un graphe. La combinaison de ces deux facteurs a permis

de traiter pour la première fois des problèmes de taille semblable à ceux rencontrés en pratique. Les solutions retenues pour ces problèmes présentent de petits gaps et sont obtenues en des temps machine relativement courts.

A la suite de ce travail, plusieurs axes futurs de recherche se sont dessinés. Ces axes portent sur des problématiques rencontrées dans le développement de cette thèse et sur d'autres pouvant mener à des améliorations de la qualité de l'approche proposée. Trois axes de recherche ont été mis en évidence.

- Le problème de fermeture maximale sur les types de graphe rencontrés dans les applications minières est résolu par un algorithme de flot maximum. En pratique nous sommes arrivés à des implémentations qui traitent un problème de 100,000 nœuds et un million d'arcs en moins de 15 secondes CPU. Vu la structure de ces graphes, est-il possible de déterminer une meilleure borne sur la complexité théorique de ces algorithmes s'ils sont appliqués à ce type de graphe? Un résultat dans ce sens permettra d'adapter ces algorithmes aux graphes miniers.

- La relaxation des contraintes de ressource dans le problème de planification multi-périodes conduit à un problème de fermeture maximale sur un réseau multi-périodes qui ne peut être pris en mémoire d'un ordinateur à cause de sa taille. Une tentative de développement d'une méthode de décomposition pour ce problème a été présentée au chapitre 3 de cette thèse. Les résultats obtenus ne sont pas complètement satisfaisants pour permettre l'utilisation de cette décomposition dans le processus d'optimisation. Nous avons contourné cette difficulté en relaxant les contraintes inter-périodes dans ce réseau afin de se ramener à plusieurs graphes indépendants de taille raisonnable. Chacun de ces graphes correspond à une période. Nous avons tenu compte de ces

contraintes relaxées dans les heuristiques de recherche de solutions réalisables. Le développement d'une décomposition optimale est une problématique non résolue qui peut améliorer la méthode de résolution proposée et qui est d'une grande valeur théorique.

- Au chapitre 6, nous avons présenté l'heuristique de construction de solutions réalisables utilisée à chaque itération de l'optimisation duale. Cette heuristique a été moins efficace sur une certaine catégorie de problèmes générés aléatoirement, à cause de son caractère de traitement local. Des idées d'améliorations simples ont été proposées. Ces améliorations visent à donner à cette heuristique un comportement plus global. Pour la mise en œuvre efficace de ces idées, une recherche plus approfondie est nécessaire.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] AHUJA, R.K. & ORLIN, J.B. (1989). A fast and simple algorithm for the maximum flow problem. *Operations Reseach*, **37(5)**.

[2] AHUJA, R.K., MAGNANTI, T.L. & ORLIN, J.B. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall.

[3] BARNES, R.J. & JOHNSON, T.B. (1980). New ideas in the optimum ultimate pit limit problem bounding the optimum. *Proceedings of the Mining Computer Applications Symposium*, Moscow, U.S.S.R., 20-25.

[4] BRATICEVIC, D.B. (1984). Open pit optimization method. *Proceedings of the 18th APCOM Symposium*, 133-137.

[5] CHERKASKY, B.V. (1977). An algorithm for construction of maximal flows in networks with complexity $O(v^2E^{1/2})$ operations. *Mathematical Methods for the Solutions of Economic Problem (en russe)*, **7**, 117-125.

[6] CRAWFORD, J.T. (1979). Open pit limit analysis - Some observation on its use. *Proceedings of the 16th APCOM Symposium*, 625-634.

[7] DAGDELEEN, K. (1985). *Optimum multi-period open pit mine production scheduling.* Thèse de doctorat, Colorado School of Mines, Golden, USA.

[8] DAGDELEEN, K.& JOHNSON, T.B. (1986). Optimum open pit mine production scheduling by lagrangian parameterization. *Proceedings of the 19th APCOM Symposium*, 127-141.

[9] DANTZIG, G.B. & WOLFE, P. (1960). The decomposition algorithm for linear programming. *Operations Research*, **8**.

[10] DAVIS, R.E. & WILLIAMS, C.E. (1973). Optimization Procedures for Open Pit M ine Scheduling. *11th APCOM Symposium*, University of Arizona, Tucson, U SA.

[11] DINIC, E.A. (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Mathematics Doklady 11*, 1277-1280.

[12] EDMONDS, J. & KARP, R.M. (1972). Theoretical improvements in algorithm efficiency for network flow problems. *JACM*, **19**, 248-264.

[13] ELEVLI, E., DAGDELEEN, K. & SALAMON, D.G. (1989). Single time period production scheduling of open pit mines. *SME-AIME Meetings*, Las Vegas, Nevada.

[14] FORD, L.R. & FULKERSON, D.R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, **8**, 399-404.

[15] FORD, L.R. & FULKERSON, D.R. (1957). A Simple algorithm for finding maximal network flows and an applications to the Hitchcock problem. *Canadian Journal of mathematics*, **9**, 210-218.

[16] GLOVER, F. (1989). Tabu-Search Part I. *ORSA Journal on Computing* **1**, 190–206.

[17] GLOVER, F. (1990). Tabu-Search Part II. *ORSA Journal on Computing* **2**, 4–32.

[18] HIRIART-URRUTY, J.& LEMARECHAL, C. (1993). *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. A series of Comprehensive Studies in Mathematics, Springer-Verlag.

[19] GALIL, Z. (1980). An $O(V^{5/3}E^{2/3})$ algorithm for the maximal flow problem. *Acta Informatica*, **14**, 221-242.

[20] GERSHON, M. (1988). An open pit planning and scheduling system. *Computer Applications in the Mineral Industry*, Fytas, Collins & Luiglial (eds), Balkina, Rotterdam.

[21] GOLDBERG, A.V. (1985). *A new max-flow algorithm.* Technical Report MIT/LCS/TM-291, Laboratory of Computer Science, M.I.T., Cambridge, MA., USA.

[22] GOLDBERG, A.V. & TARJAN, R.E. (1986). A new approach to the maximum flow problem. *Proceedings. 18th ACM Symposium on the Theory of Computing*, 136-146.

[23] JENSEN, P. A. & BARNES, J. W. (1980). *Network flow programming.* John Wiley & Sons.

[24] JOHNSON, T.B. (1968). *Optimum open pit mine production scheduling.* Thèse de Doctorat, Dept. of Ind. Eng. and Op. Res. University of California, Berkeley.

[25] JOHNSON, T.B. & SHARP, W.R. (1971). *A three dimmensional dynamic programming method for optimal open pit design.* U.S. Bureau of Mines, Report of Investigation 7553.

[26] KARZANOV, A.V. (1974). Determining the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady*, **15**, 434-437.

[27] KELLEY, J.E. (1960). The cutting plane method for solving convex programs. *SIAM Journal*, **8**, 703–712.

[28] KOENIGSBERG, E. (1982). The optimum contours of an open pit mine : an application of dynamic programming. *Proceedings of the 17th APCOM Symposium*, 274-287.

[29] KIWIEL, K.C. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, **46(1)**, 105-122.

[30] KOROBOV, S. (1974). *Methods for determining optimal open pit limits.* Dept. of Mineral Engineering, École Polytechnique de Montréal, Paper ED-74-R-4.

[31] LEMARECHAL, C., NEMIROVSKI, A.S., & NESTEROV, Y. (1991). *New variant of bundle methods and applications.* Report RR, INRIA.

[32] LEMARECHAL, C., STRODIOT, J.J. & BIHAIN, A. (1981). On a bundle algorithm for nonsmooth optimization. *Nonlinear Programming J.*, O.L. Mangasarian, R.R. Meyer and S.M. Robinson (Eds), Academic Press, 245–282.

[33] LEMIEUX, M. (1979). Moving cone optimizing algorithm. *Proceedings of the APCOM symposium, computer methods for the 80's*, 329-345.

[34] LERCHS & GROSSMAN, (1965). Optimum design of open pit mines. *C.M.I. Bulletin*, **58 no.633**, 47-54.

[35] MARINO, J.M. & SLAMA, J.P. (1973). Ore reserve evaluation and open pit planning. *Proceedings of the 10th APCOM Symposium*, South Africa Institute of Mining and Metallurgy, Johannesburg, S.A.

[36] PHILIPS, D.A. (1973). Optimum design of an open pit. *Proceedings 10th APCOM Symposium*, S.A.I.M.M., Johannesburg, S.A.

[37] PICARD, J.C. (1976). Maximal closure of a graph and applications to combinatorial problem. *Management Science*, **22 no.11**.

[38] PICARD, J.C. & QUEYRANNE, M. (1982). Selected Applications of Minimum Cuts in Networks. *INFOR*, **20**, 394–422.

[39] SCHRAMM, H. & ZOWE, J. (1992). A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, **2(1)**, 121–152.

[40] SHILOACH, Y. & VISHKIN, U. (1982). An $O(n^2log(n))$ parallel max flow algorithm. *J. Algorithms*, **3**, 128-146.

[41] SLEATOR, D.D. (1980). *An* $O(nm\log(n))$ *algorithm for maximum network flow.* Thèse de Doctorat, Stanford University.

[42] TACHEFINE, B. (1991). *Détermination du plan d'extraction d'une mine à ciel ouvert.* Mémoire de M.Sc.A., École Ploytechnique de Montréal, Canada.

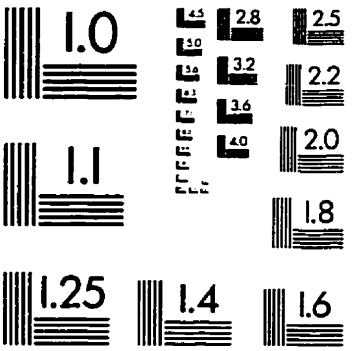
[43] TACHEFINE, B. & SOUMIS, F. (1995). Maximal Closure on a Graph with Resource Constraints. *Computers and Operations Research*, (to appear).

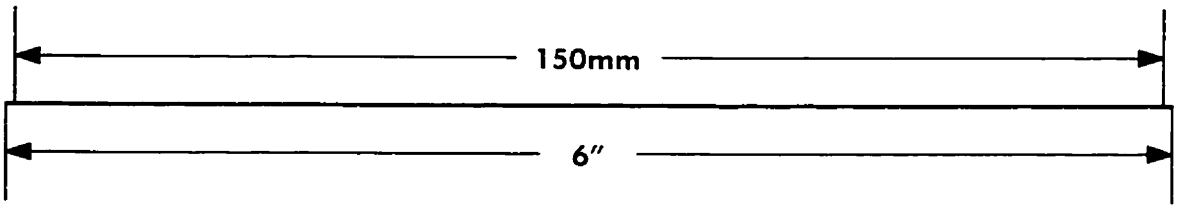
[44] TARJAN, R.E. (1984). A simple version of Karzanov blocking flow algorithm. *Operations Reseach Letters*, **2**, 265-268.

[45] ZHAO, Y. & KIM, Y.C. (1993). Optimum mine production sequencing using lagrangian parameterization approach. *Proceedings of the 24th APCOM Symposium*, **2**, 176-189.

[46] ZHAO, Y. & KIM, Y.C. (1990). A new graph theory algorithm for optimal ultimate pit design. *SME-AIME Meetings*, Salt Lake City, Utah.

TEST TARGET (QA-3)





APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989